

Sumário

1 INTRODUÇÃO.....	4
2 JUSTIFICATIVA.....	5
3 ANÁLISE DE REQUISITOS.....	6
4 OBJETIVOS.....	7
4.1 OBJETIVO GERAL.....	7
4.2 OBJETIVOS ESPECÍFICO.....	7
5 FERRAMENTAS UTILIZADAS.....	8
5.1 DELPHI.....	8
5.2 FIREBIRD.....	9
5.3 SCANNER.....	9
5.4 DELPHI TWAIN.....	11
5.5 WINISIS/MICROISIS.....	11
5.5.1 Surgimento do ISIS.....	12
5.5.2 Vantagens.....	14
5.5.3 Características do CDS/ISIS.....	16
5.5.4 Definições complementares.....	17
5.5.5 Dados técnicos.....	17
5.6 ISIS DLL.....	18
6 FUNDAMENTAÇÃO TEÓRICA.....	19
6.1 GERENCIAMENTO ELETRÔNICO DE DOCUMENTOS.....	19
6.2 DADOS INTERESSANTES.....	20
6.3 DIGITALIZAÇÃO.....	20
7 CRONOGRAMA.....	22
7.1 CALENDÁRIO DAS ATIVIDADES.....	22
7.2 DIAGRAMA DE GANNT.....	22
8 MODELAGEM.....	23
8.1 DIAGRAMA DE CASOS DE USO.....	23
8.1.1 Criar Usuário.....	23
8.1.2 Definir local de armazenamento das imagens.....	23
8.1.3 Definir local do banco de dados.....	23
8.1.4 Criar campos.....	24
8.1.5 Criar Tipo de Documento.....	24
8.1.6 Vincular Campos com Tipo de Documento.....	24
8.1.7 Digitar Documentos.....	24
8.1.8 Digitalizar Documentos.....	24
8.1.9 Consultar documentos.....	25
8.2 CENÁRIOS.....	26
8.2.1 Cenário Criar Usuários.....	26
8.2.2 Cenário Definir local de armazenamento das imagens.....	27
8.2.3 Cenário Definir local do banco de dados.....	27
8.2.4 Cenário Criar Tipo de Documento.....	29
8.2.5 Cenário Vincular Campos com Tipo de Documento cadastrado.....	30
8.2.6 Cenário Digitar Documentos.....	31
8.2.7 Cenário Digitalizar Documentos.....	33
8.2.8 Cenário Consultar Documentos.....	34
8.3 DIAGRAMA DE TELAS/SEQUÊNCIA/COLABORAÇÃO.....	36

8.3.1 TELA FrmPrincipal.....	37
8.3.2 TELA FrmCadCampos.....	38
8.3.3 Diagrama de Sequência Cadastro de Novo Campo.....	39
8.3.4 Diagrama de Colaboração Cadastro de Novo Campo.....	40
8.3.5 Diagrama de Sequência Alteração de Campo.....	41
8.3.6 Diagrama de Colaboração Alteração de Campo.....	42
8.3.7 Diagrama de Sequência Exclusão de Campo.....	43
8.3.8 Diagrama de Colaboração Exclusão de Campo.....	44
8.3.9 TELA FrmPesquisaCampos.....	45
8.3.10 Diagrama de Sequência Pesquisa de Campos.....	46
8.3.11 Diagrama de Colaboração Pesquisar Campos.....	47
8.3.12 TELA FrmTipoDoc.....	48
8.3.13 Diagrama de Sequência Cadastro de Novo Tipo de Documento.....	49
8.3.14 Diagrama de Colaboração Cadastro de Novo Tipo de Documento.....	50
8.3.15 Diagrama de Sequência Alteração de Tipo de Documento.....	51
8.3.16 Diagrama de Colaboração Alteração de Tipo de Documento.....	52
8.3.17 Diagrama de Sequência Exclusão de Tipo de Documento.....	53
8.3.18 Diagrama de Colaboração Exclusão de Tipo de Documento.....	54
8.3.19 TELA FrmVincularCampos.....	55
8.3.20 Diagrama de Sequência Vincular Campos.....	56
8.3.21 Diagrama de Colaboração Vincular Campos.....	57
8.3.22 TELA FrmCadDocumentos.....	58
8.3.23 Diagrama de Sequência Cadastro de Novo Documento.....	59
8.3.25 Diagrama de Sequência Alteração de Documento.....	61
8.3.26 Diagrama de Colaboração Alteração de Documento.....	62
8.3.27 Diagrama de Sequência Exclusão de Documento.....	63
8.3.28 Diagrama de Colaboração Exclusão de Documento.....	64
8.3.29 TELA FrmImagensDoc.....	65
8.3.30 TELA FrmDigitaliza.....	66
8.3.31 Diagrama de Sequência Cadastro de Novas Imagens do Documento.....	67
8.3.32 Diagrama de Colaboração Cadastro de Imagens do Documento.....	68
8.3.33 Diagrama de Sequência Alteração de Imagens do Documento.....	69
8.3.34 Diagrama de Colaboração Alteração de Imagens do Documento.....	70
8.3.35 Diagrama de Sequência Exclusão de Imagens do Documento.....	71
8.3.36 Diagrama de Colaboração Exclusão de Imagens do Documento.....	72
8.3.37 TELA FrmPesquisaDoc.....	73
8.3.38 Diagrama de Sequência Pesquisa de Documentos (Informando Código Registro).....	74
8.3.39 Diagrama de Sequência Pesquisa de Documentos (Informando Termos de Pesquisa).....	75
8.3.40 Diagrama de Colaboração Pesquisa de Documentos (Informando Código Registro).....	76
8.3.41 Diagrama de Colaboração Pesquisa de Documentos (Termos de Pesquisa).....	77
8.4 DIAGRAMA DE CLASSES.....	78
8.5 DIAGRAMA ENTIDADE RELACIONAMENTO.....	79
8.6 DIAGRAMA DE ESTADOS.....	80
8.7 DICIONÁRIO DE DADOS E CAMADA DE PERSISTÊNCIA.....	81
8.7.1 Tabela: Campo.....	81
8.7.2 Camada de Persistência TCampo.....	81
8.7.3 Tabela: Tipo_Documento.....	82
8.7.4 Camada de Persistência TTipoDoc.....	82

8.7.5 Tabela: Campo_Documento.....	82
8.7.6 Camada de Persistência TDocumento.....	83
8.7.7 Tabela : Banco de Dados MicroIsis.....	83
8.7.8 Camada de Persistência TDocumento.....	83
8.7.9 Tabela: Img_Documento.....	84
8.7.10 Camada de Persistência TImagem.....	84
8.7.11 Tabela: Users.....	85
8.7.12 Camada de Persistência TUsuario.....	87
8.8 SCRIPT DE CRIAÇÃO DO BANCO.....	88
REFERÊNCIA.....	91
CÓDIGO FONTE.....	92
program DigiDoc;.....	92
unit UCadCampos;.....	93
unit UCadDoc;.....	100
unit UClasse;.....	118
unit UConfiguracao;.....	148
unit UDigitaliza;.....	151
unit UDMFireBird;.....	156
unit UImagemDoc;.....	161
unit ULogin;.....	170
unit UPesquisaCampo;.....	175
unit UPesquisaDoc;.....	178
unit UPrincipal;.....	187
unit UTipoDoc;.....	193
unit UUsuarios;.....	198
unit UVincCampos;.....	204
Bibliotecas Externas Utilizadas.....	210
unit DelphiTwain;.....	211
unit DelphiTwainUtils;.....	271
unit Twain;.....	282

1 INTRODUÇÃO

Normalmente os documentos são arquivados em caixas que ficam em salas denominadas “Arquivo Morto”. Com o aumento de seu volume, localizá-los ficou muito difícil, perde-se muito tempo. Além de que, muitas vezes, alguns documentos não são encontrados. Ou quando encontrados estão muito danificado e seus dados ilegíveis.

Desta necessidade surgiu o DIGIDOC, um sistema que utiliza técnicas e princípios do Gerenciamento Eletrônico de Documentos e que visa permitir a digitalização e organização de documentos, facilitando seu arquivamento e sua localização.

2 JUSTIFICATIVA

A importância deste projeto consiste em disponibilizar um sistema que permita o gerenciamento de documentos, permitindo a criação e digitalização. Para que o usuário tenha essa liberdade de criar seu próprio documento, ele deve definir um conjunto de campos e vinculá-los ao documento. A partir deste momento os dados poderão ser incluídos e o documento digitalizado.

O sistema DIGIDOC poderá ser utilizado por diversos tipos de usuários, por exemplo, uma empresa/escritório/escola que devido a grande quantidade de documentos gostaria de uma forma prática de localizar os documentos armazenados em seu “arquivo morto”.

3 ANALISE DE REQUISITOS

Analisando o tema “Digitalização de Documentos”, identificamos que um documento nada mais é que um conjunto de campos, e estes contém informações que caracterizam tipos diferentes de documentos e possibilitam a recuperação dos mesmos.

Combinando essas informações com as definições dos sistemas de Gerenciamento de Documentos, percebemos que este sistema deve prover uma organização, controle, facilidade ao gerar novos documentos, armazenamento dos documentos que forem tratados.

Após estas análises iniciais, em conjunto com o orientador, definiu-se que os principais requisitos para sistema são:

- Criação da Campos;
- Criação de Tipos de Documento;
- Vinculação de Campos com os Tipos de Documento;
- Digitalização de Documentos;
- Pesquisa dos Documentos cadastrados;

4 OBJETIVOS

4.1 OBJETIVO GERAL

O objetivo geral do trabalho é projetar e implementar um sistema computacional que automatize o trabalho de digitalização de documentos e possibilite uma pesquisa prática aos documentos já digitalizados.

4.2 OBJETIVOS ESPECÍFICO

Desenvolver um sistema para armazenar informações sobre Documentos, como os de uma empresa, escritório, escola. O sistema deve permitir:

- Criação/Armazenamento de vários tipos de documento;
- Digitalização dos documentos;
- Pesquisa de documentos armazenados;
- Impressão do documento digitalizado;
- Possibilitar vários perfis de acesso para os usuários;

5 FERRAMENTAS UTILIZADAS

Para o desenvolvimento e modelagem do projeto foram utilizadas as seguintes ferramentas:

- Desenvolvimento: Delphi 7;
- Banco de dados: WinISIS 1.5.3, Firebird 1.5.3;
- API para manipular a Base ISIS: ISIS DLL 7;
- Componente Twain: Delphi Twain;
- Gerenciamento de banco de dados: IBExpert Version 2006.10.14;
- Criação de Cronogramas: MS Project 2000;
- Modelagem de Dados: Model Maker 9;
- Relatórios, planilhas e apresentações: OpenOffice 2.1 e MSOffice 2003;
- Mensagens instantâneas: Windows Live Messenger e Google Talk;
- Scanner Genius Color Page-HR7XE;

Abaixo estamos abordando as principais tecnologias utilizadas no sistema.

5.1 DELPHI

Delphi é um compilador e uma IDE para o desenvolvimento de softwares. Ele é produzido pela Borland Software Corporation (por algum tempo chamada Inprise). A linguagem utilizada pelo Delphi, o Object Pascal (Pascal com extensões orientadas a objetos) a partir da versão 7 passou a se chamar Delphi Language. O Delphi originalmente direcionado para a plataforma Microsoft Windows, agora desenvolve aplicações nativas para Linux com o Kylix, e para o Microsoft .NET framework em suas versões mais recentes. O nome Delphi é inspirado na cidade de Delfos, o único local na Grécia antiga em que era possível consultar o Oráculo de Delfos. Os desenvolvedores do compilador buscavam uma ferramenta capaz de acessar um banco de dados Oracle. Daí veio o trocadilho "a única maneira de acessar o oráculo é usando Delphi".

Sendo o mais conhecido dos programas do tipo RAD (Rapid Application

development) o Delphi não pode ser usado para desenvolvimento de software de base ou aplicativos de sistema. Entre os engenheiros de software o Delphi é muitas vezes caracterizado com um "aplicativo programável". O Delphi é largamente utilizado no desenvolvimento de aplicações desktop, aplicações multicamadas e cliente/servidor, compatível com os banco de dados mais conhecidos no mercado. Como uma ferramenta de desenvolvimento genérica, o Delphi pode ser utilizado para diversos tipos de desenvolvimento de projeto, abrangendo desde Serviços a Aplicações Web e CTI.

5.2 FIREBIRD

Firebird (algumas vezes chamado de FirebirdSQL) é um sistema gerenciador de bases de dados. Corre em Linux, Windows, Mac OS e uma variedade de plataformas Unix. A Fundação FirebirdSQL coordena a manutenção e desenvolvimento do Firebird, sendo que os códigos fonte são disponibilizados sob o CVS da sourceforge.net .

Baseado no código do InterBase da Borland, quando da abertura de seu código na versão 6.0 (em 25 de Julho de 2000), alguns programadores em associação, assumiram o projecto de identificar e corrigir inúmeros bugs da versão original, surgindo aí o Firebird, que se tornou um banco com características próprias, obtendo uma aceitação imediata no círculo de programadores, a primeira versão 1.0. Quase que totalmente ainda compatível com sua origem, estando atualmente em sua versão 1.5, com muitas novidades. A versão 2.0 em fase de produção ainda, deverá trazer muitas inovações, já está se falando até em uma versão 3.0 que hoje tem o codinome Vulcan, cujas características já então seria de um super banco de dados, seu maior diferencial ainda se baseia na gratuidade, o banco é free em todos os sentidos: não há limitações de uso, e seu suporte amplamente discutido em listas na internet, o que facilita enormemente a obtenção de ajuda técnica. O produto se mostrou bastante seguro, suportando sistemas com dezenas de usuários simultâneos e bases de dados acima de 2GB de tamanho.

5.3 SCANNER

Scanner (ou digitalizador) é um periférico de entrada responsável por digitalizar imagens, fotos e textos impressos para o computador, um processo inverso ao da

impressora. Ele faz varreduras na imagem física gerando impulsos elétricos através de um captador de reflexos. É dividido basicamente em duas categorias:

- Scanner de mão: parecido com um rato bem grande, no qual deve-se passar por cima do desenho ou texto a ser transferido para o computador. Este tipo não é mais apropriado para trabalhos semi-profissionais devido à facilidade para o aparecimento de ruídos na transferência;
- Scanner de mesa: parecido com uma fotocopadora, no qual deve-se colocar o papel e abaixar a tampa para que o desenho ou texto seja então transferido para o computador;
- Oversized: para todos os tamanhos de documentos;
- Scanners para cheques: lê o número MICR no cheque, agilizando o processamento;
- Microfilme: para digitalizar documentos baseados em microfilmes;

Abaixo temos algumas características de scanners que devem ser verificadas:

- Sheetfed ou flatbed: para uma saída de grande volume, um scanner sheetfed (alimentação por folha) com um alimentador automático é necessário. Documentos frágeis, ou aqueles com formato pequeno podem requerer um scanner flatbed (de berço). Scanners especiais para livros e documentos frágeis, como os de grandes formatos (desenhos de engenharia etc) também são disponíveis.
- Resolução: 100 a 1600 DPIs. Resolução muito alta significa mais armazenamento e mais largura de banda, mesmo com compressão. 300 DPIs é recomendado para resultados sólidos de reconhecimento (Reconhecimento óptico de caracteres - OCR).
- Cor, preto&branco ou escala de cinza: escala de cinza é com freqüência o compromisso; é bom para OCR e permite que gráficos sejam lidos. Cor é uma opção, mas a melhoria na claridade pode ser compensada pela diminuição na velocidade de saída e tamanho do arquivo.
- Duplex ou simplex: se você tem muitos formulário frente e verso, considere a compra de um scanner duplex que digitaliza os dois lados de uma só vez.

5.4 DELPHI TWAIN

O Delphi Twain é um componente que possibilita o acesso a funcionalidades Twain (Twain é protocolo padrão de acesso a dispositivos que capturam imagens) através do Delphi de forma “transparente” para o desenvolvedor.

Como ele, através de um programa Delphi, é possível utilizar a maioria dos Scanners, webcams, fax modems, câmeras digitais para capturar imagens.

Nós escolhemos este componente, pois ele é OpenSource e proporcionou um maior entendimento do funcionamento do Padrão TWAIN.

5.5 WINISIS/MICROISIS

Segundo o informativo do IBICT sobre Características do micro CDS/ISIS o "CDS/ISIS é um sistema genérico de armazenamento e recuperação de informações, desenvolvido pela UNESCO, especialmente projetado para o gerenciamento computadorizado de bases de dados não numéricas, isto é, bases de dados cujo principal conteúdo é texto."

O CDS/ISIS é uma tecnologia de tratamento genérico de informações, desenvolvida e mantida pela Divisão de Informação e Informática da UNESCO, responsável pela sua manutenção e aprimoramento. Criado sob a tipologia da norma ISO 2709, dentro do OIT - Organização Internacional do Trabalho.

Ele permite:

- a construção;
- o armazenamento;
- a recuperação de informação e gerenciamento de Base de Dados estruturadas não-numéricas com maior direcionamento em texto.

O texto processado pelo CDS/ISIS é estruturado em elementos de dados que você define. Você os consulta por palavras, por termos prefixados, sufixados, por campos abrangendo uma consulta booleana de forma bem rápida e direta. Assim, o Microisis armazena grandes ou pequenos volumes de informação e o recupera muito mais rápido

que qualquer programa de banco de dados em rede, dentro do arquivo ou mesmo via Internet. Cada unidade de informação armazenada numa base de dados consiste em elementos de dados distintos, cada um contendo uma característica particular da entidade sendo descrita. Pôr exemplo, uma Base de Dados bibliográfica conterá informações sobre livros, reportagens, artigos de jornais, etc. Cada unidade, neste caso, conterá elementos de dados como autor, título, data de publicação, podendo ainda inserir imagens, etc.

Hoje encontramos o CDS/ISIS nas versões: VAX, DOS, UNIX, WINDOWS e INTERNET e soluções mistas desenvolvida pela Bireme, tais quais o ISIS_DLL que permite desenvolver soluções particulares, o CISIS (rotinas e programas em C++) que permitem o desenvolvimento de soluções usando as bases de dados CDS/ISIS em diversas plataformas.

5.5.1 Surgimento do ISIS

Inserido nas atividades do Programa Geral de Informação, a UNESCO iniciou em 1980 um plano de apoio à informatização de Bibliotecas com incidência especial para os países em vias de desenvolvimento que careciam de recursos econômicos para aquisição de um software para gestão de Bibliotecas, existentes apenas nos países mais desenvolvidos.

Nesta época a UNESCO possuía um sistema desenvolvido para computadores de grande porte, cujas capacidades se adequavam perfeitamente aos planos estabelecidos. Seria apenas necessário adaptá-lo de forma a que funcionasse em microcomputadores que tivessem ao alcance das Bibliotecas.

O primeiro antecedente deste programa, nasceu no final de 1960, com o desenvolvimento na Organização Internacional do Trabalho (OIT) de um conjunto de programas designado por ISIS (Integrated Set of Information System, desenhado especialmente para computadores Mainframe. Com o referido programa era possível fazer a gestão de todo o arquivo da OIT, assim como os extensos fundos documentais existentes na mesma. A OIT cedeu este pacote a todas as instituições que o solicitaram e que estavam relacionadas com as suas atividades.

Paralelamente a UNESCO desenvolveu outro sistema de gestão documental que designou por CDS (Computerized Documentation System. Naturalmente que a inexistência de normalização nas características dos equipamentos da época faziam com

que ambos os sistemas fossem incompatíveis, e assim utilizados por ambas as organizações de uma forma independente, até que em 1975 a UNESCO decidiu renovar os seus equipamentos, substituindo o equipamento da ICL por um equipamento da IBM.

Com esta mudança surgiram grandes dificuldades para instalar o antigo programa CDS nos novos equipamentos da IBM, o que levou a UNESCO a solicitar à OIT a ceder seu programas ISIS, os quais tiveram que ser modificados para que pudessem funcionar no novo sistema operativo dos modernos Mainframes IBM, NVS (OS1) e pudessem ler a informação armazenada nos antigos equipamentos ICL.

Da referida adaptação resultou na fusão entre dois sistemas, o CDS da UNESCO e o ISIS da OIT, dando origem a um novo produto que se designou CDS/ISIS, desenhado, conforme foi anteriormente referido, para os Mainframes com o sistema operativo MVS. Um ano antes, em 1974, a OIT cedeu a versão para DOS do seu programa ISIS à International Development Research Center do Canadá, que entre 1976 e 1977 desenvolveu uma nova versão do ISIS, designada MINI-ISIS. Esta versão foi especialmente criada para correr em microcomputadores Hewlett-Packard da série 3000, que foi entregue à OIT. A nova versão do MINI-ISIS foi, a partir de então, adaptada pela OIT em substituição do antigo ISIS. A partir desse momento, a UNESCO e a OIT acordaram em ceder os seus programas às instituições relacionadas com os objetivos de cada uma.

No final de 1977 a OIT abandonou a política de distribuição por falta de recursos, e solicitou à UNESCO que a assumisse e desse continuidade ao ISIS e MINI-ISIS. A OIT distribuiu cópias dos seus programa a cerca de 50 instituições em todo o mundo.

No início dos anos oitenta, alguns dos utilizadores da versão Mainframe do CDS/ISIS da UNESCO, exigiram que esta desenvolvesse uma versão que fosse capaz de correr em microcomputadores que começavam a surgir no mercado. Entre 1982 e 1983 fez-se o desenvolvimentos de um versão do CDS/ISIS numa máquina PDP11 da DEC, que foi posteriormente adaptada para microcomputadores da IBM quando se anunciou a saída no mercado das máquinas IBM-PC.

A primeira versão do MICROISIS (versão do CDS/ISIS para computadores pessoais) correu pela primeira vez num equipamento IBM PC-XT de 150 kb de memória e com 10 Mb de disco, tendo sido apresentada num reunião de utilizadores da versão para

Mainframe, realizada em Buenos Aires em 1985.

Designado oficialmente por "CDS/ISIS Mini-micro version" é também conhecido por "CDS/ISIS" ou simplesmente ISIS. Na América Latina, onde a versão para minicomputadores MINISIS prevalece (versão desenvolvida pela IDRC como atrás referido) é conhecido por MICROISIS.

A partir de então, iniciou-se a distribuição do programa tal como o conhecemos hoje. Apenas no primeiro ano a UNESCO distribuiu 1000 cópias do novo software tendo tido uma excelente aceitação nas Bibliotecas e instituições de todo o mundo. Tal foi o sucesso que a UNESCO logo se enfrentou com a impossibilidade de suportar os custos da distribuição, que incluíam os custos de edição, tempo e pessoal dedicado as ditas funções.

Esta situação levou a UNESCO a recorrer a uma política que se tinha criada com a distribuição das sucessivas versões para Mainframe, decidindo ceder a sua difusão a distribuidores oficiais, que são geralmente, instituições relacionadas com os mesmo fins e objetivos do Programa Geral de Informação, que com caráter voluntário se encarregaram da distribuição.

5.5.2 Vantagens

Uma das maiores vantagens oferecidas pelo projeto genérico do sistema é que o CDS/ISIS é capaz de manipular um número ilimitado de bases de dados, as quais podem conter elementos de dados completamente diferentes.

Em particular, o CDS/ISIS permite:

- definir bases de dados contendo os elementos de dados requeridos;
- incluir novos registros em uma base de dados;
- modificar, corrigir ou retirar registros existentes;
- construir e manter, automaticamente, estruturas que permitam o acesso rápido às bases de dados;
- recuperar registros na sequência desejada, através de uma sofisticada linguagem de busca;
- mostrar, em tela, registros (inteiros ou parte) de acordo com um formato desejado;

- gerar produtos impressos tais como índices e catálogos, de boa qualidade técnica;
- desenvolver aplicações especializadas, usando as facilidades de programação do CDS/ISIS. Através dessas facilidades, o usuário pode adaptar o sistema às necessidades de cada instalação;
- intercambiar (exportar/importar) dados entre as bases de dados do CDS/ISIS e/ou entre outros sistemas por meio da norma ISO 2709.

Outro ponto muito forte do Winisis é a forma como ele acessa as informações nele guardadas, o Winisis guarda a informação em índices baseando-se no tamanho das palavras, esses índices utilizam uma estrutura de dados muito conhecida chamada árvore balanceada a vantagem de se usar uma árvore é a velocidade com que se acha a informação. Imagine que você tenha no Winisis uma base com 32768 registros, quando a busca por uma palavra fosse iniciada aconteceriam no máximo 15 interações na árvore, (isso ocorre por que a ordem de grandeza dos algoritmos de busca em árvores é representado pelo logaritmo de n na base 2) imaginado que cada interação fosse realizada a 0,00001 segundo (o que muito mais lento do que a maioria dos computadores de hoje pode fazer) teríamos 15 interações de 0,00001 segundos, ou seja, nossa busca duraria cerca de 0,0015 segundos, por sua vez, se você estiver usando um banco de dados relacional qualquer, como Mysql, HSQLDB ou Oracle caso não se tenha tomado uma quantidade considerável de cuidados, como a adoção de índices, esse banco executaria uma query sql do tipo: (select * from livros where titulo equals "%palavra%") no pior caso teríamos uma busca seqüencial no título de todos os livros cadastrados nas bases, voltando então as contas seriam 32768 ciclos de 0,0001 segundos, ou seja, essa busca demoraria 3,2768 segundos, isso só para a busca nos títulos dos livros e desconsiderado o uso de expressões booleanas, faça uma teste, encontre um Profissional de informática com bastante experiência em banco de dados, um DBA, pergunte a ele o que ele acha dessa query (select * from livros where titulo equals "%palavra%") para buscar uma palavra num banco de dados, provavelmente ele vai dizer algo do tipo: Isso é loucura.

Além de que, no Winisis somente se ocupa espaço em disco para os dados que efetivamente compreendem o registro. Já que ele trata cada registro à parte e inclui apenas os campos (tags) que possuírem dados.

5.5.3 Características do CDS/ISIS

Embora o CDS/ISIS lide com texto e com palavras, e apresente muitas das características normalmente encontradas em editores de texto, ele faz mais do que apenas processar texto. Isto se deve ao fato de que o texto que o CDS/ISIS processa é estruturado em elementos de dados definidos pelo usuário.

Em termos mais gerais, pode-se pensar em uma base de dados CDS/ISIS como um arquivo de dados relacionado, coletados para satisfazer as necessidades de informação de determinada comunidade de usuários. Pode ser um simples arquivo de endereços, ou um arquivo mais complexo, como um catálogo bibliográfico ou um catálogo de projetos de pesquisa. Cada unidade de informação armazenada em uma base de dados consiste de elementos de dados distintos, cada um contendo uma característica particular da entidade que estiver sendo descrita. Por exemplo: uma base de dados bibliográfica pode conter informações sobre livros, relatórios, artigos de periódicos, etc. Neste caso, cada unidade consistirá de elementos de dados tais como autor, título, imprensa, etc.

Os elementos de dados são armazenados em campos, a cada um dos quais é atribuído uma etiqueta numérica indicativa de seu conteúdo. Pode-se pensar na etiqueta como sendo o nome do campo na forma como é conhecida pelo CDS/ISIS.

A coleção de campos contendo todos os elementos de dados de determinada unidade de informação é chamada registro.

Característica singular do CDS/ISIS é que ele é especificamente definido com a capacidade de trabalhar com campos (e conseqüentemente registros) de tamanho variável, permitindo, por um lado, uma completa liberdade na definição do tamanho máximo de cada campo. Um campo pode estar ausente em um ou mais registros, pode conter um único elemento de dados, ou dois ou mais elementos de dados de tamanho variável. Neste caso, diz-se que o campo contém subcampos, cada um identificado por um delimitador de subcampo precedendo o elemento de dados correspondente. Além disso, um campo pode ser repetitivo, isto é, qualquer registro pode conter mais de um fato ou ocorrência do campo.

O arquivo mestre contém todos os registros de determinada base de dados, resulta que cada um deles formado de um conjunto de campos definidos. Cada registro criado é

integrado ao arquivo mestre com um número específico, chamado de Master File Number - MFN (número de registro principal) ".

5.5.4 Definições complementares

- Informação é o conjunto de fatos ou dados organizados de uma forma lógica. É necessariamente um dado significativo. A informação são dados arranjados em forma útil. A informação não tem valor se não pode ser localizada ou preparada em tempo.
- Dado é a representação formal de um fato, idéia, item, etc., representado por letras, números e caracteres especiais.
- Banco de dados é uma coleção de dados inter-relacionados, armazenados com redundância controlada, para servir a múltiplas aplicações: os dados armazenados de forma a serem independente dos programas que usam; uma abordagem comum e controlada e utilizadas para adicionar novos dados, bem como modificar e buscar dados existentes na base de dados.

Um sistema de informações bem planejado deve proporcionar informações necessárias a cada setor de uso. Fornecer informações relevantes em uma forma útil a pessoa certa, no tempo certo para uso em tomada de decisão. Um sistema de informação é bem projetado somente se fornece informações certas em tipo, qualidade e no tempo devido, e faz isso de forma econômica.

5.5.5 Dados técnicos

- Versão Atual : WinIsis 1.5.3
- As seguintes restrições do sistema encontram-se atualmente em vigor:
- Número máximo de bases de dados: ilimitado;
- Número máximo de registros na base de dados, dentro do limite de 500 Mb: 16 milhões;
- Tamanho máximo do registro (4 páginas A-4): 8000 caracteres;
- Máximo de campos (definíveis na FDT): 200;

- N. máx. de linhas da FST: 200;
- Tamanho máximo do campo: 1650 caracteres;
- Máximo de campos em uma planilha: 19;
- Máximo de páginas em uma planilha: 20;
- Tamanho máximo de um formato de exibição (2 páginas A-4): 4000 caracteres;
- Máximo de palavras proibidas (arquivo stopword): 799;

5.6 ISIS DLL

A ISIS_DLL - CDS/ISIS Dynamic Link Library - foi desenvolvida pela BIREME e UNESCO como o componente da versão Windows para a programação de aplicações específicas. Ela cumpre, em princípio, o mesmo papel que o módulo ISIS Pascal na versão para DOS.

A ISIS_DLL possui, entre outras, as características:

- É um componente independente do sistema Microisis para Windows e, mesmo assim, é parte integrante integrante da família ISIS e suas funções operam com compatibilidade total com Microisis;
- Pode-se desenvolver aplicações em qualquer linguagem que tenha compilador no sistema operacional Windows capaz de chamar DLLs (C, C++, Java, Visual Basic e Delphi);

6 FUNDAMENTAÇÃO TEÓRICA

O Sistema DIGIDOC utiliza técnicas de sistemas de Gerenciamento de Documentos Eletrônicos. Abaixo estamos incluindo algumas informações que utilizamos para a implementação do sistema.

6.1 GERENCIAMENTO ELETRÔNICO DE DOCUMENTOS

Gerenciamento eletrônico de documentos (GED) é uma tecnologia que provê um meio de facilmente gerar, controlar, armazenar, compartilhar e recuperar informações existentes em documentos. Os sistemas GED permitem aos usuários acessar os documentos de forma ágil e segura, a capacidade de gerenciar documentos é uma ferramenta indispensável para a Gestão do Conhecimento.

GED revoluciona o arquivamento de informação e provém meios de rapidamente recuperar e compartilhar todos documentos em seu sistema. Todos os sistemas GED devem possuir os seguintes componentes básicos:

- Ferramentas de escanear para trazer os documentos para o sistema;
- Métodos de arquivamento e armazenamento de documentos;
- Ferramentas de recuperação para encontrar documentos;
- Controle de acesso para prover documentos para pessoas autorizadas;

Documentos formam a grande massa de conhecimentos de uma empresa. O GED permite preservar esse patrimônio e organizar eletronicamente a documentação, para assegurar a informação necessária, na hora exata, para a pessoa certa. O GED lida com qualquer tipo de documentação.

Qualquer tipo de empresa, pequena, média ou grande, pode usar o GED, entre: escolas; empresas de advocacia; hospitais; administradoras de condomínios; empresas de recrutamento; escritórios de arquitetura, design e engenharia; assessorias de imprensa e de comunicação; e consultorias. Nas médias e grandes empresas, o GED poderá ser aplicado para setores específicos (RH, Treinamento, Contabilidade, Marketing, Informática).

6.2 DADOS INTERESSANTES

A humanidade gerou a mesma quantidade de informação nos últimos 50 anos que nos 5 mil anteriores. Esse número duplicará nos próximos 26 meses. Em 2010, a informação duplicará a cada 11 horas.

Cada vez mais estamos gerando mais documentos em papel. Segundo a AIIM International – Association for Information and Image Management International, EUA, a maior associação do mundo sobre gerenciamento da documentação, 95% das informações dos Estados Unidos estavam em papel em 1990. Este ano, cerca de 92% das informações ainda estarão em papel.

Essa avalanche de papel gera a cada dia maiores problemas:

- Um executivo gasta em média quatro semanas por ano procurando documentos.
- Faz-se, em média, 19 cópias de cada documento.
Gasta-se US\$ 250,00 para recriar cada documento perdido.
- A imagem de um documento digitalizada a 200 dpi (pontos por polegada) e comprimida a 10:1 requer 50KB de armazenamento. Um gigabyte acomoda 20 mil imagens.
- Quinhentas páginas de texto requerem 1 MB de armazenamento.
- Um arquivo de quatro gavetas, com 2.500 folhas de papel por gaveta, comporta, em média 10 mil imagens de documento.
- Um CD-R mede 120mm de diâmetro e pode armazenar até 650 MB de informação. Isso corresponde a 13 mil páginas de documentos.
- Estudos revelam que os escritórios criam cerca de 1 bilhão de páginas de papel por dia. Segundo uma pesquisa do IDC, EUA, esse total é constituído de 600 milhões de páginas de relatórios de computador, 234 milhões de fotocópias e 24 milhões de documentos diversos. Isso somente nos Estados Unidos.

6.3 DIGITALIZAÇÃO

É a transformação de documentos em papel numa imagem eletrônica. Também pode significar digitalizar microfilmes. Imagens eletrônicas também podem ser captadas,

além do scanner, através de:

- Fax: O software pode ler a partir do servidor de fax. Esteja atento, pois a qualidade da imagem será baixa, o que poderá afetar negativamente a qualidade do reconhecimento.
- Câmera de telefone: Câmeras de alta resolução em telefones celulares e software projetado para trabalhar num celular permitem a captação e conversão de documentos.
- Dispositivo multifuncional: MFDs conectados em rede pode bastar para necessidade de baixo volume.

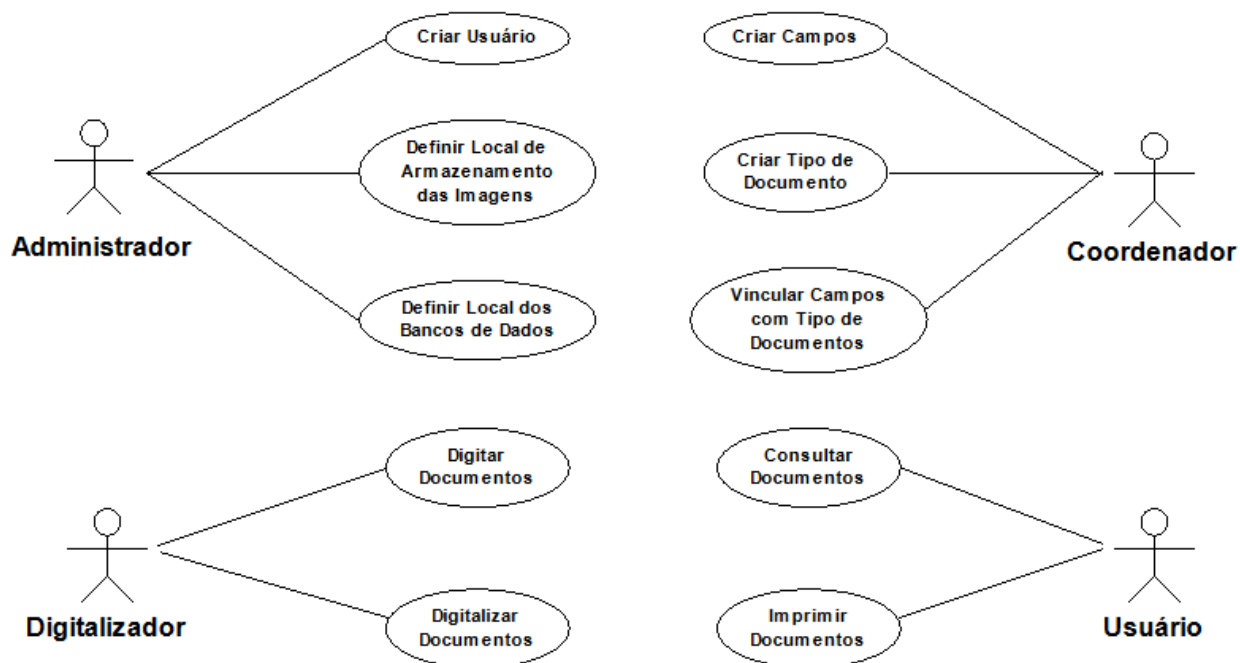
As imagens podem ser salvas nos seguintes formatos:

- TIFF: Tagged Image File Format – Geralmente usado para documentos oficiais. Pode ser comprimido. Padrão De Facto. Existem múltiplos tipos.
- JPEG: Frequentemente usado para documentos coloridos. Também é um método de compressão. Padrão ISO. Existem múltiplos tipos.
- PDF: Padrão De Facto. Uma réplica do documento. Com texto OCR pode ser indexado por full text ou pesquisado.
- GIF: Graphics Interchange Format – Muda e disponibiliza gráficos com resolução de alta qualidade.

8 MODELAGEM

8.1 DIAGRAMA DE CASOS DE USO

FIGURA 3 – DIAGRAMA DE CASOS DE USO



8.1.1 Criar Usuário

Este caso de uso tem como objetivo o cadastramento de usuários que terão acesso disponível ao sistema, além da definição do perfil a ser desempenhado por cada usuário. Este caso de uso fica a cargo do ator denominado Administrador.

8.1.2 Definir local de armazenamento das imagens

Este caso de uso tem como objetivo a definição da configuração do local de armazenamento das imagens dos documentos digitalizados. Este caso de uso fica a cargo do ator denominado Administrador.

8.1.3 Definir local do banco de dados

Este caso de uso tem como objetivo a definição dos locais onde estão

armazenados os bancos de dados do sistema. Este caso de uso fica a cargo do ator denominado Administrador.

8.1.4 Criar campos

Este caso de uso tem como objetivo a criação de campos que serão utilizados para vinculação a documentos. Este caso de uso fica a cargo do ator denominado Coordenador.

8.1.5 Criar Tipo de Documento

Este caso de uso tem como objetivo a criação de documentos que serão digitalizados no sistema. Este caso de uso fica a cargo do ator denominado Coordenador.

8.1.6 Vincular Campos com Tipo de Documento

Este caso de uso tem como objetivo a vinculação dos campos com os tipos de documentos cadastrados. Este caso de uso fica a cargo do ator denominado Coordenador.

8.1.7 Digitar Documentos

Este caso de uso tem como objetivo a inclusão de dados dos documentos que serão digitalizados. Este caso de uso fica a cargo do ator denominado Digitalizador.

8.1.8 Digitalizar Documentos

Este caso de uso tem como objetivo a digitalização das imagens dos documentos cadastrados. Este caso de uso fica a cargo do ator denominado Digitalizador.

8.1.9 Consultar documentos

Este caso de uso tem como objetivo a consulta de documentos que já estão digitados no sistema. Este caso de uso fica a cargo do ator denominado Usuário.

8.2 CENÁRIOS

8.2.1 Cenário Criar Usuários

Na tela principal, o usuário clica no menu superior em Administração e depois em Usuários. Nesta tela são apresentados todos os usuários já cadastrados.

Cadastro de novos usuários

- Cenário Principal

Clicando no botão Novo, habilitam-se os campos para inserção das informações, e somente os botões Salvar e Cancelar ficam ativos, após a inserção das informações, o usuário clica em Salvar, os campos são desabilitados, e o usuário que acabou de ser cadastrado passa a constar na listagem de usuários cadastrados da tela.

- Cenário Secundário

O usuário tenta inserir informações nos campos antes de clicar no botão Novo, porém não consegue, pois os campos continuam desabilitados. O usuário clica no botão Novo e os campos são habilitados para inserção de informações, porém o usuário que está efetuando o cadastro deixa campos obrigatórios sem preenchimento e clica no botão Salvar, nesse momento é apresentada a mensagem “Favor preencher os campos obrigatórios!”.

Edição de usuários cadastrados

- Cenário Principal

O usuário do sistema seleciona um dos usuários cadastrados no sistema e clica no botão Alterar, assim ficam habilitados somente os botões Salvar e Cancelar, as informações do usuário selecionado são apresentadas nos campos, que estão habilitados para alteração das informações. Após a alteração, o usuário clica em Salvar.

- Cenário Secundário

O usuário tenta alterar as informações dos campos sem antes clicar no botão Alterar, porém não consegue, pois os campos não estão habilitados para edição. Após clicar em Alterar, segue-se o fluxo do cenário principal.

Exclusão de usuários cadastrados

- Cenário Principal

O usuário do sistema seleciona um dos usuários cadastrados no sistema e clica no botão Excluir, então é apresentada uma mensagem de confirmação de exclusão, caso o usuário confirme, o registro é excluído fisicamente, deixando de ocupar espaço em disco.

- Cenário Secundário

O usuário tenta excluir um usuário que sem selecioná-lo, porém não consegue, o sistema apresenta a mensagem “Não há nenhum registro selecionado para exclusão.”, e então o sistema volta para o estado inicial.

8.2.2 Cenário Definir local de armazenamento das imagens

Na tela principal, o usuário clica no menu superior em Administração e depois em Configurações.

Configurações

- Cenário Principal

Nesta tela há a possibilidade de se selecionar o caminhos do diretório onde ficarão as imagens digitalizadas. Para isso, o usuário posicionará o cursor num dos edits de path, e após alteração do caminho, clicará no botão salvar.

8.2.3 Cenário Definir local do banco de dados

Externo ao sistema, o usuário entra no módulo de Configuração do Banco de Dados.

Configurações

- Cenário Principal

Nesta tela há a possibilidade de se selecionar o diretório onde ficará o banco de dados WinIsis e o arquivo do banco de dados FireBird. Para isso, o usuário posicionará o cursor num dos edits de path, e após alteração do caminho, clicará no botão salvar.

Cenário Criar Campos

Na tela principal, o usuário clica no menu superior em Documentos > Cadastrar Campos. Nesta tela são apresentados todos os campos já cadastrados.

Cadastrando Novos Campos

- Cenário Principal

Clicando no botão Novo, habilitam-se os edits para inserção das informações, e somente os botões Salvar e Cancelar ficam ativos, após a inserção das informações, o usuário clica em Salvar, os edits são desabilitados, e o campo que acabou de ser cadastrado passa a constar na listagem de campos cadastrados da tela.

- Cenário Secundário

O usuário tenta inserir informações nos edits antes de clicar no botão Novo, porém não consegue, pois os edits continuam desabilitados.

O usuário clica no botão Novo e os edits são habilitados para inserção de informações, porém o usuário que está efetuando o cadastro deixa edits obrigatórios sem preenchimento e clica no botão Salvar, nesse momento é apresentada a mensagem “É obrigatório o preenchimento de todos os campos.”.

Edição de Campos cadastrados

- Cenário Principal

O usuário do sistema seleciona um dos campos cadastrados no sistema e clica no botão Alterar, assim ficam habilitados somente os botões Salvar e Cancelar, as informações do campo selecionado são apresentadas nos edits, que estão habilitados para alteração das informações. Após a alteração, o usuário clica em Salvar.

- Cenário Secundário

O usuário tenta alterar as informações dos edits sem antes clicar no botão Alterar, porém não consegue, pois os edits não estão habilitados para edição. Após clicar em Alterar, segue-se o fluxo do cenário principal.

Exclusão de Campos cadastrados

- Cenário Principal

O usuário do sistema seleciona um dos campos cadastrados e clica no botão Excluir. É apresentada uma mensagem de confirmação de exclusão, caso o usuário

confirme, o registro é excluído, caso o usuário não confirme a exclusão, o sistema volta ao status inicial.

- Cenário Secundário

O usuário tenta excluir um campo que já possui dados cadastrados ou vínculo com algum Tipo de Documento, porém não consegue, o sistema apresenta a mensagem “Campo possui dados.” ou “Campo possui vínculo com algum Tipo de Documento” e então o sistema volta para o estado inicial.

8.2.4 Cenário Criar Tipo de Documento

Na tela principal, o usuário acessa no menu a opção Documentos > Cadastrar Tipo de Documento, ou no botão de atalho Cadastrar Tipo Documento.

Cadastrando novo Tipo de Documento

- Cenário Principal

Na tela que se abrirá, o usuário clica no botão Novo, então ficam habilitados somente os botões Salvar e Cancelar, e os edits ficam habilitados para inserção de informações, após a inserção, o usuário clica em Salvar, o registro é gravado e os edits são novamente desabilitados, juntamente com os botões Novo, Excluir e Alterar. Neste momento também são novamente desabilitados os botões Salvar e Cancelar.

- Cenário Secundário

O usuário tenta inserir informações nos edits sem antes clicar no botão Novo, porém não consegue, pois os edits estão desabilitados, após clicar no botão Novo, os edits se habilitam para receberem informações e segue-se o fluxo do cenário principal.

Edição de Tipo de Documento cadastrado

- Cenário Principal

O usuário seleciona na lista de tipos de documentos já cadastrados, qual o registro que deseja alterar, após isso, clica no botão Alterar, e os edits se habilitam para edição das informações, juntamente com os botões Salvar e Cancelar, desabilitando-se os botões Novo, Alterar e Excluir. Após a edição, o usuário clica em salvar, gravando as

informações editadas. Nesse momento os edits são novamente desabilitados, juntamente com os botões Salvar e Cancelar, habilitando-se os botões Novo, Alterar e Excluir.

- Cenário Secundário

O usuário clica no botão Editar sem selecionar nenhum Tipo de Documento na listagem de registros da tela, porém sem sucesso. Após selecionar-se um Tipo de Documento na listagem da tela, pode-se clicar em Editar, e então se segue o fluxo normal do Cenário Principal.

Exclusão de Tipo de Documento cadastrado

- Cenário Principal

O usuário seleciona na lista de tipos de documentos já cadastrados, qual o registro que deseja excluir, após isso, clica no botão Excluir. O sistema apresenta uma mensagem de confirmação de exclusão. Caso o usuário confirme a exclusão, clicando em “Sim”, o registro é excluído, caso o usuário não confirme a exclusão, clicando em “Não”, o registro é preservado e o sistema volta à situação inicial.

- Cenário Secundário

O usuário clica no botão Excluir sem selecionar nenhum Tipo de Documento na listagem de registros da tela, porém sem sucesso. Após selecionar-se um Tipo de Documento na listagem da tela, pode-se clicar em Excluir, e então se segue o fluxo normal do Cenário Principal.

O usuário seleciona um Tipo de Documento cadastrado, clica em Excluir, porém, se o Tipo de Documento campos vinculados, o sistema não permite a exclusão do Tipo de Documento, apresentando a mensagem: “Existem campos vinculados para este Tipo de Documento”.

8.2.5 Cenário Vincular Campos com Tipo de Documento cadastrado

Na tela principal, o usuário acessa o menu Documentos > Cadastrar Tipo de Documento. Serão apresentados os Tipos de Documentos já cadastrados no sistema, e clicando-se duas vezes em um dos registros, pode-se clicar no botão Vincular Campos.

Vincular Campos

- Cenário Principal

Na tela de vínculo de campos, do lado esquerdo são listados os campos disponíveis para vínculo com o tipo de documento selecionado, basta clicar em um dos campos, selecionando o mesmo, e após isso, clica-se no botão “>”, para que este campo seja vinculado para o tipo de documento e apareça no lado direito da tela. Há também a opção de clicar no botão “>>”, que faz com que todos os campos sejam vinculados ao tipo de documento de uma só vez e apareçam do lado direito da tela. Após feito o vínculo dos campos, pode-se selecionar do lado direito da tela os campos vinculados para se atribuir os status de campo obrigatório ou campo não-obrigatório, campo repetitivo ou campo não-repetitivo. Caso o usuário deseje desvincular algum campo de um tipo de documento, basta selecioná-lo do lado direito da tela, e clicar no botão “<” para que o vínculo seja desfeito e o campo seja novamente passado para o lado esquerdo da tela. Clicando-se no botão “<<”, são desfeitos os vínculos de todos os campos com o tipo de documento.

- Cenário Secundário

O usuário tenta clicar no botão Vincular Campos, sem selecionar um dos tipos de documento previamente cadastrados. O sistema apresenta a mensagem “Não há nenhum documento selecionado para poder vincular os campos”. Após clicar duas vezes em um dos itens da listagem, segue-se o fluxo normal do cenário principal.

8.2.6 Cenário Digitar Documentos

Na tela principal, o usuário clica no menu superior em Documentos > Cadastrar Documentos.

Digitando Novo Documento

- Cenário Principal

O usuário seleciona o tipo de documento e o sistema automaticamente cria os campos vinculados com o mesmo no formulário. Clicando no botão Novo, habilitam-se os edits para inserção das informações, e somente os botões Salvar e Cancelar ficam ativos, após a inserção das informações, o usuário clica em Salvar, os edits são desabilitados. Surge uma mensagem perguntando se o usuário deseja digitalizar as imagens para o documento. Se confirmada abre a tela de imagens para o documento.

- Cenário Secundário

O usuário seleciona um tipo de documento que não possui campos vinculados. O sistema exibe então a mensagem “Não há campos vinculados para este tipo de documento.”. O usuário tenta inserir informações nos edits antes de clicar no botão Novo, porém não consegue, pois os edits continuam desabilitados.

O usuário clica no botão Novo e os edits são habilitados para inserção de informações, porém o usuário que está efetuando o cadastro deixa edits obrigatórios sem preenchimento e clica no botão Salvar, nesse momento é apresentada a mensagem “Favor preencher todos os campos obrigatórios.”.

Edição Documentos Cadastrados

- Cenário Principal

O usuário do sistema informa o número do registro e clica no botão aplicar, o sistema automaticamente cria os campos vinculados para o tipo de documento do registro no formulário e preenche-os com os dados do registro. O usuário clica no botão Alterar, assim ficam habilitados somente os botões Salvar e Cancelar e os edits do formulário com os dados do registro. Após a alteração, o usuário clica em Salvar.

- Cenário Secundário

O usuário tenta alterar as informações dos edits sem antes clicar no botão Alterar, porém não consegue, pois os edits não estão habilitados para edição. Após clicar em Alterar, segue-se o fluxo do cenário principal.

Exclusão de Documentos Cadastrados

- Cenário Principal

O usuário do sistema informa o número do registro e clica no botão aplicar, o sistema automaticamente cria os campos vinculados para o tipo de documento do registro no formulário e preenche-os com os dados do registro. O usuário do sistema clica no botão Excluir. É apresentada uma mensagem de confirmação de exclusão e informado que se o documento possuir imagens as mesmas também serão excluídas. Caso o usuário confirme, o registro é excluído e consequentemente suas imagens também. Caso

o usuário não confirme a exclusão, o sistema volta ao status inicial.

8.2.7 Cenário Digitalizar Documentos

Na tela principal, o usuário clica no menu superior em Documentos > Digitalizar. Abre a tela de imagens, o usuário informa o número do registro e clica no botão aplicar. Os dados e as imagens existentes para o registro aparecem nos campos do formulário.

Digitalizando Nova Imagem para o Documento

- Cenário Principal

O usuário clica no botão Novo e o sistema abre a tela de digitalização. Nesta tela clicando no botão Digitalizar, o sistema pede para selecionar o Scanner, no qual a imagem será digitalizada. Selecionando o Scanner, o mesmo é aquecido e a imagem é digitalizada. Os botões Salvar e Cancelar ficam ativos. O usuário clica em Salvar e um thumbnail da imagem é salvo no banco de dados junto com o caminho onde a imagem digitalizada será salva.

- Cenário Secundário

O usuário tenta inserir alguma imagem sem informar o registro, mas não consegue. Depois de informar o registro e clicar no botão Aplicar segue-se o fluxo do Cenário Principal.

Edição de Imagens Digitalizadas

- Cenário Principal

O usuário seleciona uma imagem, dando um duplo clique sobre ela, depois clica no botão Alterar e o sistema abre a tela de digitalização, carregando o thumbnail da imagem selecionada na tela. Nesta tela clicando no botão Digitalizar, o sistema pede para selecionar o Scanner, no qual a imagem será digitalizada. Selecionando o Scanner, o mesmo é aquecido e a imagem é digitalizada. Os botões Salvar e Cancelar ficam ativos. O usuário clica em Salvar e um thumbnail da imagem é salvo no lugar do antigo assim como a imagem digitalizada é salva no lugar da antiga.

- Cenário Secundário

O usuário tenta alterar alguma imagem sem seleciona alguma, mas não consegue.

Depois de selecionada a imagem, segue-se o fluxo do Cenário Principal.

Exclusão de Imagens Digitalizadas

- Cenário Principal

O usuário seleciona uma imagem, dando um duplo clique sobre ela, depois clica no botão Excluir, o sistema pede a confirmação e depois disto, exclui tanto o registro no banco de dados, quanto a imagem no diretório de.

- Cenário Secundário

O usuário tenta excluir alguma imagem sem seleciona alguma, mas não consegue. Depois de selecionada a imagem, segue-se o fluxo do Cenário Principal.

8.2.8 Cenário Consultar Documentos

Na tela principal, o usuário clica no menu superior em Documentos > Pesquisar. Abre a tela de Pesquisa.

Consultar Documento pelo número do registro

- Cenário Principal

O usuário informa o número do registro e clica no botão pesquisar. O sistema verifica o tipo de documento cria os campos e preenche cada um com os dados do devido registro. Quando o registro é selecionado com um duplo clique os botões digitalizar e editar são habilitados.

- Cenário Secundário

O usuário informa um número de registro que não existe e clica no botão Pesquisar. O sistema exibe a seguinte mensagem “Registro não encontrado.”.

Consultar Documento por termos de pesquisa

- Cenário Principal

O usuário informa o tipo de documento que deseja procurar, informa os dados que devem ser pesquisados (por exemplo um nome ou algum dado que ele quer que o

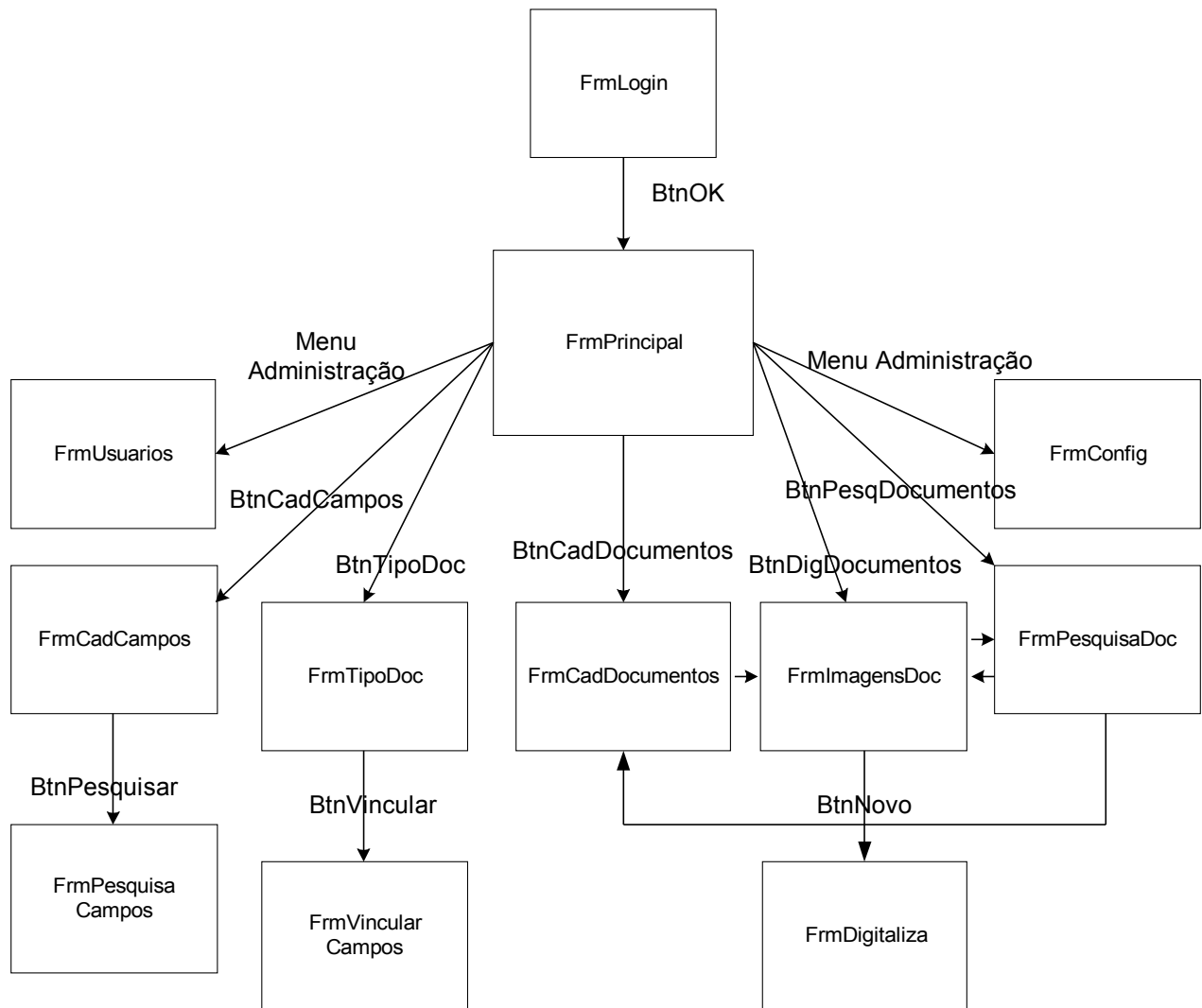
documento possua), seleciona o tipo de pesquisa, se é por palavra inteira, por termo e se todas as palavras devem conter no registro ou se o registro deve possuir qualquer umas das palavras e clica no botão pesquisar. O sistema executa a pesquisa e retorna os dados encontrados. Quando algum registro é selecionado com um duplo clique os botões digitalizar e editar são habilitados.

- Cenário Secundário

O usuário informa um dados que não possuem em nenhum registro e clica no botão Pesquisar. O sistema exibe a seguinte mensagem “Registro não encontrado.”.

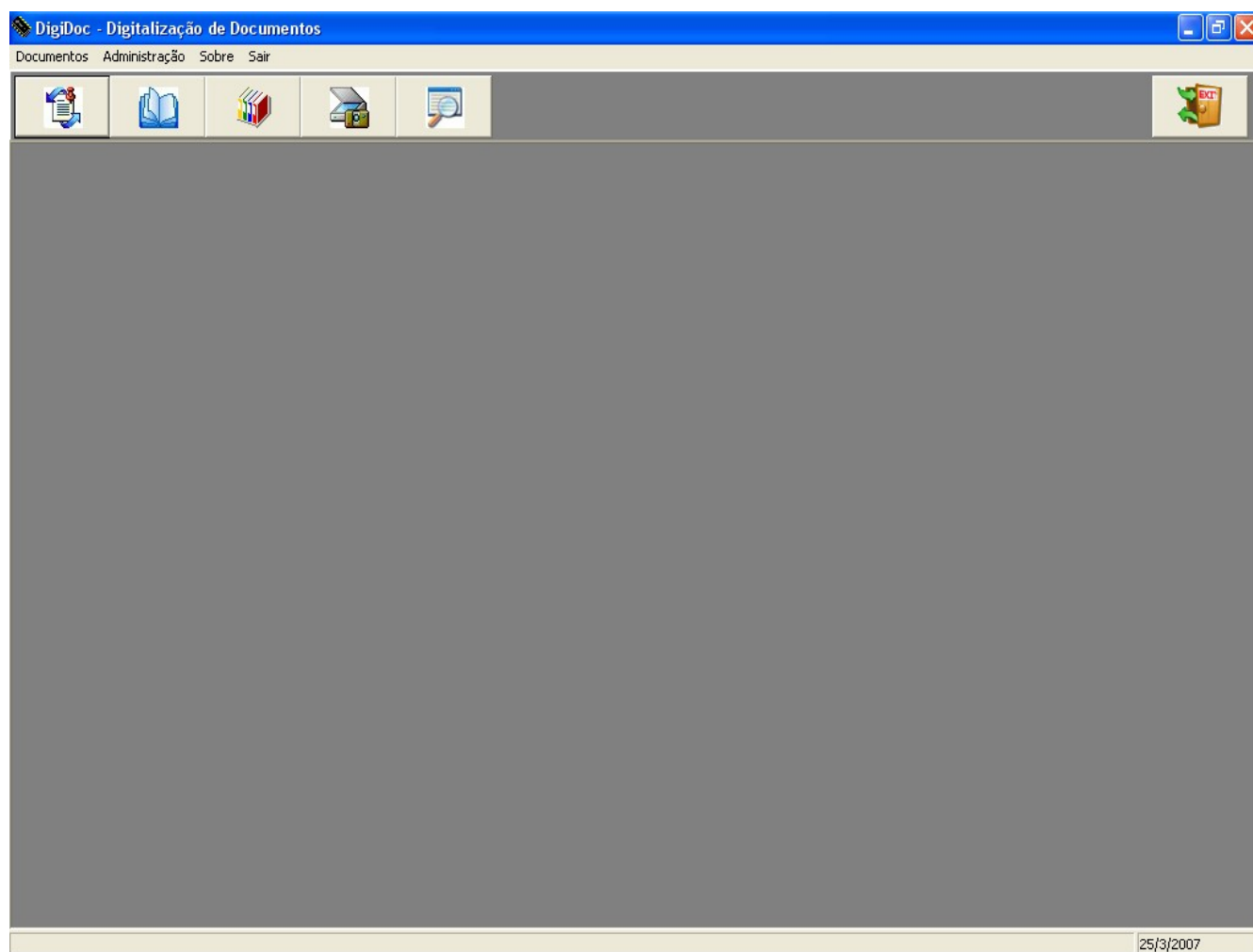
8.3 DIAGRAMA DE TELAS/SEQUÊNCIA/COLABORAÇÃO

FIGURA 4 – DIAGRAMA DE TELAS



8.3.1 TELA FrmPrincipal

FIGURA 5 – TELA PRINCIPAL



8.3.2 TELA FrmCadCampos

FIGURA 6 – TELA CADASTRO DE CAMPOS

Cadastrar Campos

Nome do Campo

Tamanho Máximo

Tipo do Campo

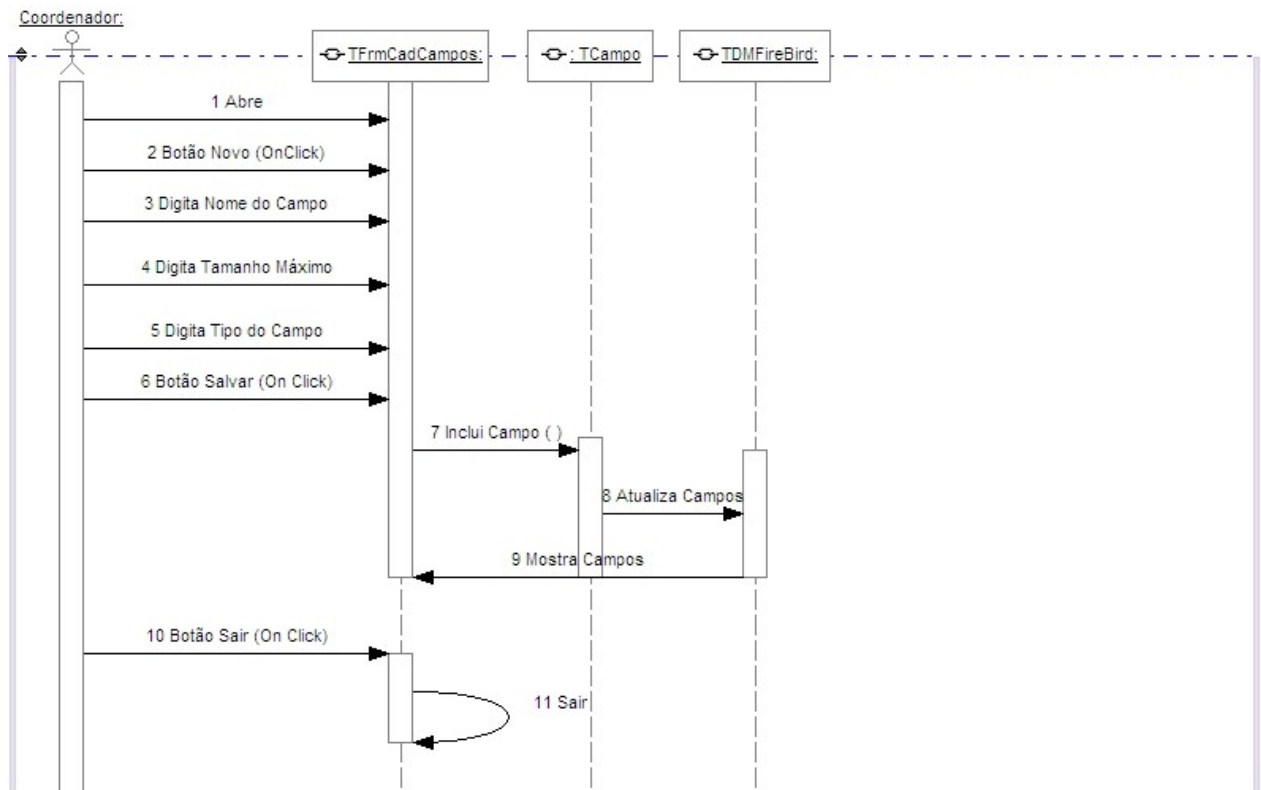
☐ Numérico

☐ Alfa Numérico

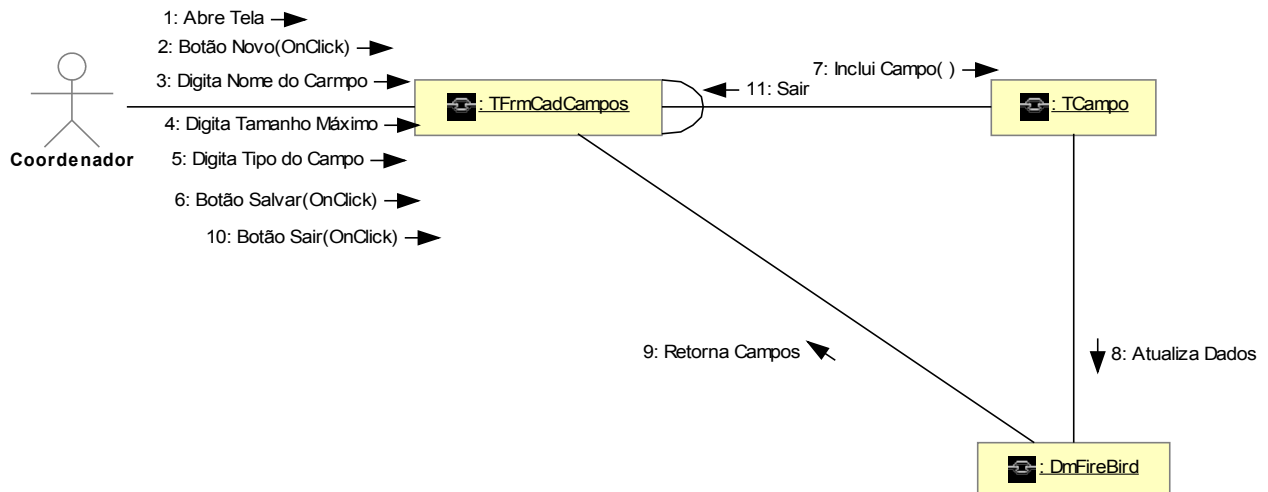
Nome	Tam.	Tipo
CAMPO 2	20	Alfanum.
CAMPO 3	13	Alfanum.
CAMPO 4	4	Numérico
CAMPO 5	20	Alfanum.
CAMPO 6	8	Alfanum.
CAMPO 7	30	Alfanum.
CAMPO 8	10	Alfanum.
CAMPO 9	20	Alfanum.
ENDERECO	30	Alfanum.

NOVO **EDITAR** **SALVAR** **EXCLUIR** **CANCELAR** **SAIR**

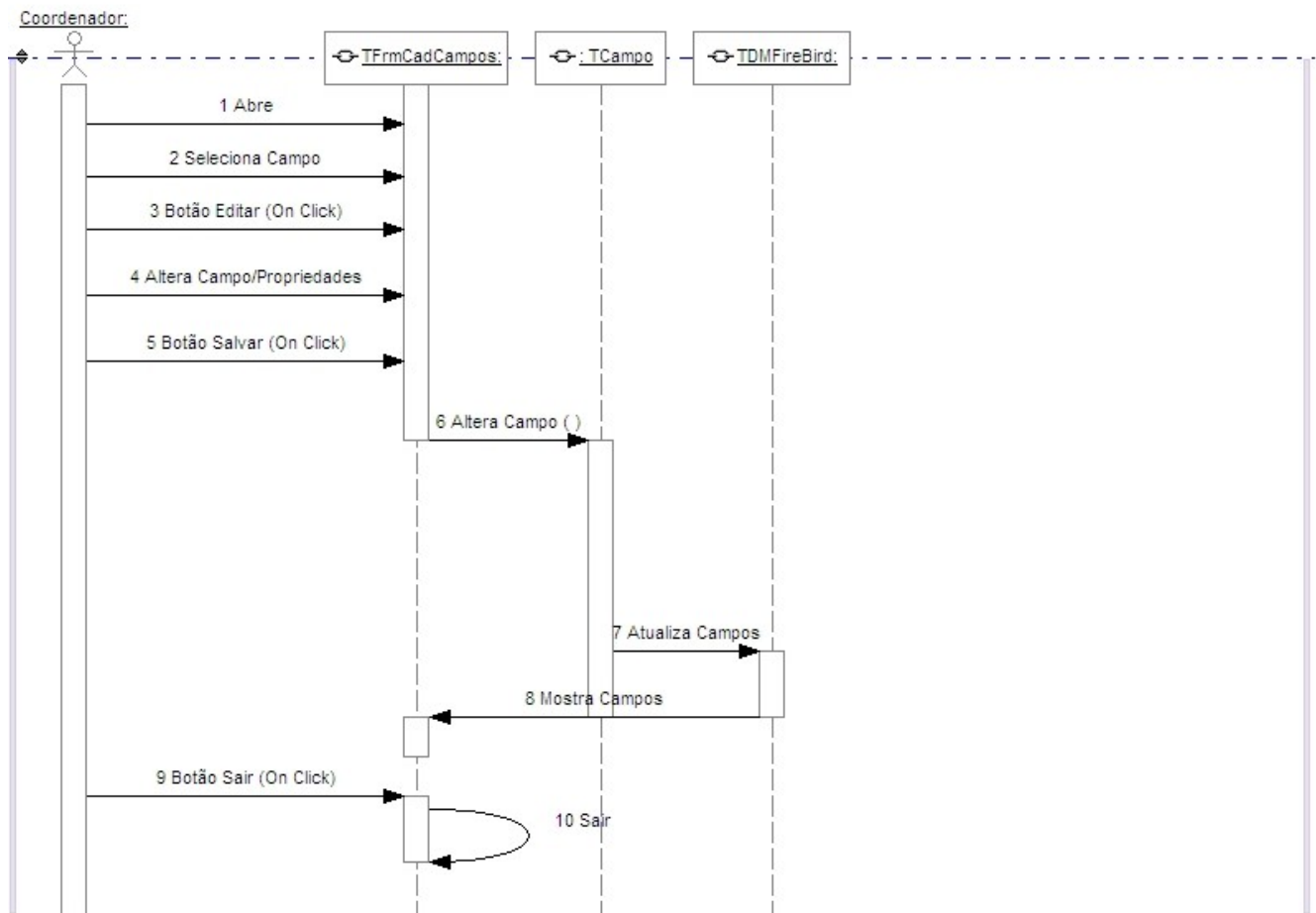
8.3.3 Diagrama de Seqüência Cadastro de Novo Campo



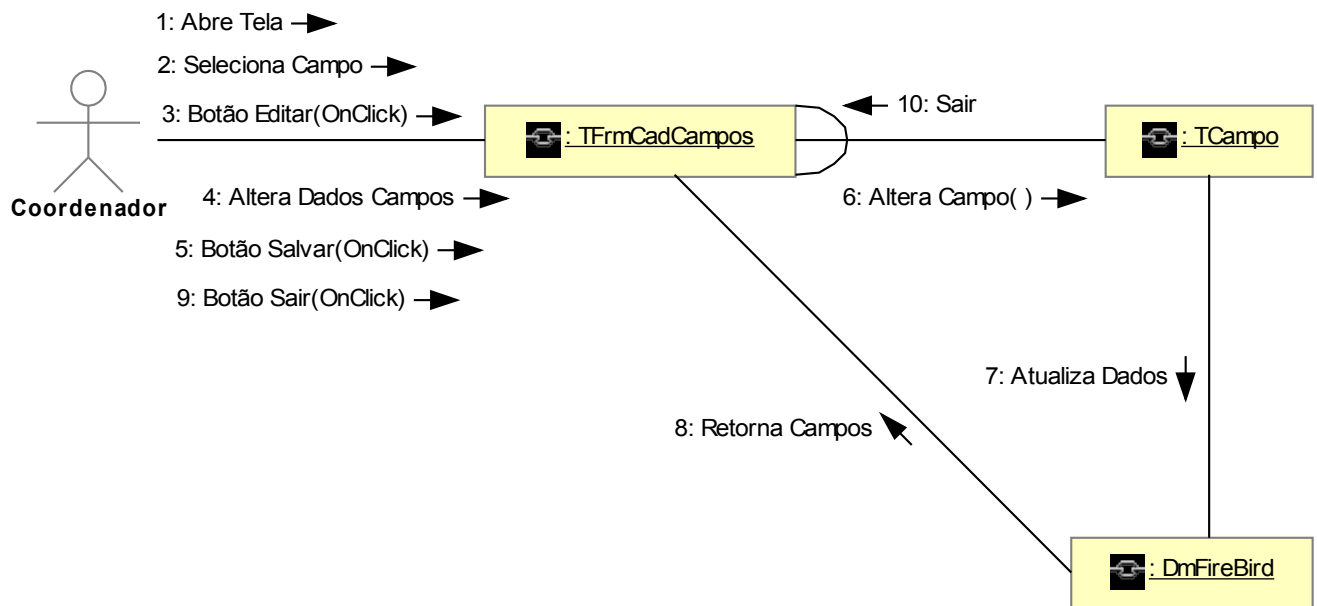
8.3.4 Diagrama de Colaboração Cadastro de Novo Campo



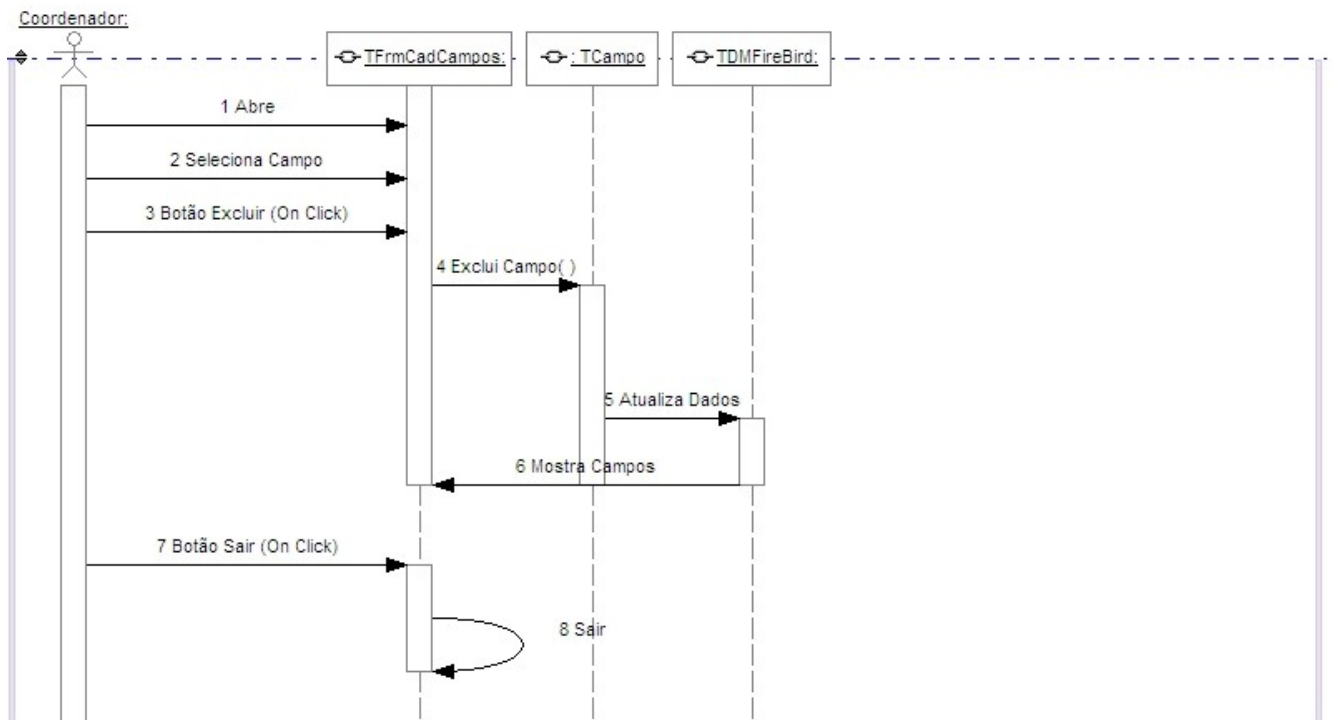
8.3.5 Diagrama de Sequência Alteração de Campo



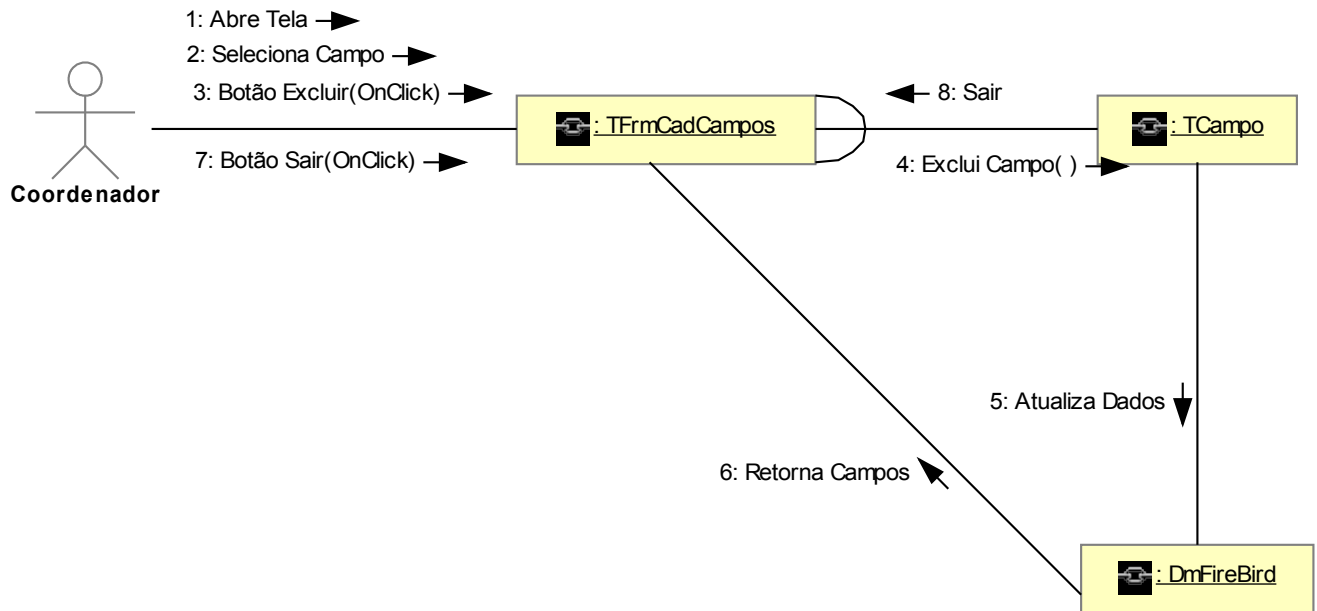
8.3.6 Diagrama de Colaboração Alteração de Campo



8.3.7 Diagrama de Seqüência Exclusão de Campo



8.3.8 Diagrama de Colaboração Exclusão de Campo



8.3.9 TELA FrmPesquisaCampos

FIGURA 7 – TELA PESQUISA DE CAMPOS

Pesquisar Campos

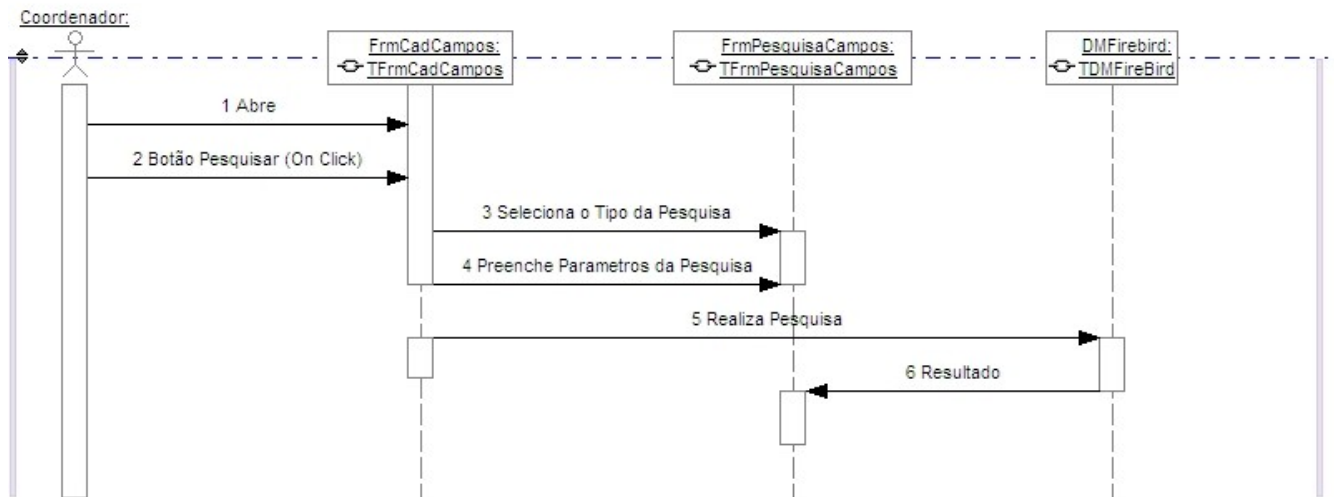
Pesquisar por:

☒ Nome do Campo ☐ Tipo do Campo

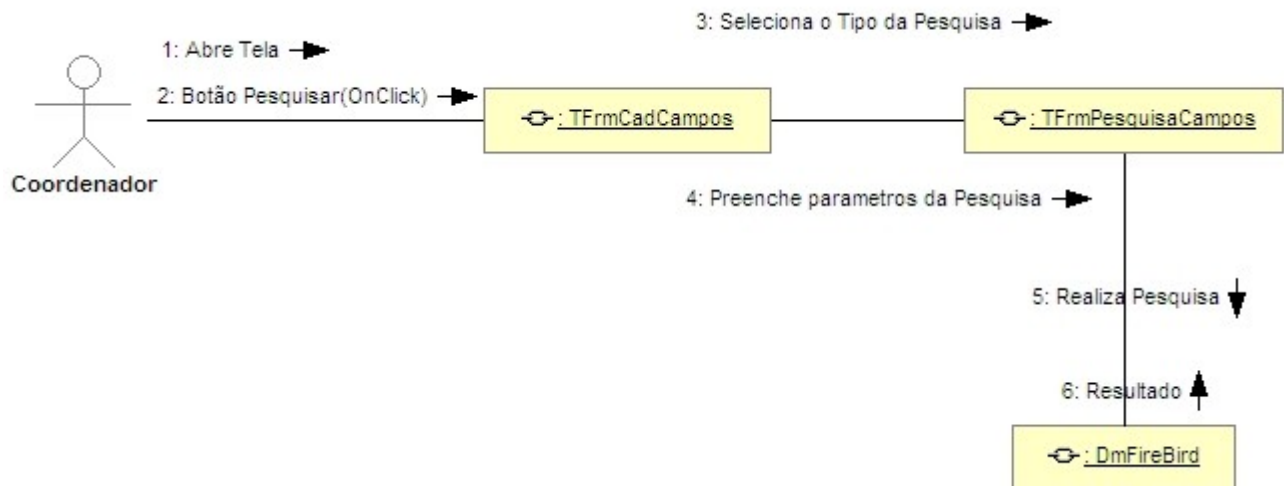
Código	Nome	Tam.	Tipo
2	CAMPO 2	20	Alfanum.
3	CAMPO 3	13	Alfanum.
4	CAMPO 4	4	Númerico
5	CAMPO 5	20	Alfanum.
6	CAMPO 6	8	Alfanum.
7	CAMPO 7	30	Alfanum.

SAIR

8.3.10 Diagrama de Seqüência Pesquisa de Campos



8.3.11 Diagrama de Colaboração Pesquisar Campos



8.3.12 TELA FrmTipoDoc

FIGURA 8 – TELA CADASTRO DE TIPOS DE DOCUMENTO

Cadastrar Tipos de Documento

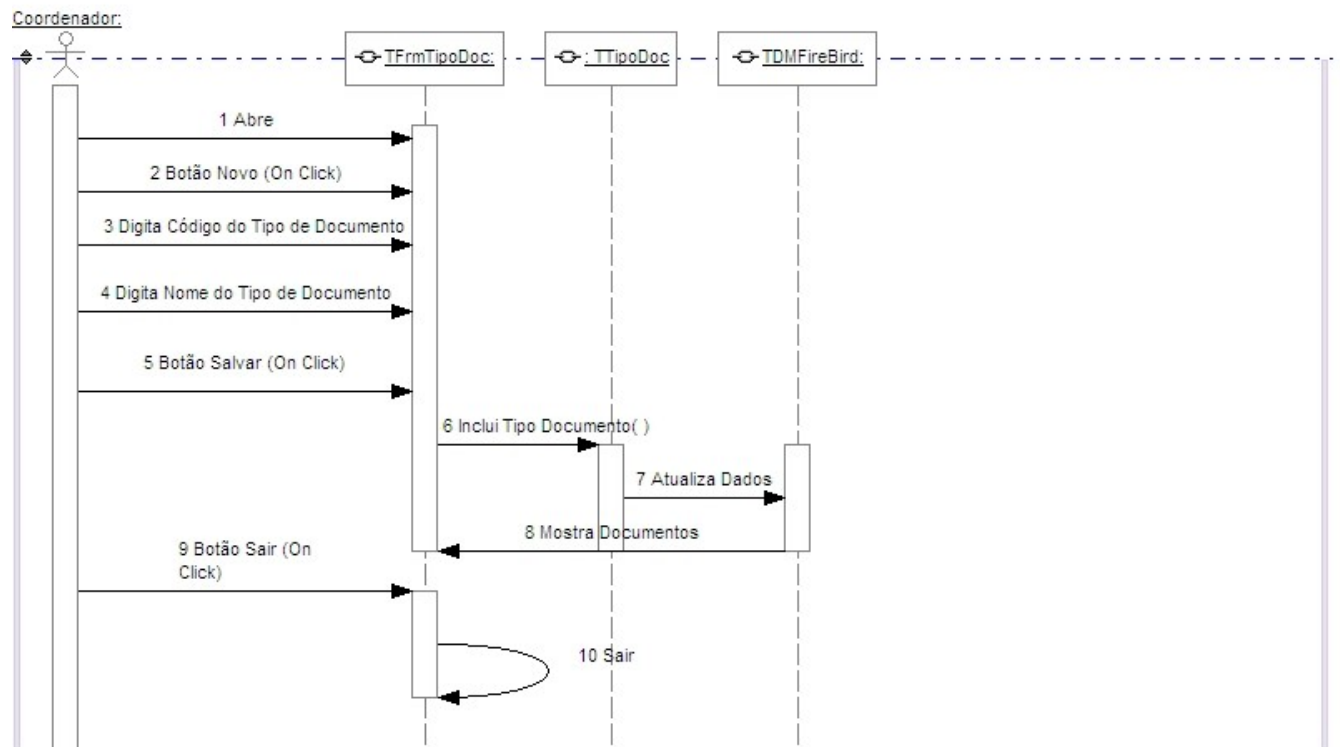
Código **Nome do Documento**

Vincular Campos

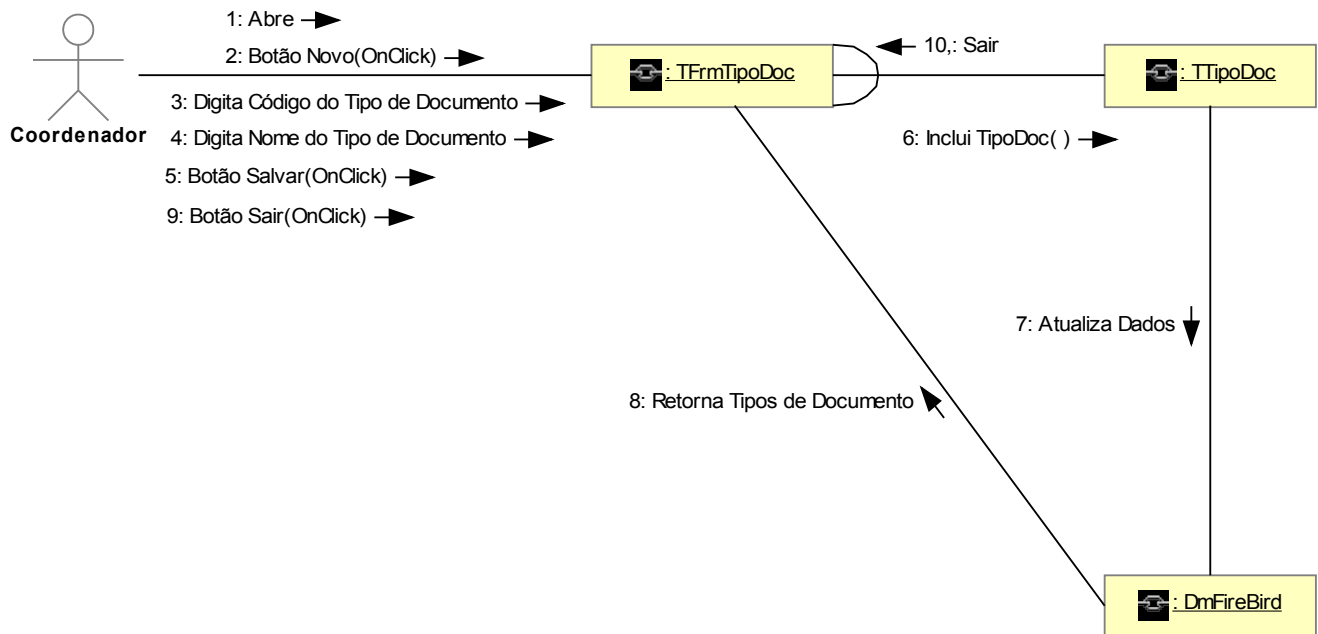
Código	Descrição
NFET	NOTA FISCAL ENTRADA
NFSD	NOTA FISCAL SAIDA
OFCD	OFICIO
DDTH	TESTE

NOVO EDITAR SALVAR EXCLUIR CANCELAR SAIR

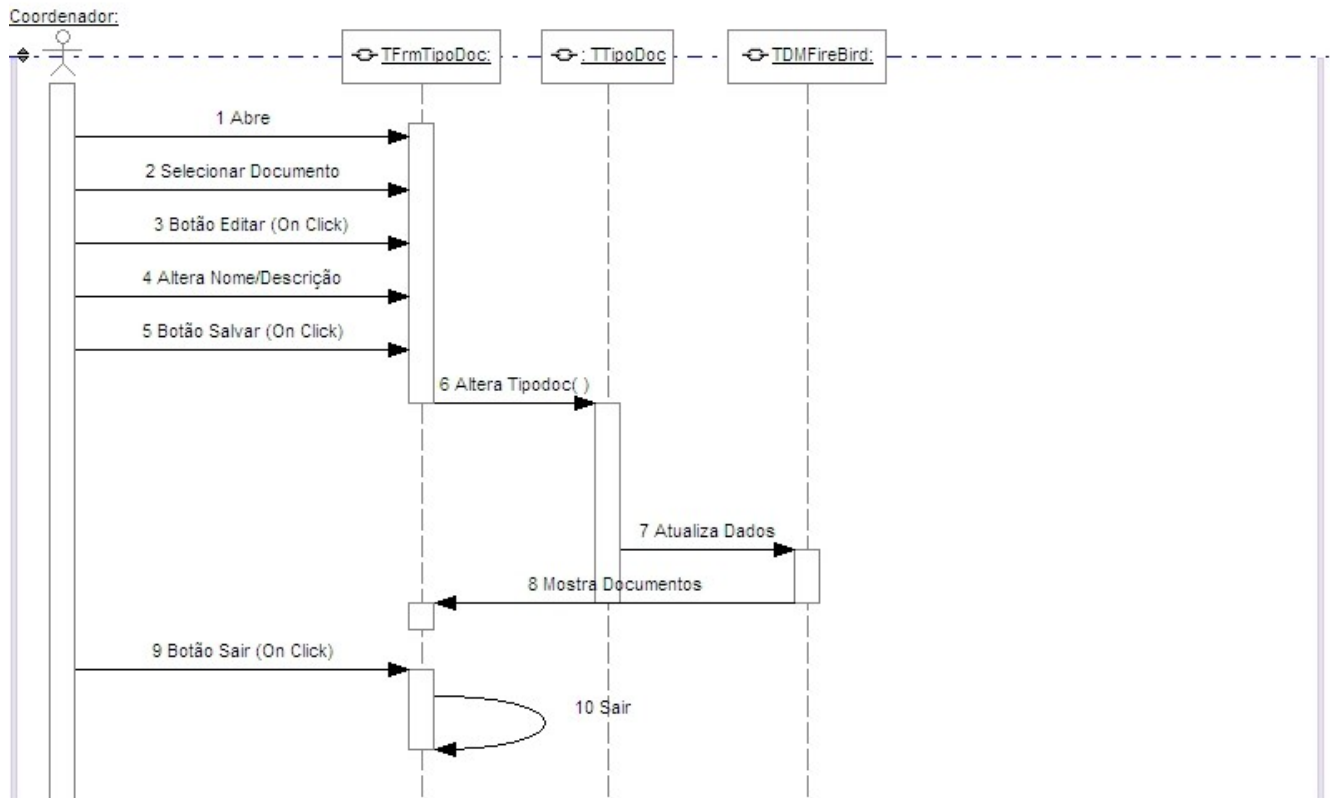
8.2.13 Diagrama de Seqüência Cadastro de Novo Tipo de Documento



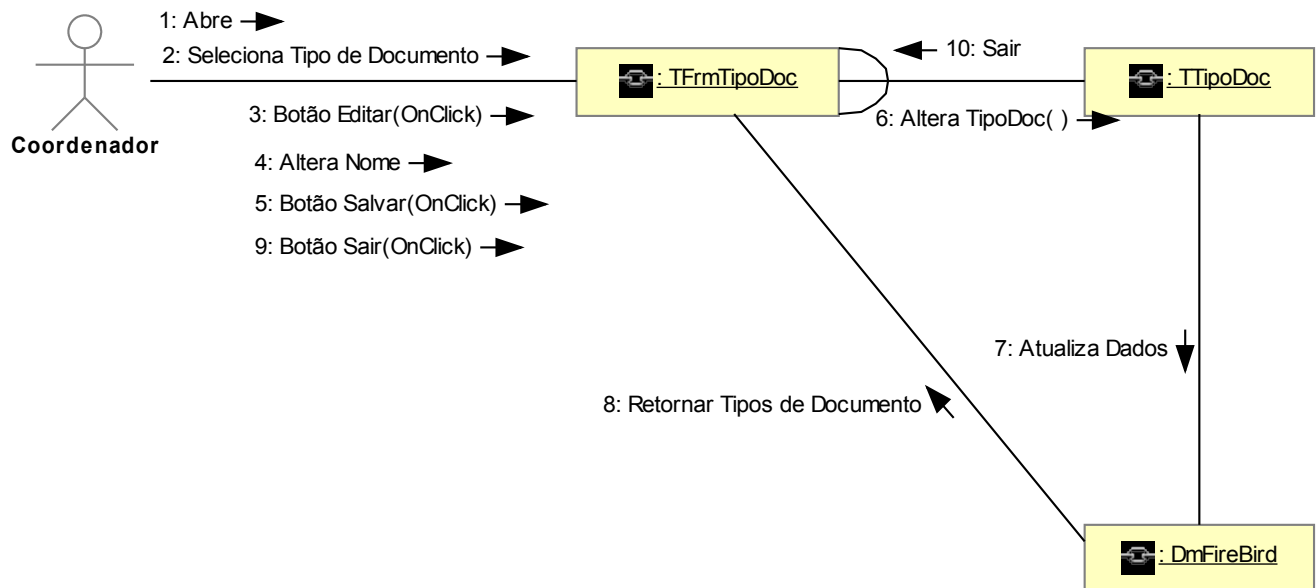
8.3.14 Diagrama de Colaboração Cadastro de Novo Tipo de Documento



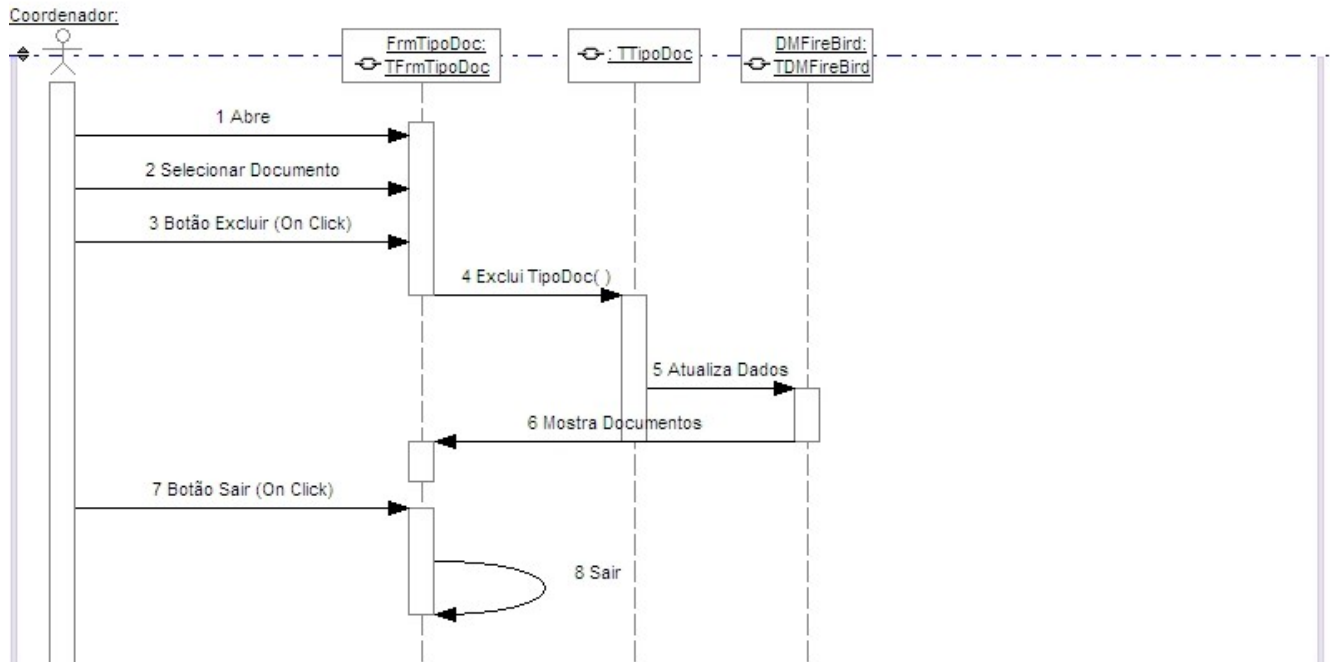
8.3.15 Diagrama de Seqüência Alteração de Tipo de Documento



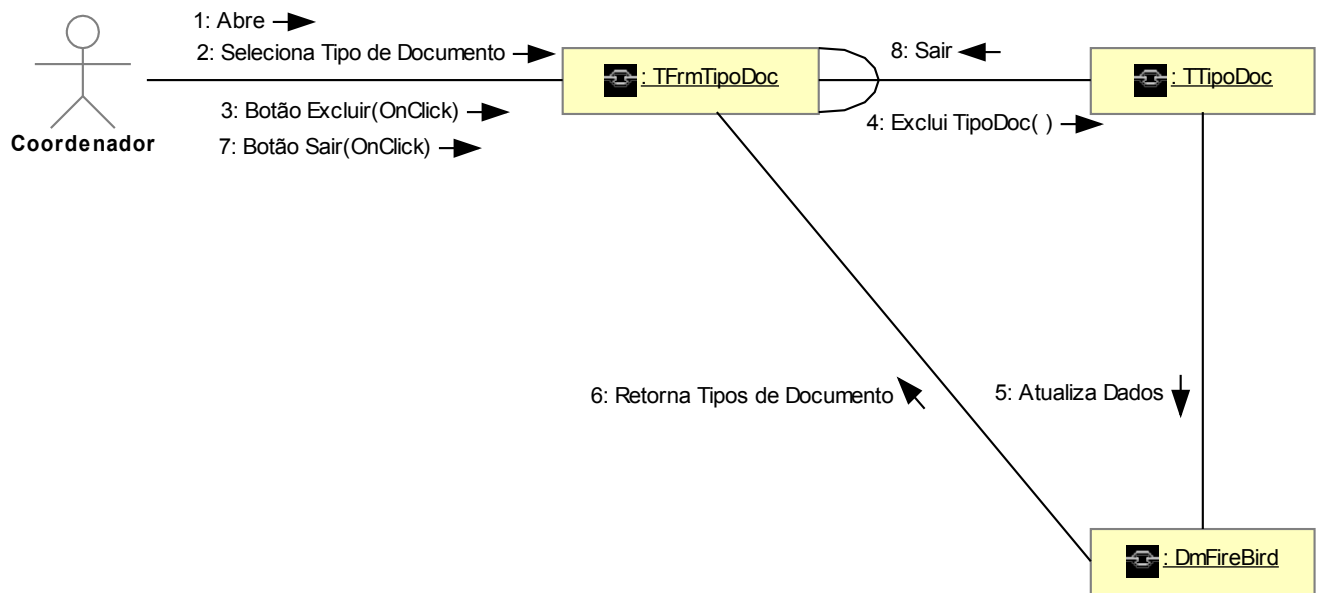
8.3.16 Diagrama de Colaboração Alteração de Tipo de Documento



8.3.17 Diagrama de Seqüência Exclusão de Tipo de Documento



8.3.18 Diagrama de Colaboração Exclusão de Tipo de Documento



8.3.19 TELA FrmVincularCampos

FIGURA 9 – TELA DE VINCULAÇÃO DE CAMPOS

Vincular Campos

Documento
 NOTA FISCAL ENTRADA

Código	Nome	Tam.
6	CAMPO 6	
7	CAMPO 7	
8	CAMPO 8	
9	CAMPO 9	
10	ENDERECO	
11	CIDADE	
12	NOME	

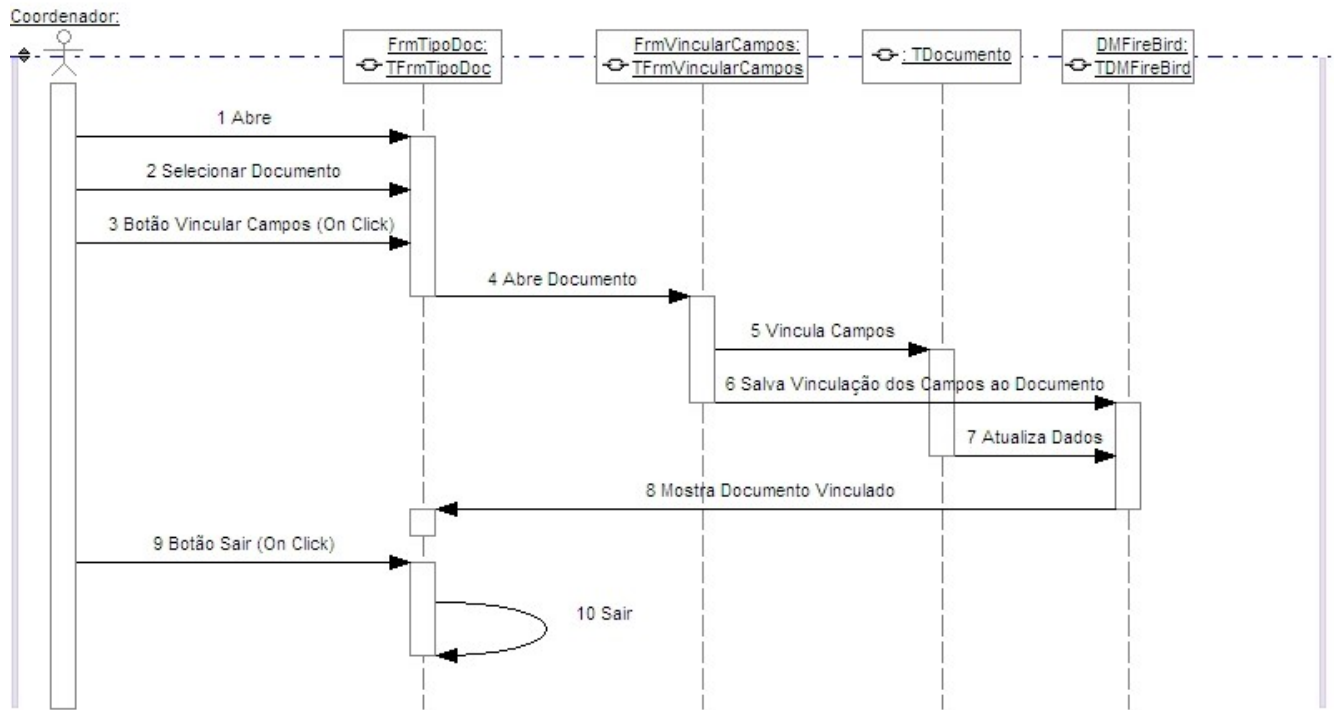
> >> < <<

Código	Nome	Obrigatório	Repetitivo
2	CAMPO 2	Não	Não
3	CAMPO 3	Sim	Sim
4	CAMPO 4	Não	Não
5	CAMPO 5	Sim	Não

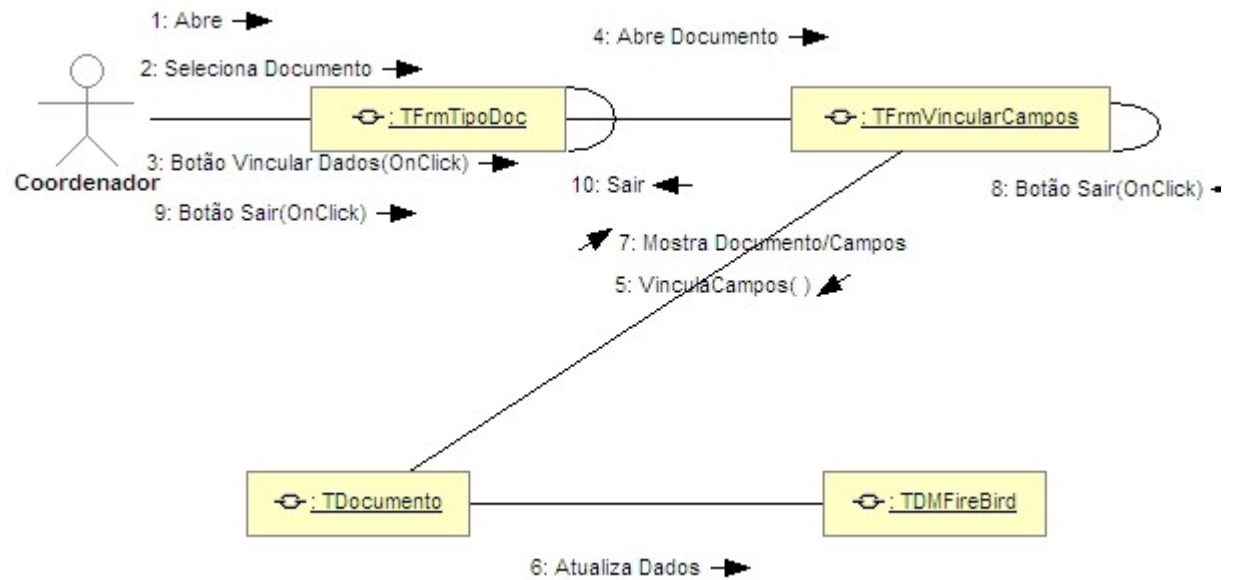
Obrigatório Não-Obrig. Repetitivo Não-Repet.

SAIR

8.3.20 Diagrama de Seqüência Vincular Campos



8.3.21 Diagrama de Colaboração Vincular Campos

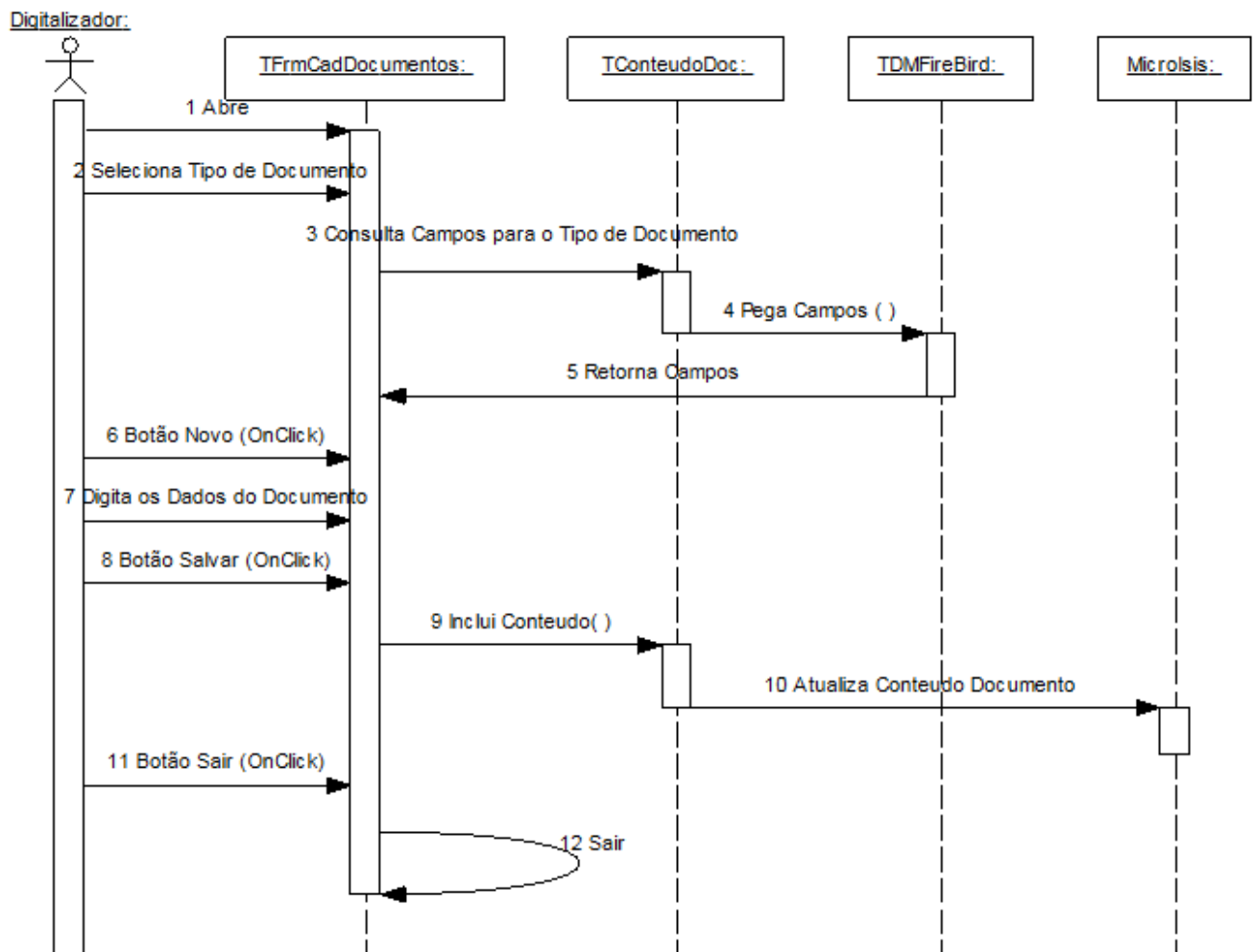


8.3.22 TELA FrmCadDocumentos

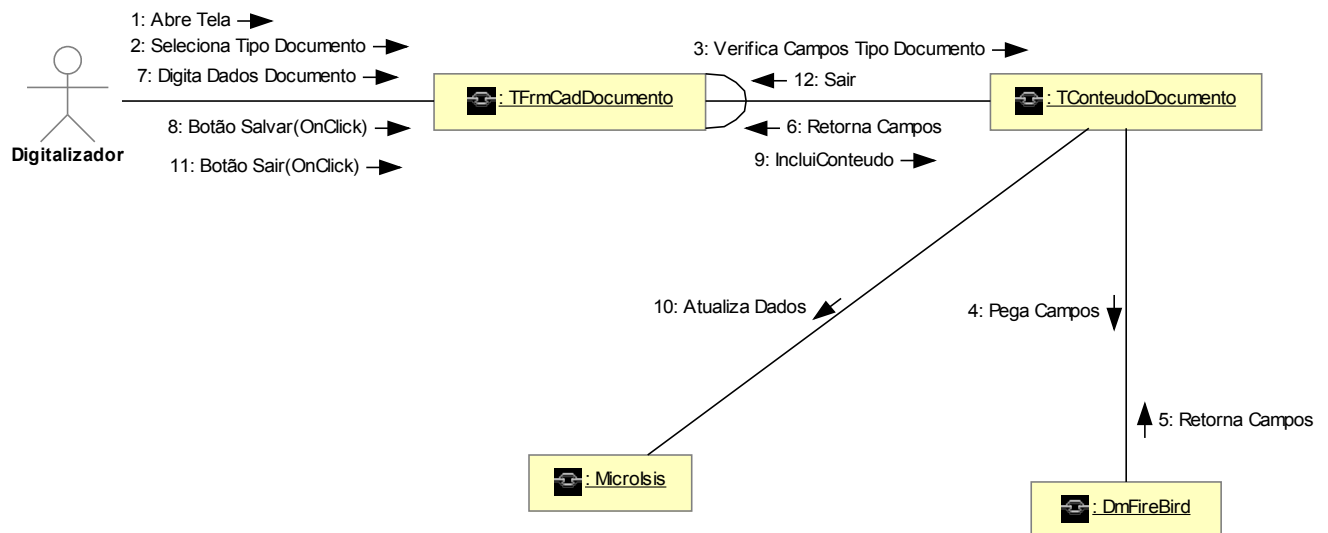
FIGURA 10 – TELA DE VINCULAÇÃO DE CAMPOS

The screenshot shows a software window titled "Cadastrar Documentos". At the top, there is a header bar with standard window controls (minimize, maximize, close). Below the header, the window is divided into several sections. On the left, there is a "Código" field and a "Tipo de Documento" dropdown menu currently set to "NFET - NOTA FISCAL ENTRADA". To the right of the dropdown is a section labeled "Ordenar por:" with two radio buttons: "Nome" (selected) and "Código". The main area of the window is a large light blue rectangle containing four text input fields. On the left side of this area, there are two fields: "CAMPO 5 (Obrig.)" and "CAMPO 4". On the right side, there are two fields: "CAMPO 2" and "CAMPO 3 (Obrig.)". The "CAMPO 3 (Obrig.)" field is currently selected, indicated by a blue highlight. To the right of the "CAMPO 3 (Obrig.)" field is a small button with a "+" sign. At the bottom of the window, there is a toolbar with six icons and their corresponding labels: "NOVO" (document icon), "EDITAR" (pencil icon), "SALVAR" (floppy disk icon), "EXCLUIR" (trash can icon), "CANCELAR" (prohibited sign icon), and "SAIR" (exit door icon).

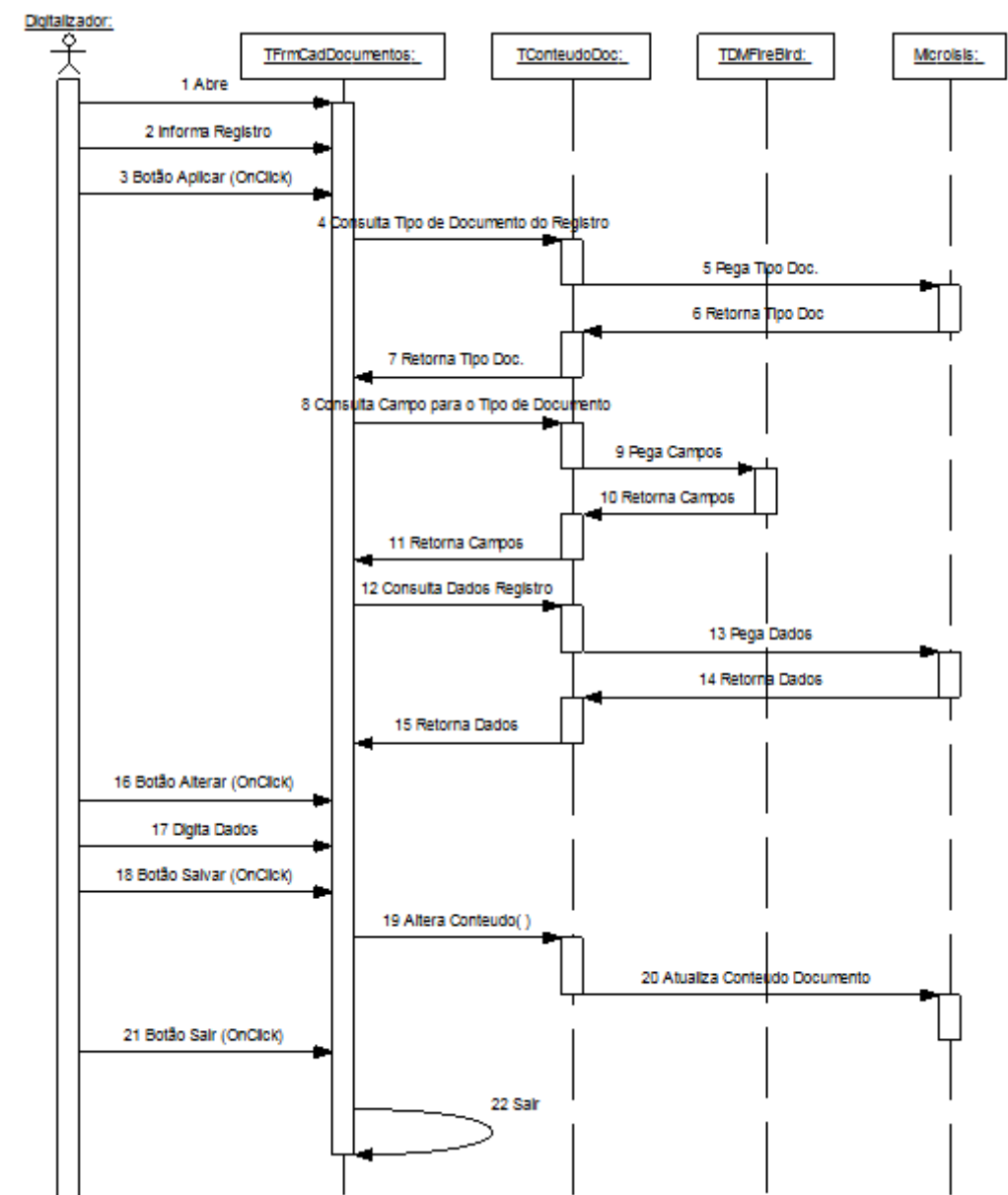
8.3.23 Diagrama de Seqüência Cadastro de Novo Documento



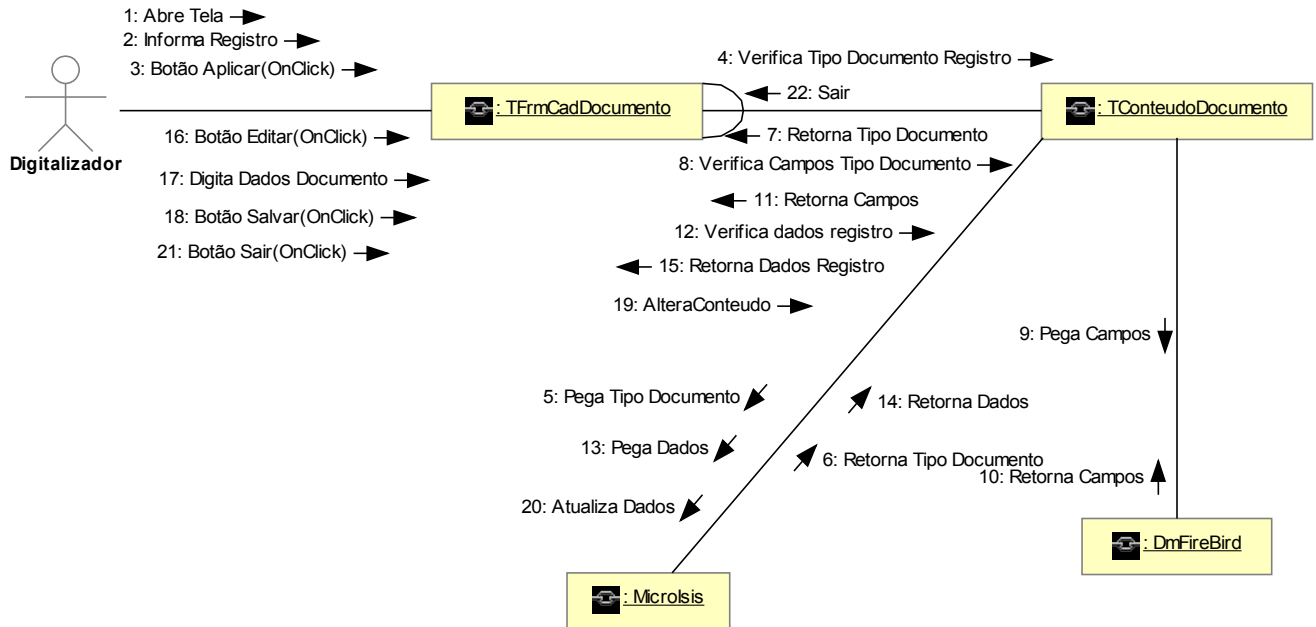
8.3.24 Diagrama de Colaboração Cadastro de Novo Documento



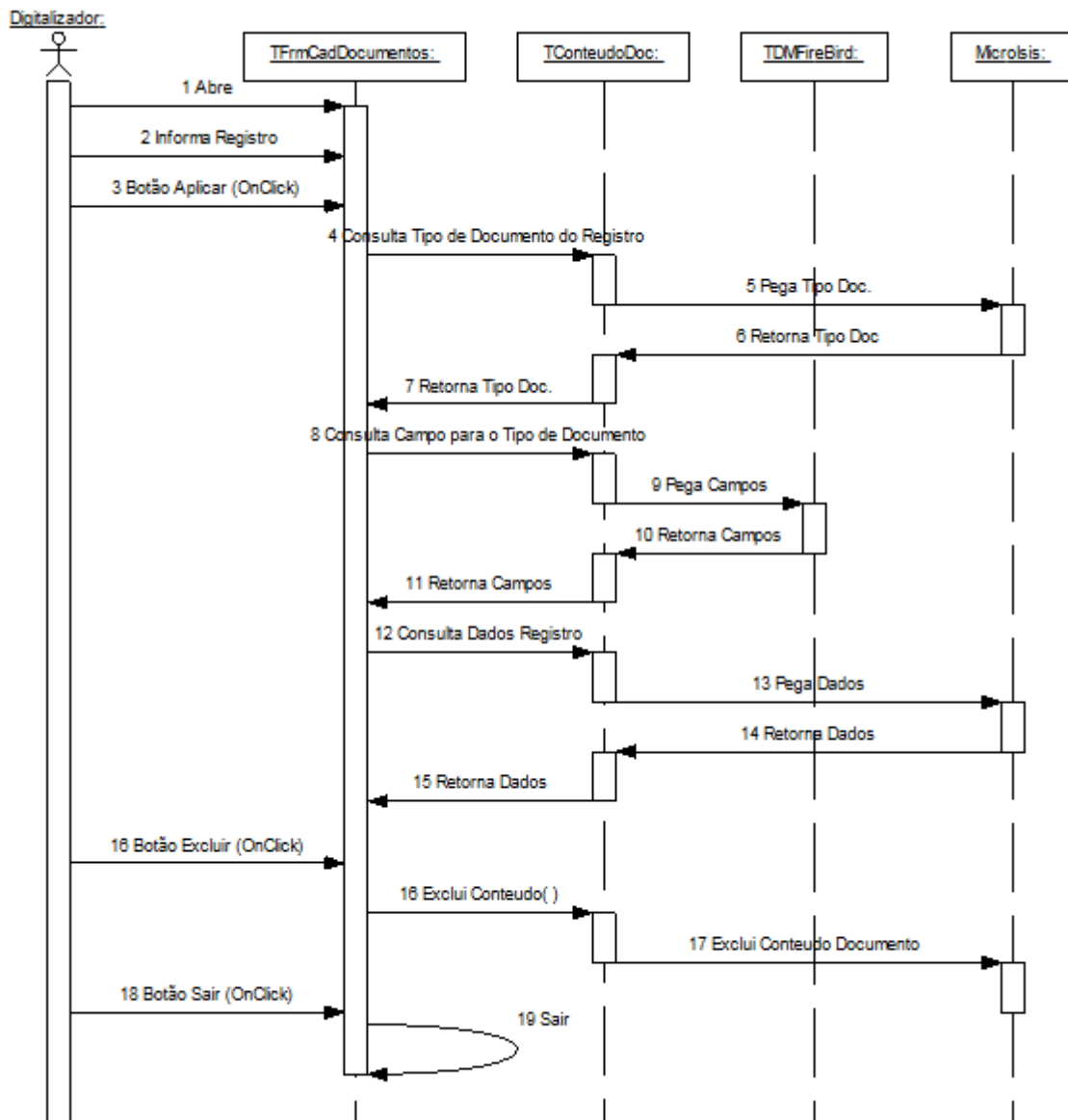
8.3.25 Diagrama de Seqüência Alteração de Documento



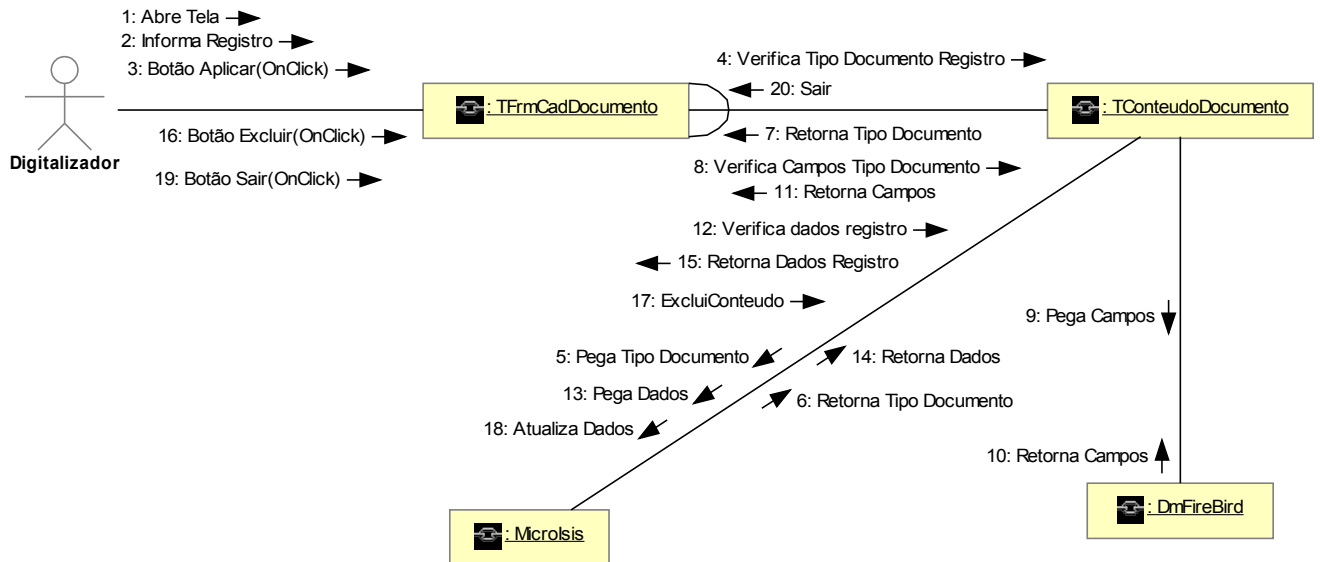
8.3.26 Diagrama de Colaboração Alteração de Documento



8.3.27 Diagrama de Seqüência Exclusão de Documento



8.3.28 Diagrama de Colaboração Exclusão de Documento



8.3.29 TELA FrmImagensDoc

FIGURA 11 – TELA IMAGENS DO DOCUMENTO

Imagens do Documento

Código 2 **Tipo do Documento** NOTA FISCAL ENTRADA

Dados do Documento

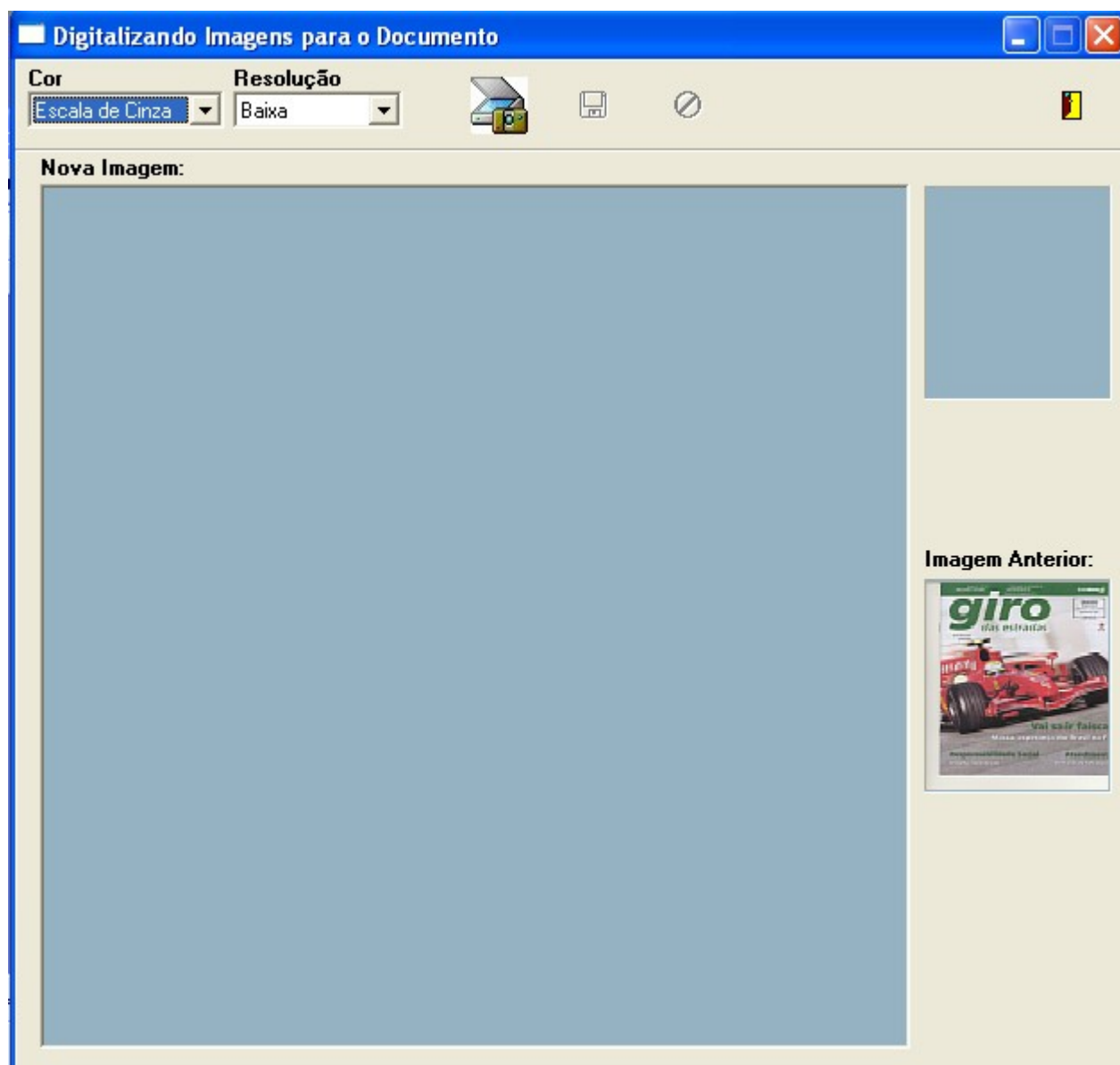
CAMPO 5
ALTERADO
CAMPO 2
MAIS UM
CAMPO 4
1212
CAMPO 3
NOVAMENTE

Registro Seleccionado = 2 Imagem Seleccionada =

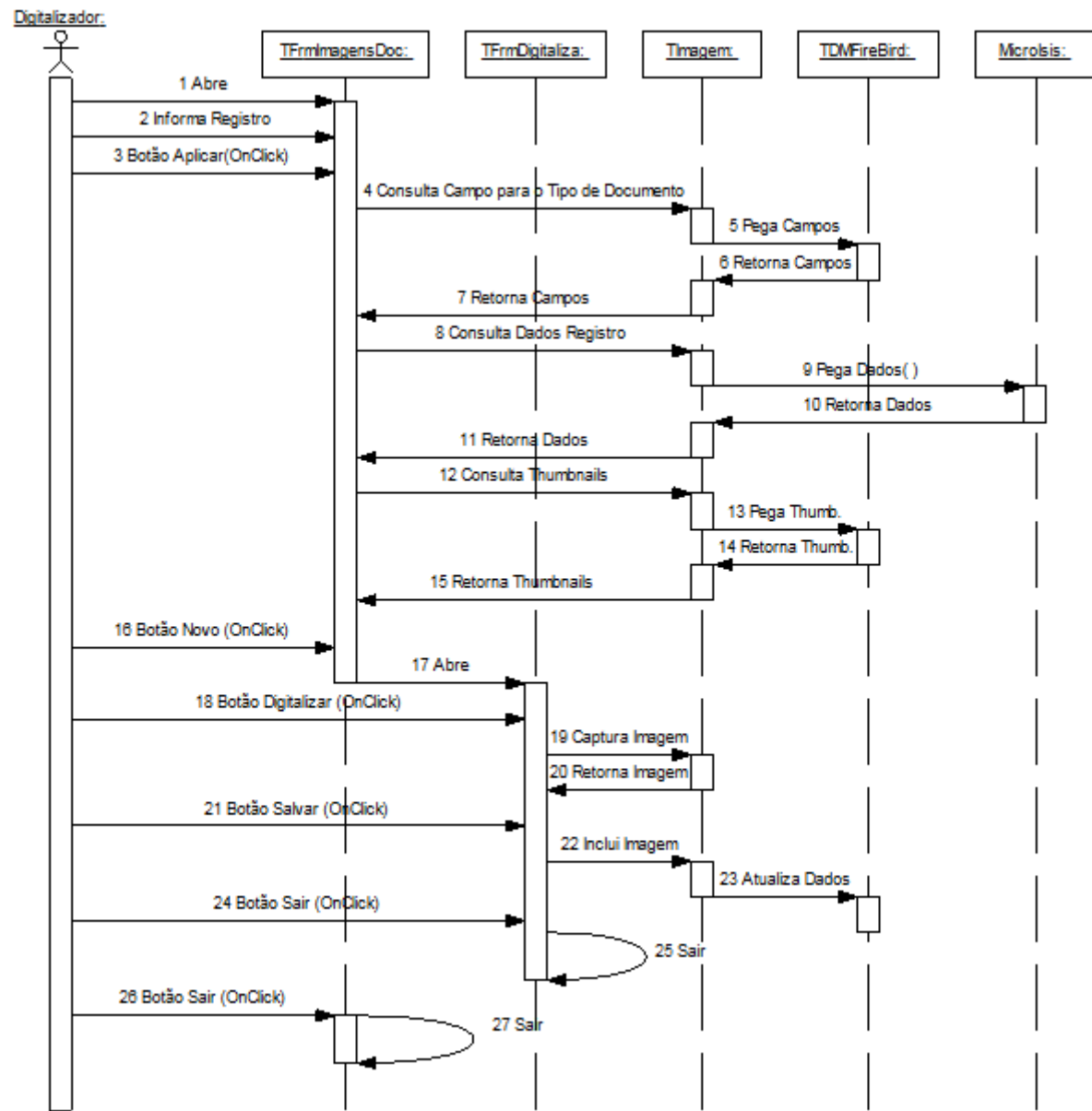
NOVO EDITAR EXCLUIR SAIR

8.3.30 TELA FrmDigitaliza

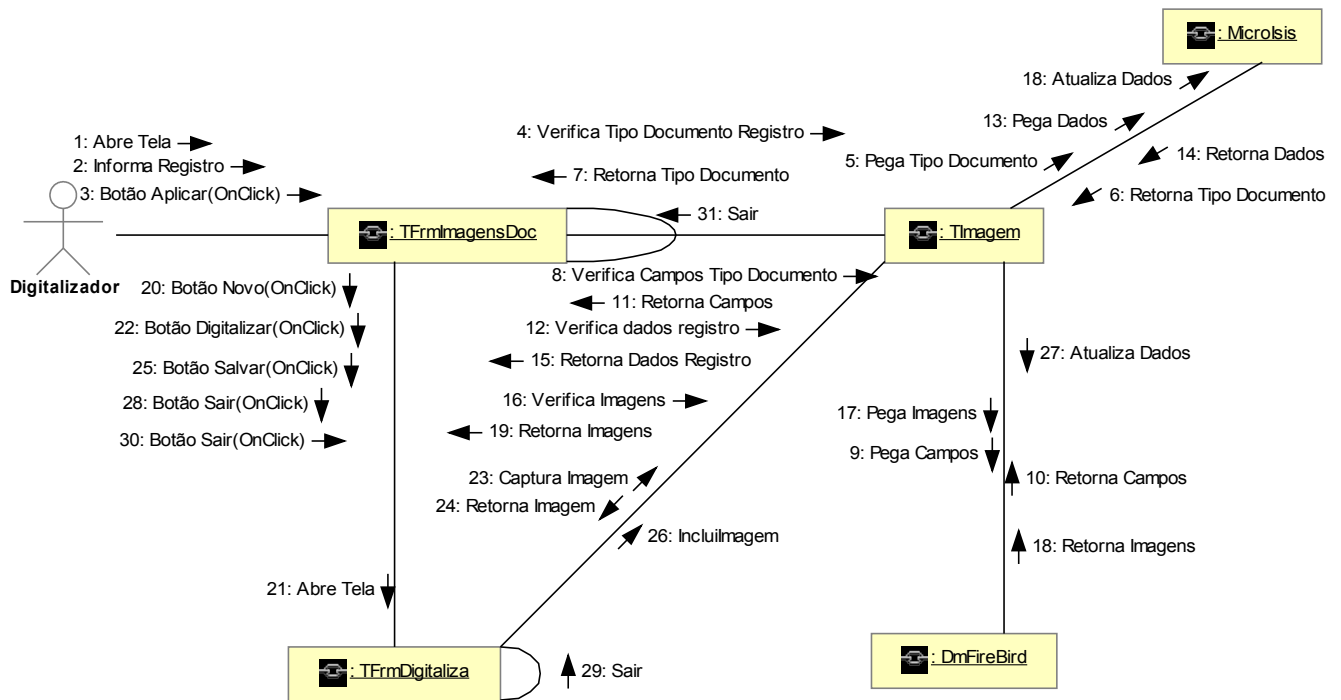
FIGURA 12 – TELA IMAGENS DO DOCUMENTO



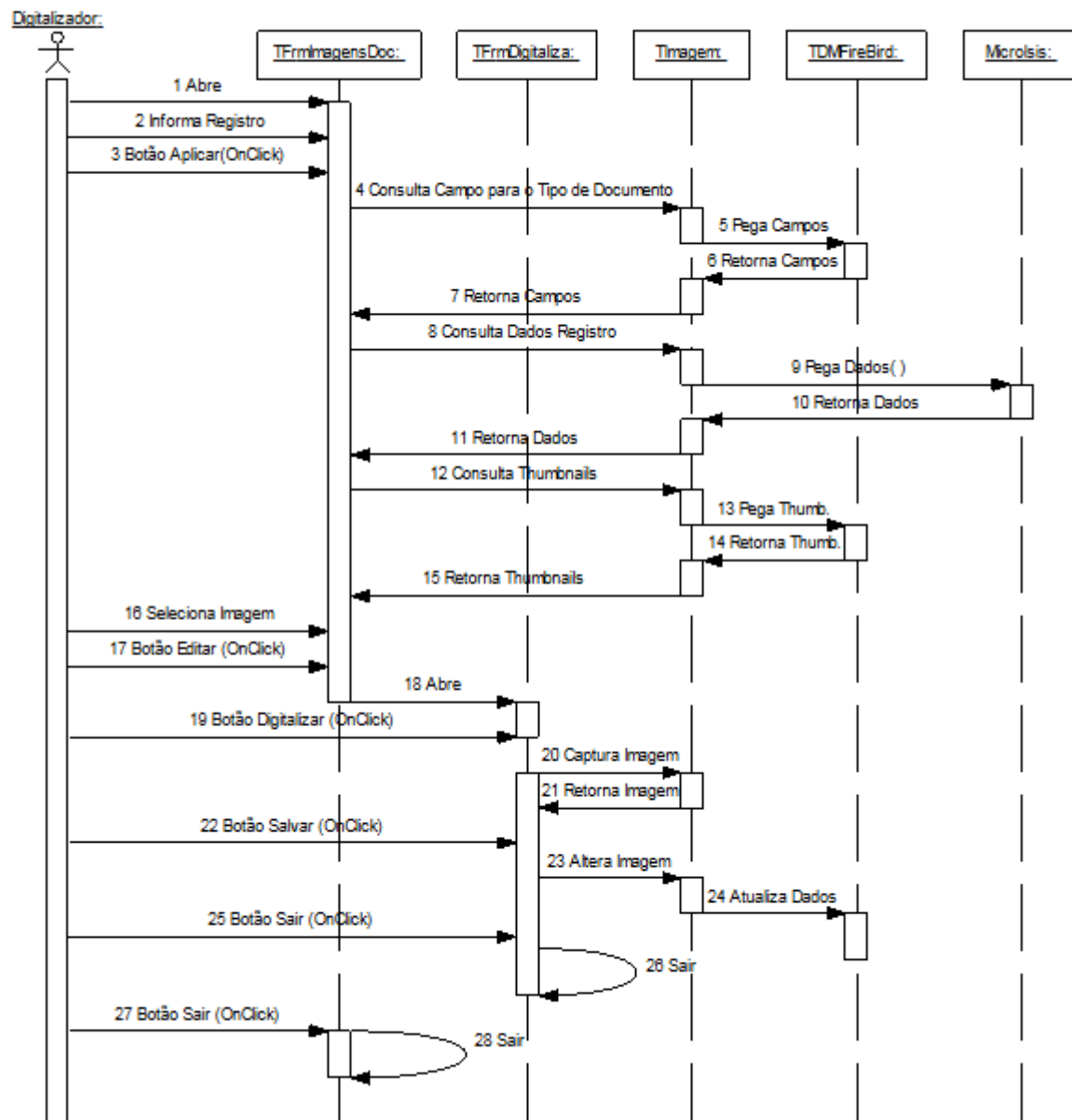
8.3.31 Diagrama de Seqüência Cadastro de Novas Imagens do Documento



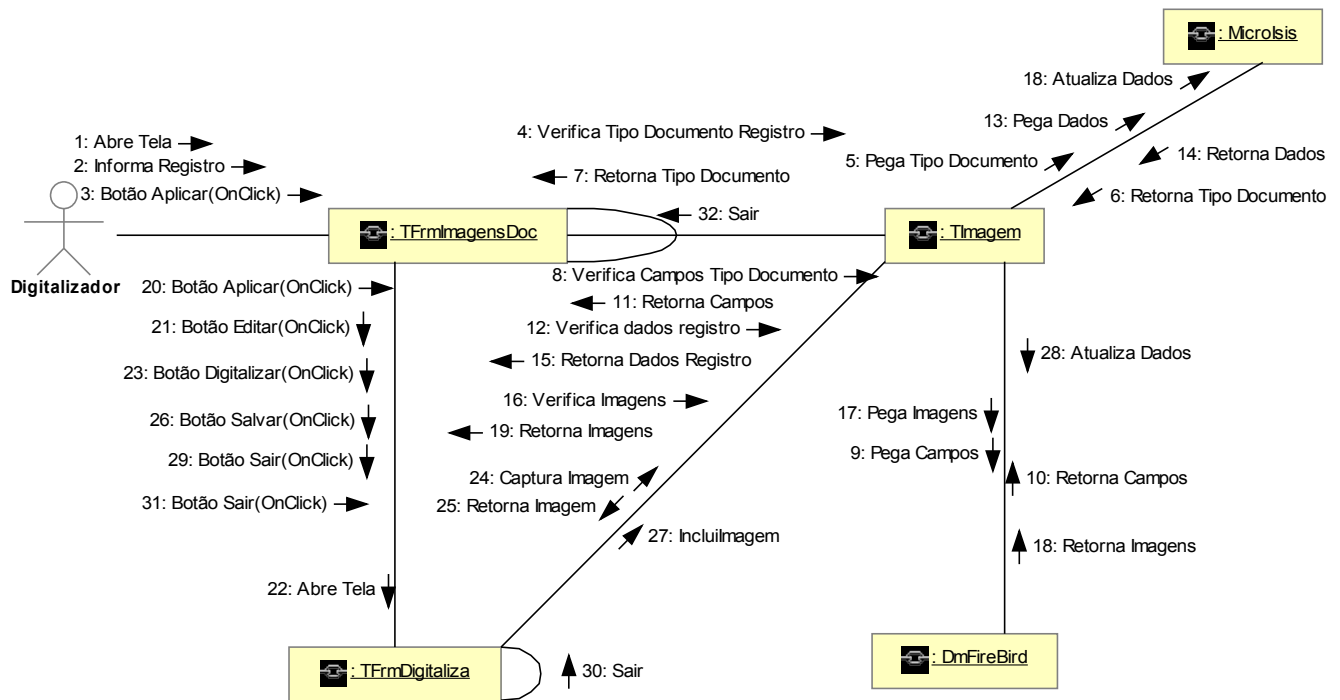
8.3.32 Diagrama de Colaboração Cadastro de Imagens do Documento



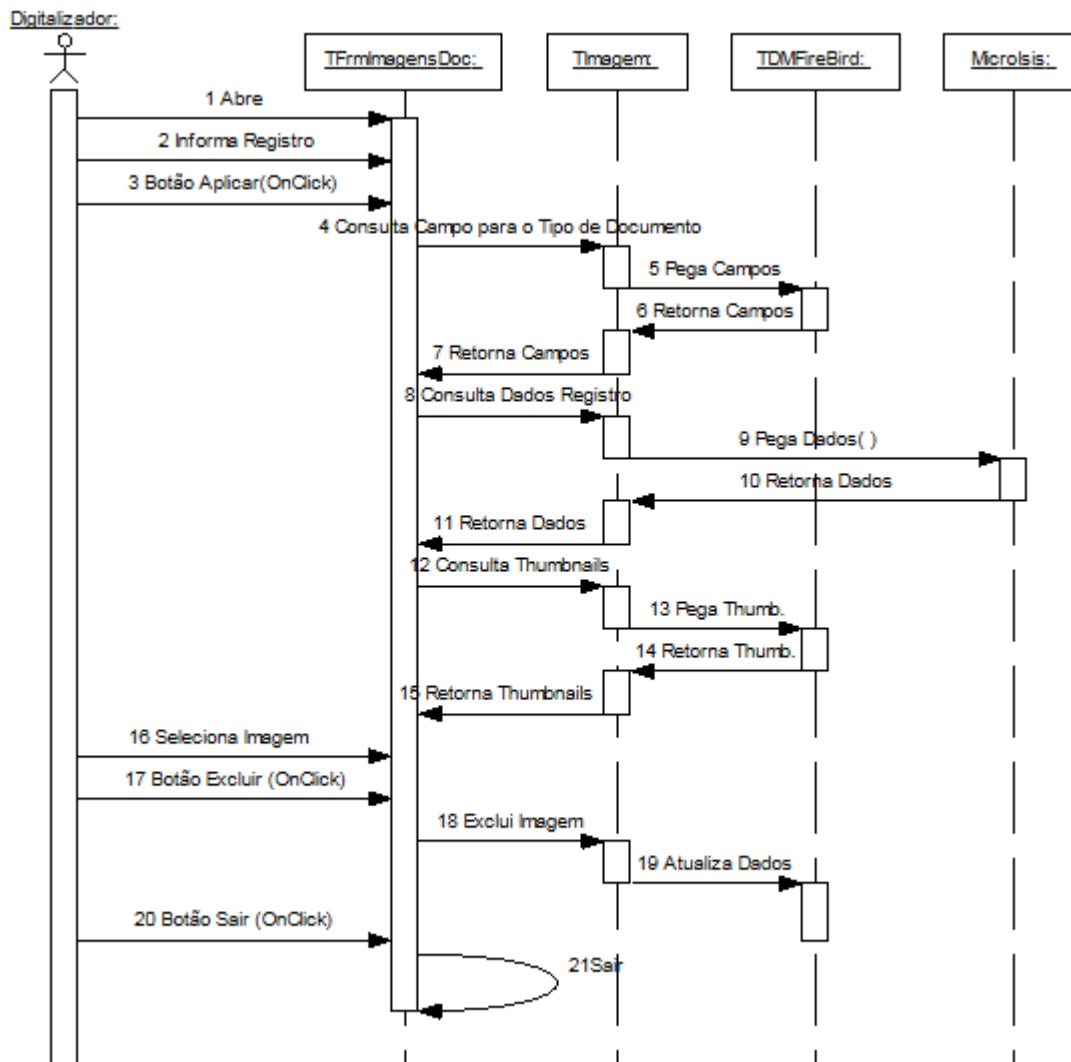
8.3.33 Diagrama de Seqüência Alteração de Imagens do Documento



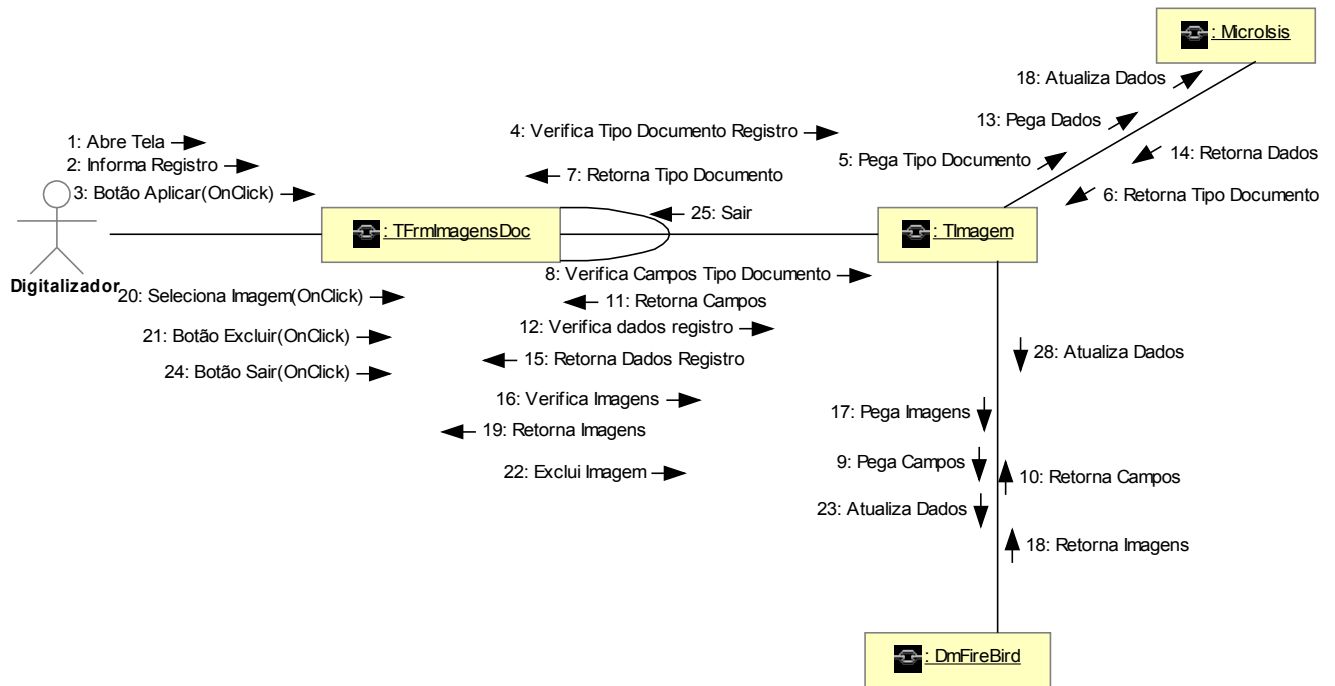
8.3.34 Diagrama de Colaboração Alteração de Imagens do Documento



8.3.35 Diagrama de Seqüência Exclusão de Imagens do Documento



8.3.36 Diagrama de Colaboração Exclusão de Imagens do Documento



8.3.37 TELA FrmPesquisaDoc

FIGURA 13 – TELA PESQUISA DOCUMENTO

DigiDoc - Pesquisa de Documentos

Tipo de Documento
NFET - NOTA FISCAL ENTRADA

Ordenar por:
☒ Nome ☐ Código

Critérios Pesquisa

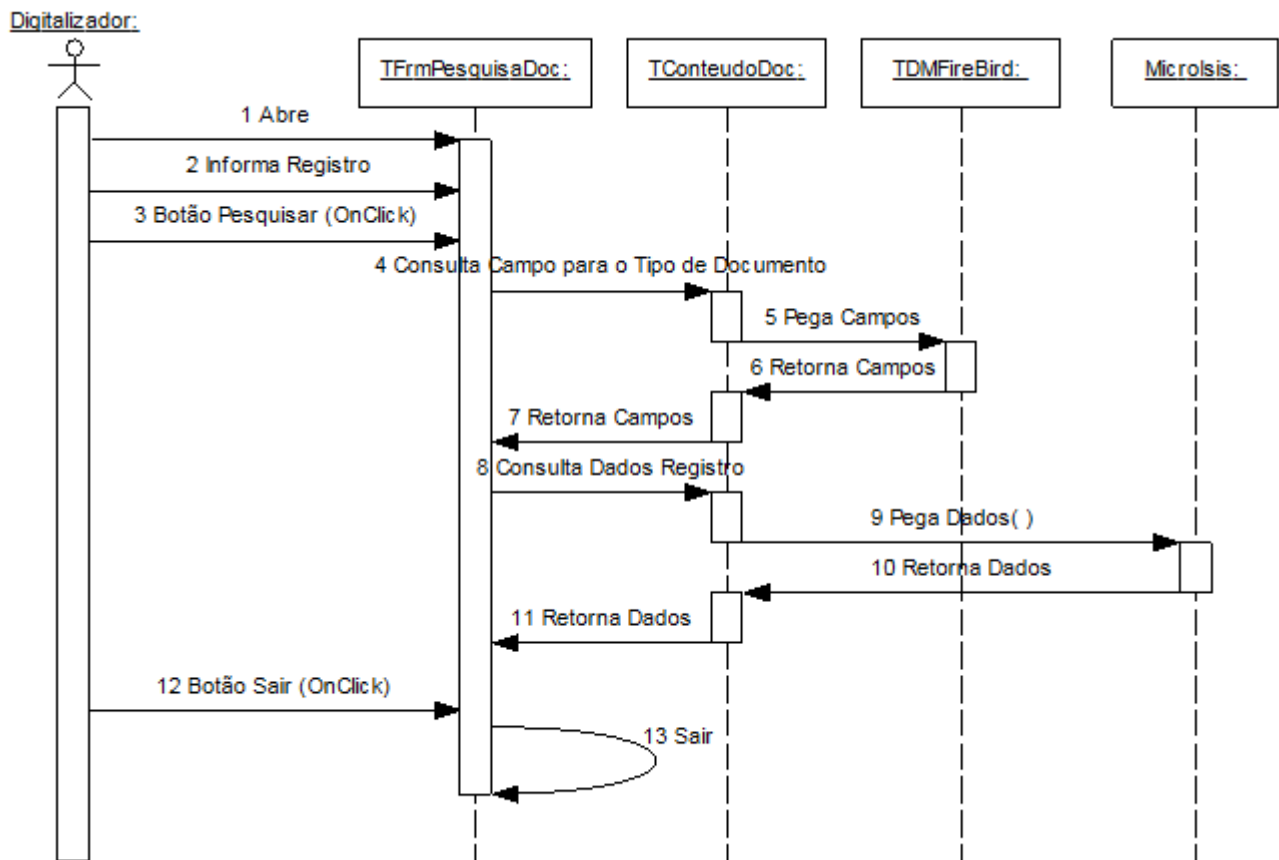
Por:
☒ Palavra Inteira ☐ Termo

Contém
☒ Qualquer Palavra ☐ Todas as Palavras

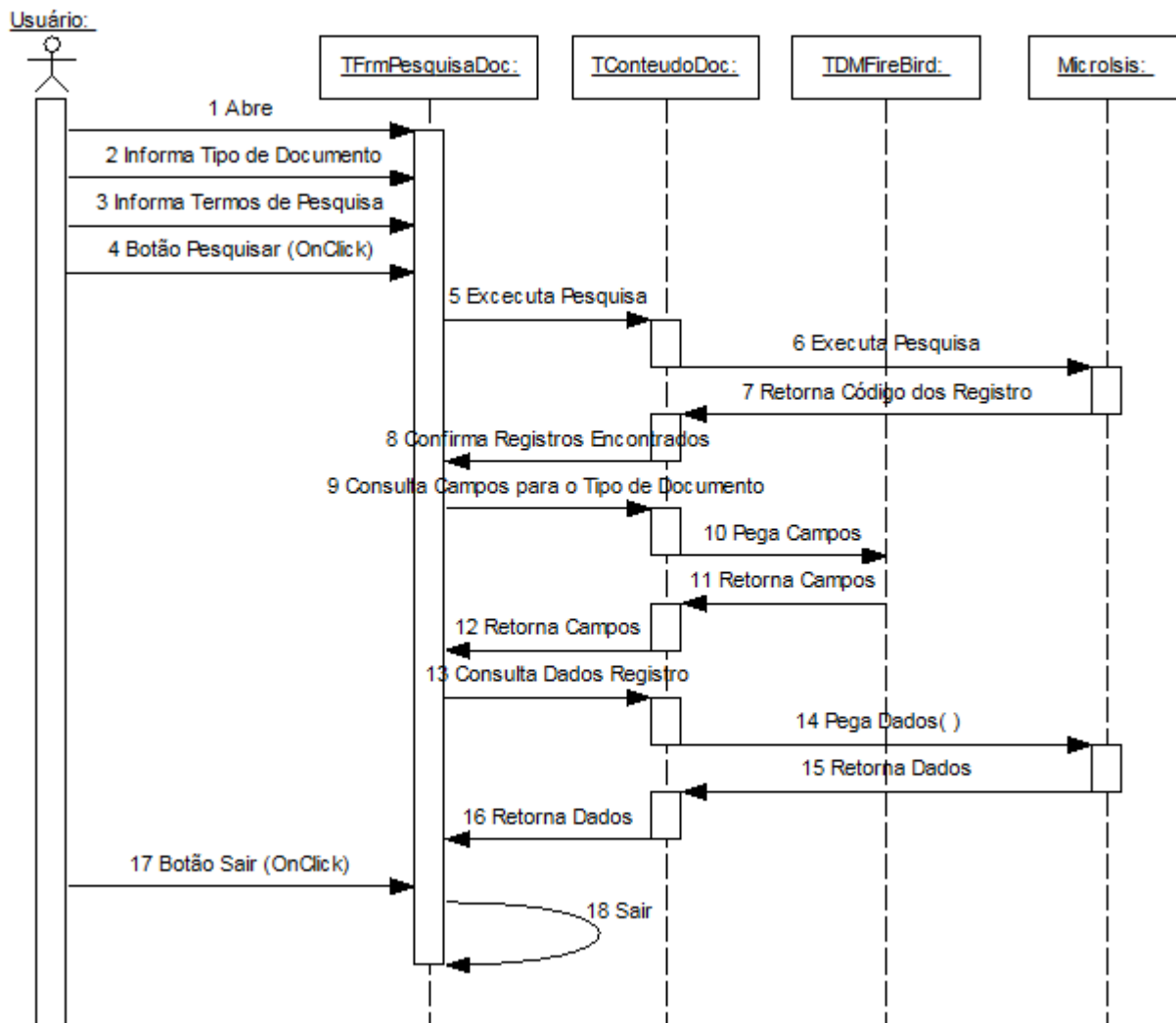
Limpar Critérios Pesquisar

SAIR

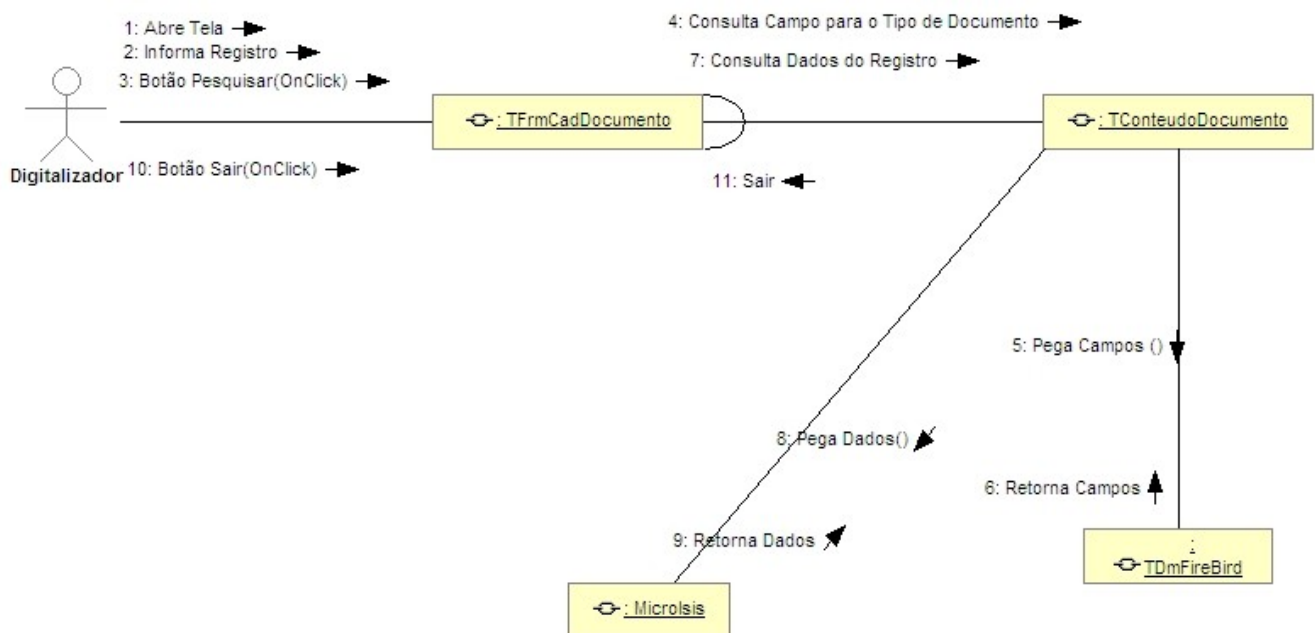
8.3.38 Diagrama de Seqüência Pesquisa de Documentos (Informando Código Registro)



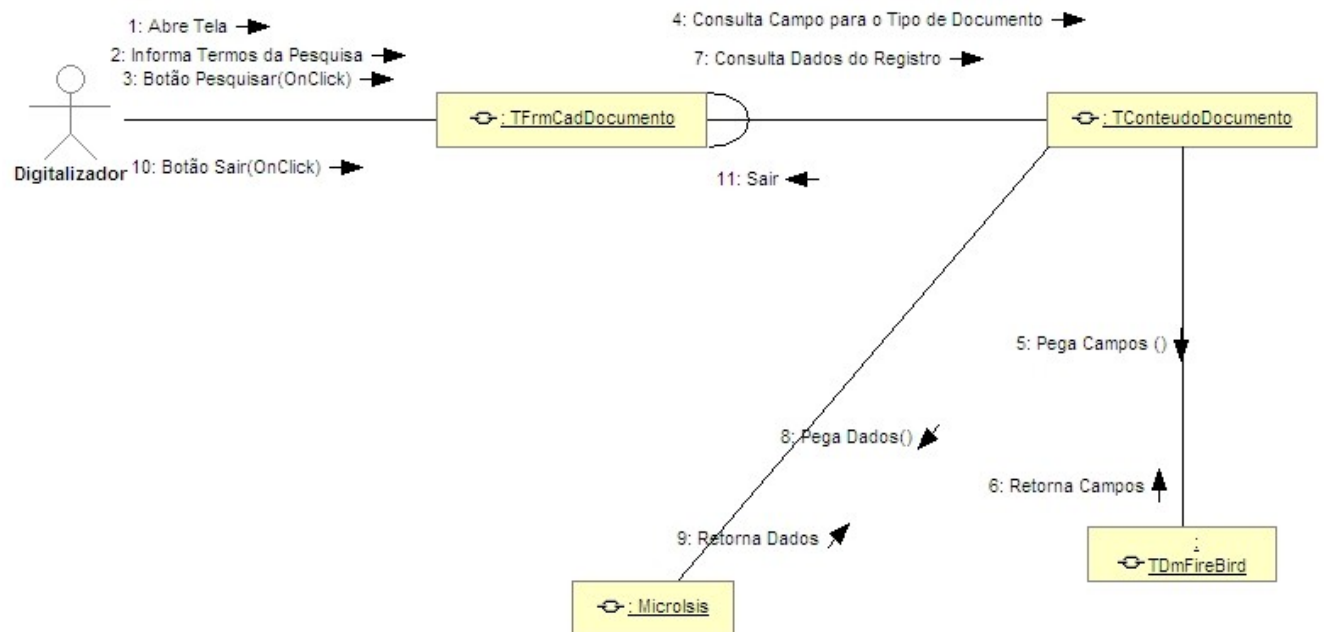
8.3.39 Diagrama de Seqüência Pesquisa de Documentos (Informando Termos de Pesquisa)



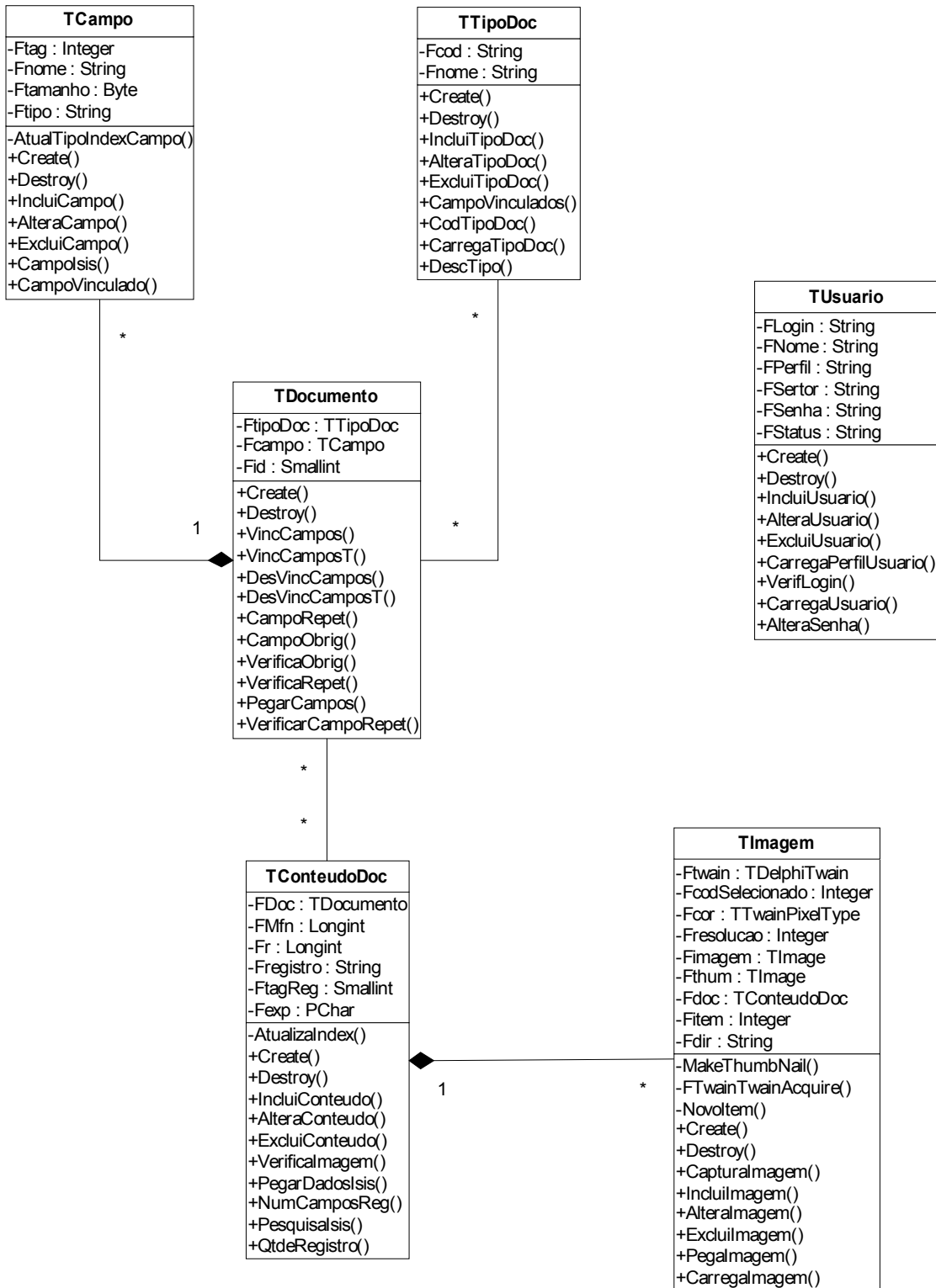
8.3.40 Diagrama de Colaboração Pesquisa de Documentos (Informando Código Registro)



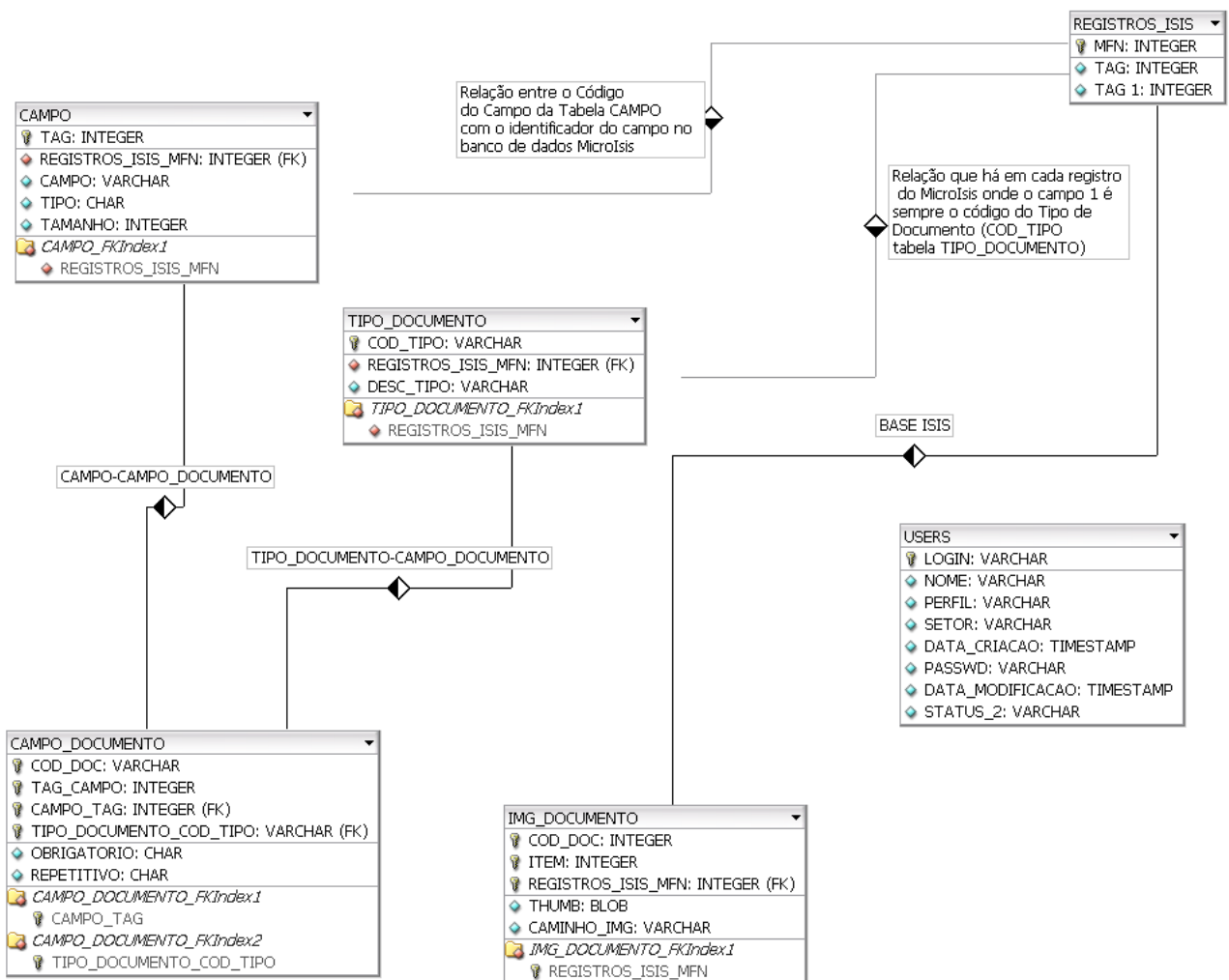
8.3.41 Diagrama de Colaboração Pesquisa de Documentos (Termos de Pesquisa)



8.4 DIAGRAMA DE CLASSES

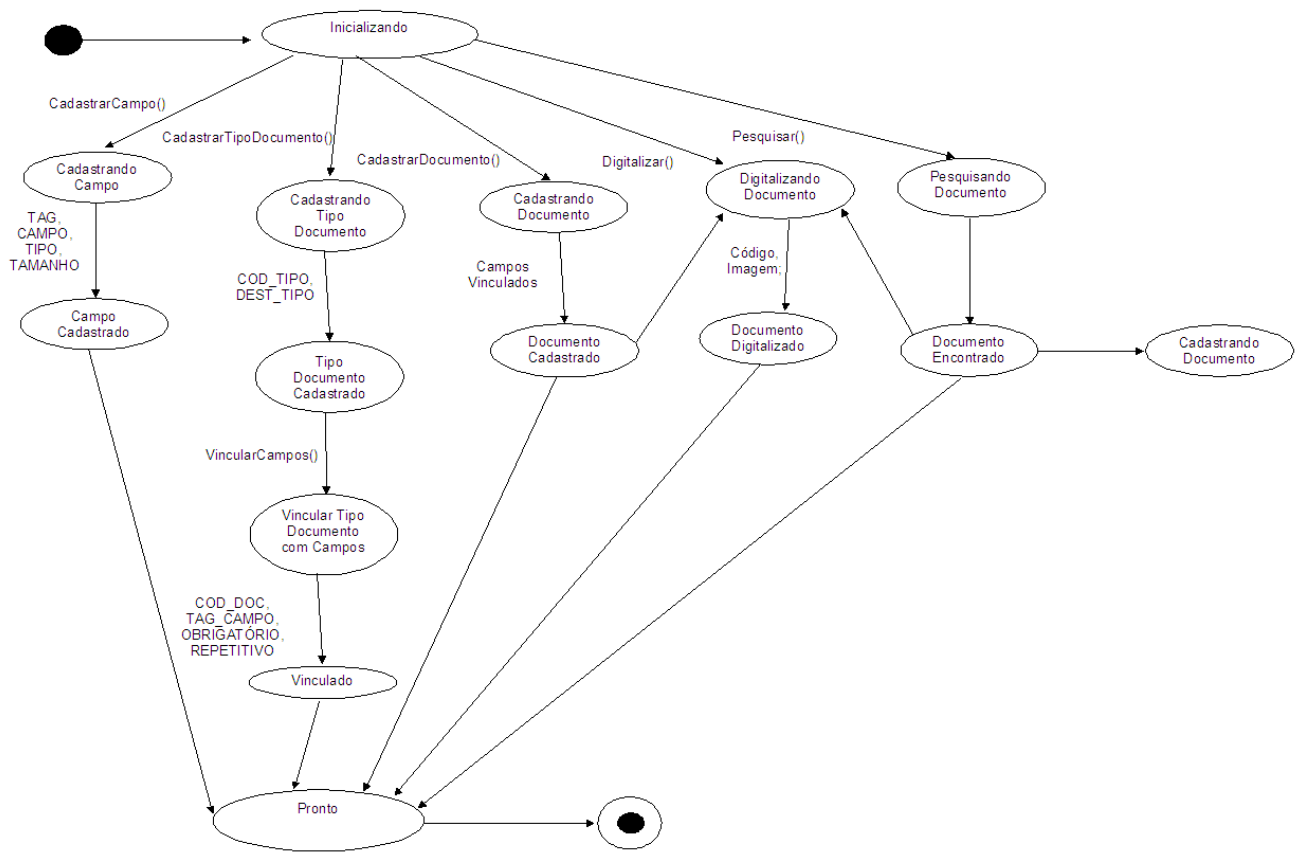


8.5 DIAGRAMA ENTIDADE RELACIONAMENTO



8.6 DIAGRAMA DE ESTADOS

Diagrama de Estado



8.7 DICIONÁRIO DE DADOS E CAMADA DE PERSISTÊNCIA

8.7.1 Tabela: Campo

Armazenamento de campos criados pelo usuário

TAG	Código do campo	integer	
CAMPO	Nome do Campo	varchar	20
TIPO	Tipo de informação aceita pelo campo (Numérico ou Alfanumérico)	char	8
TAMANHO	Número de caracteres aceitos pelo campo	integer	

8.7.2 Camada de Persistência TCampo

Classe : TCampo		
Atributo	Tabela	Campo
Ftag	CAMPO	TAG
Fnome	CAMPO	CAMPO
Ftamanho	CAMPO	TAMANHO
Ftipo	CAMPO	TIPO

Comando Inserção:

```
QryTemp.SQL.Text := 'INSERT INTO CAMPO (TAG,CAMPO,TAMANHO,TIPO)'+
' VALUES (:TAG,:CAMPO,:TAMANHO,:TIPO)';
```

```
QryTemp.ParamByName('Tag').Value := Ftag;
```

```
QryTemp.ParamByName('Campo').Value := Fnome;
```

```
QryTemp.ParamByName('Tamanho').Value := Ftamanho;
```

```
QryTemp.ParamByName('Tipo').Value := Ftipo;
```

8.7.3 Tabela: Tipo_Documento

Armazenamento dos tipos de documentos criados pelo usuário

COD_TIPO	Código do tipo de documento	varchar	4
DESC_TIPO	Descrição do tipo de documento	varchar	30

8.7.4 Camada de Persistência TTipoDoc

Classe : TTipoDoc		
Atributo	Tabela	Campo
Fcod	TIPO_DOCUMENTO	COD_TIPO
Fnome	TIPO_DOCUMENTO	DESC_TIPO

Comando Inserção:

```
QryTemp.SQL.Text := 'INSERT INTO TIPO_DOCUMENTO (COD_TIPO,'+
                    ' DESC_TIPO)'+
                    ' VALUES (:Cod,:Descricao)';
QryTemp.ParamByName('Cod').Value := Fcod;
QryTemp.ParamByName('Descricao').Value := Fnome;
```

8.7.5 Tabela: Campo_Documento

Armazenamento do vínculo dos tipos de documentos com os campos criados

COD_DOC	Código do Tipo do Documento armazenado na tabela Tipo_Documento	varchar	4
TAG_CAMPO	Código do Campo armazenado na tabela Campo	integer	
OBRIGATORIO	Obrigatoriedade ou não do campo para o tipo de documento	char	3

COD_DOC	Código do Tipo do Documento armazenado na tabela Tipo_Documento	varchar	4
REPETITIVO	Repetitividade ou não do campo para o tipo de documento	char	3

8.7.6 Camada de Persistência TDocumento

Classe : TDocumento		
Atributo	Tabela	Campo
FtipoDoc	TIPO_DOCUMENTO	COD_TIPO
Fcampo	CAMPO	TAG

Comando Inserção:

```

QryTemp.SQL.Text := 'INSERT INTO CAMPO_DOCUMENTO'+
                    ' (COD_DOC, TAG_CAMPO, OBRIGATORIO,
                    REPETITIVO)'+
                    ' VALUES (:Codigo, :Tag, :Obrig, :Repet)';
QryTemp.ParamByName('Codigo').Value := FtipoDoc.Fcod;
QryTemp.ParamByName('Tag').Value := Fcampo.Ftag;
QryTemp.ParamByName('Obrig').Value := 'Não';
QryTemp.ParamByName('Repet').Value := 'Não';

```

8.7.7 Tabela : Banco de Dados Microsis

Armazenamento dos dados dos documentos digitalizados. Cada registro é formado sempre pelo Tag número 1, onde fica armazenada o Código do Tipo de Documento, originado do campo **Cod_Tipo** da tabela **Tipo_Documento**, e pelos tags dos campos vinculados ao tipo de Documento (tabela **Campo_Documento**), originados do campo **Tag** da tabela **Campo**.

8.7.8 Camada de Persistência TDocumento

Classe : TConteudoDoc		
Atributo	Tabela	Campo
Fdoc	TIPO_DOCUMENT	COD_TIPO

	O	
FMfn	MICROISIS	Corresponde ao código seqüência de registros.
Fregistro	MICROISIS	Corresponde aos dados do documento.

Comando Inserção:

Fregistro := 'a1!NFET!a2!Fulano de Tal!a3!Ciclano da Silva!';

FMfn := IsisRecNewLock(DMFireBird.h,0);

Fr := IsisRecFieldUpdate(DMFireBird.h,0,Pchar(Fregistro));

Fr := IsisRecWrite(DMFireBird.h,0);

8.7.9 Tabela: Img_Documento

Armazenamento dos thumbnails e caminhos das imagens digitalizadas.

COD_DOC	Código do documento a que as imagens estão relacionadas	Integer	
ITEM	Sequência da imagem para determinado documento	integer	
THUMB	Thumbnail da imagem digitalizada	Blob (subtype: binary)	80
CAMINHO_IMG	Caminho da imagem digitalizada em seu tamanho real	varchar	255

8.7.10 Camada de Persistência TImagem

Classe : TImagem		
Atributo	Tabela	Campo
Fdoc	MICROISIS	Número do Registro
Fitem	IMG_DOCUMENTO	ITEM
Fthumb	IMG_DOCUMENTO	THUMB

Fdir	IMG_DOCUMENTO	CAMINHO_IMG
------	---------------	-------------

Comando Inserção:

```

QryTemp.SQL.Text := 'INSERT INTO IMG_DOCUMENTO'+
                    ' (COD_DOC, ITEM, CAMINHO_IMG, THUMB)+'
                    ' VALUES (:Cod, :Item, :Caminho, :Imagem)';

QryTemp.ParamByName('Cod').Value := FDoc.FMfn;
QryTemp.ParamByName('Item').Value := Fitem;
QryTemp.ParamByName('Caminho').Value := Fdir;
Fthumb.Picture.Graphic.SaveToStream(ImgBlob);
QryTemp.ParamByName('Imagem').LoadFromStream(ImgBlob,ftBlob);

```

8.7.11 Tabela: Users

Armazenamento dos usuários do sistema e seus acessos

LOGIN	Login do usuário para acesso ao sistema	varchar	10
NOME	Nome do usuário	varchar	30
PERFIL	Perfil de acesso às funcionalidades do sistema	varchar	1
SETOR	Setor de trabalho do usuário	varchar	30
DATA_CRIACAO	Data de criação do cadastro do usuário no sistema	timestamp	
PASSWD	Senha do usuário para acesso ao sistema	varchar	15
DATA_MODIFICACAO	Data da última modificação do	timestamp	

LOGIN	Login do usuário para acesso ao sistema	varchar	10
	cadastro do usuário		
STATUS	Status do usuário para acesso ao sistema (Ativo ou Inativo)	varchar	1

8.7.12 Camada de Persistência TUsuario

Classe : TUsuario		
Atributo	Tabela	Campo
FLogin	USER	LOGIN
Fnome	USER	NOME
Fperfil	USER	PERFIL
Fsetor	USER	SETOR
Fsenha	USER	PASSWD
Fstatus	USER	STATUS

**Comando
Inserção:**

```
QryTemp.SQL.Text := 'INSERT INTO USERS (LOGIN,NOME,PERFIL,SETOR,'+
    ' DATA_CRIACAO,PASSWD,STATUS)+'
    ' VALUES (:LOGIN,:NOME,:PERFIL,:SETOR,:DTCRIACAO,'+
    ' :PASSWD,:STATUS)';
```

```
QryTemp.ParamByName('Login').Value := Flogin;
QryTemp.ParamByName('Nome').Value := Fnome;
QryTemp.ParamByName('Perfil').Value := Fperfil;
QryTemp.ParamByName('Setor').Value := Fsetor;
QryTemp.ParamByName('DtCriacao').Value := Now;
QryTemp.ParamByName('Passwd').Value := Fsenha;
QryTemp.ParamByName('Status').Value := Fstatus;
```

8.8 SCRIPT DE CRIAÇÃO DO BANCO

```

/*****
Generated by IBExpert
*****/

```

```
SET SQL DIALECT 3;
```

```
SET NAMES NONE;
```

```
CREATE DATABASE 'C:\DigiDoc\Dados\DIGIDOC.FDB'
```

```
USER 'SYSDBA' PASSWORD 'masterkey'
```

```
PAGE_SIZE 16384
```

```
DEFAULT CHARACTER SET NONE;
```

```

/*****
Tables
*****/

```

```

CREATE TABLE CAMPO (
    TAG    INTEGER NOT NULL,
    CAMPO  VARCHAR(20) NOT NULL,
    TIPO   CHAR(8) NOT NULL,
    TAMANHO INTEGER NOT NULL
);

```

```
CREATE TABLE CAMPO_DOCUMENTO (
```

```
COD_DOC    VARCHAR(4) NOT NULL,  
TAG_CAMPO  INTEGER NOT NULL,  
OBRIGATORIO CHAR(3),  
REPETITIVO CHAR(3)  
);
```

```
CREATE TABLE IMG_DOCUMENTO (  
    COD_DOC    INTEGER NOT NULL,  
    ITEM       INTEGER NOT NULL,  
    THUMB      BLOB SUB_TYPE 0 SEGMENT SIZE 80,  
    CAMINHO_IMG VARCHAR(255)  
);
```

```
CREATE TABLE TIPO_DOCUMENTO (  
    COD_TIPO  VARCHAR(4) NOT NULL,  
    DESC_TIPO VARCHAR(30) NOT NULL  
);
```

```
CREATE TABLE USERS (  
    NOME          VARCHAR(55) NOT NULL,  
    DATA_ADMISSAO  TIMESTAMP NOT NULL,  
    SETOR         VARCHAR(30),  
    DATA_CRIACAO   TIMESTAMP NOT NULL,  
    PASSWD         VARCHAR(15) NOT NULL,  
    DATA_MODIFICACAO  TIMESTAMP,
```

```

STATUS          VARCHAR(15),
CAD_CAMPO        INTEGER NOT NULL,
CAD_DOCUMENTOS   INTEGER NOT NULL,
DIGITALIZAR      INTEGER NOT NULL,
PESQUISAR        INTEGER NOT NULL,
CONFIGURACOES    INTEGER NOT NULL,
ADM_USUARIOS     INTEGER NOT NULL,
VINCULAR_CAMPOS  INTEGER NOT NULL,
CAD_TIPO_DOCUMENTO INTEGER NOT NULL,
LOGIN            VARCHAR(10) NOT NULL,
COD_USER         INTEGER NOT NULL
);

```

```

/*****

```

```

/****          Primary Keys          ****/

```

```

/*****

```

```

ALTER TABLE CAMPO ADD CONSTRAINT PK_CAMPO PRIMARY KEY (TAG);

```

```

ALTER TABLE CAMPO_DOCUMENTO ADD CONSTRAINT PK_CAMPO_DOCUMENTO
PRIMARY KEY (COD_DOC, TAG_CAMPO);

```

```

ALTER TABLE IMG_DOCUMENTO ADD CONSTRAINT PK_IMG_DOCUMENTO
PRIMARY KEY (COD_DOC, ITEM);

```

```

ALTER TABLE TIPO_DOCUMENTO ADD CONSTRAINT PK_TIPO_DOCUMENTO
PRIMARY KEY (COD_TIPO);

```

REFERÊNCIA

<http://www.gruposipsis.com.br/cdsisis/cdsisis.asp>

www.unesco.org/webworld/isis/isis.htm

http://www.ced.ufsc.br/~ursula/5351/isisaula_1.html

<http://bibliolivre.blogspot.com/>)

http://www.ced.ufsc.br/~ursula/5351/isisaula_1.html)

<http://www.ibict.br/biblioteca/microisis.htm>)

<http://delphitwain.sourceforge.net/>

GED DE A A Z, CENADEM 2004

DELPHI 7 A BIBLIA, CANTÚ, MARCO 2003

CÓDIGO FONTE

```

program DigiDoc;

{%File 'DigiDoc.~dpr'}
{%File 'UPrincipal.~ddp'}

uses
  Forms,
  ULogin in 'ULogin.pas' {FrmLogin},
  UPrincipal in 'UPrincipal.pas' {FrmPrincipal},
  USobre in 'USobre.pas' {FrmSobre},
  UCadCampos in 'UCadCampos.pas' {FrmCadCampos},
  UDigitaliza in 'UDigitaliza.pas' {FrmDigitaliza},
  UPesquisaDoc in 'UPesquisaDoc.pas' {FrmPesquisaDoc},
  UUsuarios in 'UUsuarios.pas' {FrmUsuarios},
  UDMFireBird in 'UDMFireBird.pas' {DMFireBird: TDataModule},
  UTipoDoc in 'UTipoDoc.pas' {FrmTipoDoc},
  UCadDoc in 'UCadDoc.pas' {FrmCadDocumentos},
  UPesquisaCampo in 'UPesquisaCampo.pas' {FrmPesquisaCampos},
  UVincCampos in 'UVincCampos.pas' {FrmVincularCampos},
  UConfiguracao in 'UConfiguracao.pas' {FrmConfig},
  DelphiTwain in 'DelphiTwain.pas',
  DelphiTwainUtils in 'DelphiTwainUtils.pas',
  Twain in 'Twain.pas',
  UImagemDoc in 'UImagemDoc.pas' {FrmImagensDoc},
  UClasse in 'UClasse.pas';

begin
  Application.Initialize;
  Application.Title := 'DigiDoc';
  Application.CreateForm(TFrmPrincipal, FrmPrincipal);
  Application.CreateForm(TDMFireBird, DMFireBird);
  Application.Run;
end.

```

```
unit UCadCampos;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, Grids, DBGrids, StdCtrls, Buttons, ExtCtrls, DB, DBTables, DBXpress,  
Spin, Menus, Isis001, Isis32, UClasse;
```

```
type
```

```
TFrmCadCampos = class(TForm)  
    Panel1: TPanel;  
    BtnNovo: TSpeedButton;  
    BtnSalvar: TSpeedButton;  
    BtnCancelar: TSpeedButton;  
    BtnSair: TSpeedButton;  
    BtnEditar: TSpeedButton;  
    BtnExcluir: TSpeedButton;  
    Panel2: TPanel;  
    DbCampos: TDBGrid;  
    PopupMenu1: TPopupMenu;  
    AtualizarGrid1: TMenuItem;  
    LbTamanho: TLabel;  
    EdNome: TEdit;  
    RgTipo: TRadioGroup;  
    BtnPesquisar: TBitBtn;  
    LbNome: TLabel;  
    EdCodigo: TEdit;  
    EdTamMax: TComboBox;  
    procedure FormClose(Sender: TObject; var Action: TCloseAction);  
    procedure BtnNovoClick(Sender: TObject);  
    procedure BtnSalvarClick(Sender: TObject);  
    procedure BtnSairClick(Sender: TObject);  
    procedure BtnCancelarClick(Sender: TObject);  
    procedure FormCreate(Sender: TObject);  
    procedure AtualizarGrid1Click(Sender: TObject);  
    procedure BtnEditarClick(Sender: TObject);  
    procedure BtnPesquisarClick(Sender: TObject);  
    procedure BtnExcluirClick(Sender: TObject);  
    procedure DbCamposDblClick(Sender: TObject);
```

```
private
```

```
    procedure HabilitaCampos;  
    procedure LimpaCampos;  
    function Consistencias : Boolean;
```

```

    procedure AtualizaGrid;

    public
    { Public declarations }
    end;

var
    FrmCadCampos: TFrmCadCampos;
    QryTemp : TQuery;
    Campo : TCampo;

implementation

uses UDMFireBird, Math, UPesquisaCampo;

{$R *.dfm}

procedure TFrmCadCampos.FormClose(Sender: TObject;
    var Action: TCloseAction);
begin
    FrmCadCampos := nil;
    Action := CaFree;
end;

procedure TFrmCadCampos.HabilitaCampos;
var Status: boolean;
begin
    if EdNome.Enabled = true then
        Status := false
    else
        Status := true;

    if Status = true then begin
        BtnPesquisar.Enabled := False;
        EdNome.Enabled := true;
        EdTamMax.Enabled := true;
        RgTipo.Enabled := true;
        EdNome.Color := clWindow;
        EdTamMax.Color := clWindow;

        EdNome.SetFocus;
        BtnNovo.Enabled := false;
        BtnExcluir.Enabled := false;
        BtnEditar.Enabled := false;
        BtnCancelar.Enabled := true;
        BtnSalvar.Enabled := true;
        BtnSair.Enabled := false;
        DbCampos.Enabled := False;
    end;
end;

```

end;

```

if Status = false then begin
  BtnPesquisar.Enabled := True;
  EdNome.Enabled := false;
  EdTamMax.Enabled := false;
  RgTipo.Enabled := false;
  EdNome.Color := clScrollBar;
  EdTamMax.Color := clScrollBar;
  BtnNovo.Enabled := true;
  BtnExcluir.Enabled := true;
  BtnEditar.Enabled := true;
  BtnCancelar.Enabled := false;
  BtnSalvar.Enabled := false;
  BtnSair.Enabled := true;
  DbCampos.Enabled := True;
  DbCampos.Refresh;
end;

```

end;

```

procedure TFrmCadCampos.LimpaCampos;
begin
  EdNome.Text := '';
  EdTamMax.ItemIndex := -1;
  RgTipo.ItemIndex := -1;
  EdCodigo.Text := '';
end;

```

```

procedure TFrmCadCampos.BtnNovoClick(Sender: TObject);
begin
  DbCampos.SelectedIndex := -1;
  HabilitaCampos;
  LimpaCampos;
  BtnSalvar.Tag := 0;
  EdTamMax.Text := '';
  EdTamMax.Style := csDropDownList;
end;

```

```

procedure TFrmCadCampos.BtnSalvarClick(Sender: TObject);
begin
  if Consistencias then begin
    Campo := TCampo.Create;
    Campo.NewInstance;
    Campo.Fnome := EdNome.Text;
    Campo.Ftamanho := EdTamMax.ItemIndex + 1;
    if RgTipo.ItemIndex = 0 then begin
      Campo.Ftipo := 'Numérico';
    end else begin

```

```

    Campo.Ftipo := 'Alfanum.';
end;
if BtnSalvar.Tag = 0 then begin
    if DMFireBird.MensagemInclusao then begin
        if Campo.IncluiCampo then begin
            DMFireBird.ConfirmacaoInclusao;
            BtnCancelar.Click;
            BtnNovo.Click;
        end else begin
            DMFireBird.MensagemErroBanco;
        end;
    end;
end else begin
    if DMFireBird.MensagemAlteracao then begin
        Campo.Ftag := StrToInt(EdCodigo.Text);
        if Campo.AlterarCampo then begin
            DMFireBird.ConfirmacaoAlteracao;
            BtnCancelar.Click;
        end else begin
            DMFireBird.MensagemErroBanco;
        end;
    end;
end;
Campo.Destroy;
AtualizaGrid;
end;
end;

procedure TFrmCadCampos.BtnSairClick(Sender: TObject);
begin
    Close;
end;

procedure TFrmCadCampos.BtnCancelarClick(Sender: TObject);
begin
    HabilitaCampos;
    LimpaCampos;
    EdTamMax.Style := csDropDown;
end;

function TFrmCadCampos.Consistencias : Boolean;
begin
    Result := True;
    if EdNome.Text = '' then begin
        MessageDlg('O nome do campo não pode ser salvo em branco!', mtWarning, [mbOk], 0);
        Result := False;
    end;
end;

```

```

    if (EdTamMax.Text = "") or (EdTamMax.Text = '0') then begin
        MessageDlg('O tamanho do campo não pode ser salvo em branco ou com valor
zerado!', mtWarning, [mbOk], 0);
        Result := False;
    end;

    if (RgTipo.ItemIndex < 0) or (RgTipo.ItemIndex > 1) then begin
        MessageDlg('Deve-se escolher o tipo do campo!', mtWarning, [mbOk], 0);
        Result := False;
    end;
end;

procedure TFrmCadCampos.FormCreate(Sender: TObject);
var
    x : Integer;
begin
    AtualizaGrid;
    for x := 1 to 255 do begin
        EdTamMax.Items.Add(IntToStr(x));
    end;
end;

procedure TFrmCadCampos.AtualizaGrid;
begin
    DMFireBird.QryCampo.Close;
    DMFireBird.QryCampo.SQL.Text := 'SELECT * FROM CAMPO ORDER BY TAG';
    DMFireBird.QryCampo.Open;
end;

procedure TFrmCadCampos.AtualizarGrid1Click(Sender: TObject);
begin
    AtualizaGrid;
end;

procedure TFrmCadCampos.BtnEditarClick(Sender: TObject);
var
    Tamanho : Integer;
begin
    Tamanho := 0;
    if EdCodigo.Text <> '' then begin
        if EdNome.Text <> '' then begin
            BtnSalvar.Tag := 1;
            Tamanho := StrToInt(EdTamMax.Text) - 1;
            HabilitaCampos;
            EdTamMax.Style := csDropDownList;
            EdTamMax.ItemIndex := Tamanho;
        end;
    end;
end;

```

```

end else begin
    MessageDlg('Não há nenhum registro selecionado para
edição.',mtInformation,[mbOK],0);
end;
end;

procedure TFrmCadCampos.BtnPesquisarClick(Sender: TObject);
begin
    if Assigned(FrmPesquisaCampos) then begin
        FrmPesquisaCampos.Show;
        FrmPesquisaCampos.BringToFront;
        FrmPesquisaCampos.WindowState := wsNormal;
    end
    else begin
        FrmPesquisaCampos := TFrmPesquisaCampos.Create(Self);
        FrmPesquisaCampos.Show;
    end;
    Self.Enabled := False;
end;

procedure TFrmCadCampos.BtnExcluirClick(Sender: TObject);
begin
    if EdCodigo.Text <> " then begin
        Campo := TCampo.Create;
        Campo.NewInstance;
        Campo.Ftag := StrToInt(EdCodigo.Text);
        if Campo.Campolsis then begin
            if Campo.CampoVinculado then begin
                if DMFireBird.MensagemExclusao then begin
                    if Campo.ExcluiCampo then begin
                        DMFireBird.ConfirmacaoExclusao;
                    end else begin
                        DMFireBird.MensagemErroBanco;
                    end;
                end;
            end;
            Campo.Destroy;
            AtualizaGrid;
        end else begin
            MessageDlg('Impossível excluir, campo possui vínculo com algum tipo de
documento.',mtError,[mbOK],0);
        end;
    end else begin
        MessageDlg('Impossível excluir, campo possui dados
cadastrados.',mtError,[mbOK],0);
    end;
end else begin
    MessageDlg('Não há nenhum registro selecionado para
exclusão.',mtInformation,[mbOK],0);
end;

```

```
    end;  
end;  
  
procedure TFrmCadCampos.DbCamposDbClick(Sender: TObject);  
begin  
    EdCodigo.Text := DMFireBird.QryCampo.FieldByName('Tag').Value;  
    EdNome.Text := DMFireBird.QryCampo.FieldByName('Campo').Value;  
    EdTamMax.Text := DMFireBird.QryCampo.FieldByName('TAMANHO').Value;  
    if DMFireBird.QryCampo.FieldByName('TIPO').Value = 'Numérico' then  
        FrmCadCampos.RgTipo.ItemIndex := 0  
    else  
        FrmCadCampos.RgTipo.ItemIndex := 1;  
    end;  
  
end.
```

```
unit UCadDoc;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, StdCtrls, DBCtrls, Buttons, ExtCtrls, DBTables, Grids, Isis001,  
Isis32, UClasse;
```

```
type
```

```
TFrmCadDocumentos = class(TForm)
```

```
Panel2: TPanel;
```

```
CbTipoDoc: TComboBox;
```

```
Label1: TLabel;
```

```
Panel3: TPanel;
```

```
BtnNovo: TSpeedButton;
```

```
BtnEditar: TSpeedButton;
```

```
BtnSalvar: TSpeedButton;
```

```
BtnExcluir: TSpeedButton;
```

```
BtnCancelar: TSpeedButton;
```

```
BtnSair: TSpeedButton;
```

```
Panel1: TPanel;
```

```
SBoxCampo: TScrollBar;
```

```
RgOrdenar: TRadioGroup;
```

```
Label2: TLabel;
```

```
EdtCodigo: TEdit;
```

```
BtnAplicar: TSpeedButton;
```

```
BtnPesqDocumentos: TBitBtn;
```

```
procedure FormClose(Sender: TObject; var Action: TCloseAction);
```

```
procedure BtnSairClick(Sender: TObject);
```

```
procedure FormCreate(Sender: TObject);
```

```
procedure CbTipoDocChange(Sender: TObject);
```

```
procedure BtnNovoClick(Sender: TObject);
```

```
procedure BtnCancelarClick(Sender: TObject);
```

```
procedure BtnSalvarClick(Sender: TObject);
```

```
procedure RgOrdenarClick(Sender: TObject);
```

```
procedure BtnAplicarClick(Sender: TObject);
```

```
procedure EdtCodigoKeyPress(Sender: TObject; var Key: Char);
```

```
procedure BtnEditarClick(Sender: TObject);
```

```
procedure BtnExcluirClick(Sender: TObject);
```

```
procedure BtnPesqDocumentosClick(Sender: TObject);
```

```
private
```

```
procedure Ativar_Campos;
```

```
procedure Desativar_Campos;
```

```
procedure ClicarBotao(Sender: TObject);
```

```
procedure ClicarLinhaGrid(Sender : TObject);
```

```
procedure Hab_Desab_Botoes(var Item : Smallint);
```

```

procedure Esc_TipoDoc;
function MontarRegistro : PChar;
function CodigoDocumento : String;
function VerificarObrig : Boolean;
procedure LimparCampos;
procedure LimparEditRepet;
procedure CarregarComboTipoDoc;
procedure Formar_Campos(Var Cod : String);
procedure LimparFormulario;
procedure MontReg(var Codigo : Longint);
procedure VerificarFormulario;
procedure VerificarCampoNumerico(Sender : TObject; var Key: Char);
public
  { Public declarations }
end;

```

```

var
  FrmCadDocumentos: TFrmCadDocumentos;
  ConteudoDoc : TConteudoDoc;
  TipoDoc : TTipoDoc;
  Imagem : TImagem;
  Verificacao, Posicao : integer;
  Cod : String;
  Item : Smallint;

```

implementation

```

Uses UDMFireBird, UPesquisaDoc, UImagemDoc;
{$R *.dfm}

```

```

procedure TFrmCadDocumentos.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  if Assigned(FrmPesquisaDoc) then begin
    FrmPesquisaDoc.Enabled := True;
  end;
  FrmCadDocumentos := nil;
  Action := caFree;
end;

```

```

procedure TFrmCadDocumentos.BtnSairClick(Sender: TObject);
begin
  Close;
end;

```

```

procedure TFrmCadDocumentos.FormCreate(Sender: TObject);
var
  I : integer;

```

```

begin
  CarregarComboTipoDoc;
  if Assigned(FrmPesquisaDoc) then begin
    l := FrmPesquisaDoc.Linha;
    EdtCodigo.Text := FrmPesquisaDoc.SgDados.Cells[0,l];
  end;
end;

procedure TFrmCadDocumentos.CbTipoDocChange(Sender: TObject);
var X      : integer;
    Edit   : TEdit;
    lbl    : TLabel;
    toplble : integer;
    toplbld : integer;
    topedte : integer;
    topedtd : integer;
    coluna : boolean;
    posicao : integer;
    StrGrid : TStringGrid;
    botao : TButton;

begin
  LimparFormulario;
  Esc_TipoDoc;
  toplble := 3;
  toplbld := 3;
  topedte := 20;
  topedtd := 20;
  coluna := false;

  if CbTipoDoc.ItemIndex <> 0 then begin
    EdtCodigo.Text := '';
    EdtCodigo.Enabled := False;
    BtnAplicar.Enabled := False;
    Cod := CodigoDocumento;
    Formar_Campos(Cod);
    BtnPesqDocumentos.Enabled := False;
  end else begin
    if EdtCodigo.Text <> '' then begin
      Formar_Campos(Cod);
      BtnPesqDocumentos.Enabled := True;
    end else begin
      EdtCodigo.Text := '';
      EdtCodigo.Enabled := True;
      BtnAplicar.Enabled := True;
      BtnPesqDocumentos.Enabled := True;
    end;
  end;
end;

```

end;

procedure TFrmCadDocumentos.Ativar_Campos;

var

 x : Integer;

begin

 For X := (FrmCadDocumentos.ComponentCount - 1) downto 0 do begin

 if (Components[X] is TEdit) and (TEdit(Components[X]).Tag <> 0) then begin

 TEdit(Components[X]).Enabled := true;

 end else begin

 if Components[X] is TStringGrid then begin

 TStringGrid(Components[X]).Enabled := true;

 end

 else begin

 if Components[X] is TButton and (Components[X].Tag <> 0) then

 TButton(Components[X]).Enabled := true;

 end;

 end;

 end;

end;

procedure TFrmCadDocumentos.BtnNovoClick(Sender: TObject);

begin

 BtnSalvar.Tag := 0;

 EdtCodigo.Enabled := False;

 BtnAplicar.Enabled := False;

 Ativar_Campos;

 Item := 1;

 Hab_Desab_Botoes(Item);

 BtnPesqDocumentos.Enabled := False;

end;

procedure TFrmCadDocumentos.ClicarBotao(Sender: TObject);

var

 x, y, z, QtdeLinhas : Integer;

 Marca : Integer;

 Texto : String;

begin

 For x := 1 to (FrmCadDocumentos.ComponentCount - 1) do begin

 if (Components[x] is TButton) then begin

 If TButton(Components[x]).Focused then begin

 Marca := TButton(Components[x]).Tag;

 For y := 1 to (FrmCadDocumentos.ComponentCount - 1) do begin

 If (Components[y] is TEdit) then begin

 If TEdit(Components[y]).Tag = Marca then begin

 Texto := TEdit(Components[y]).Text;

 TEdit(Components[y]).Text := "";

 Break;

```

    end;
  end;
end;
if Texto <> " then begin
  For Z := 1 to (FrmCadDocumentos.ComponentCount - 1) do begin
    If Components[z] is TStringGrid then begin
      If (TStringGrid(Components[z]).Tag = Marca) then begin
        QtdeLinhas := TStringGrid(Components[z]).RowCount;
        TStringGrid(Components[z]).Cells[0,QtdeLinhas] := Texto;
        TStringGrid(Components[z]).RowCount :=
TStringGrid(Components[z]).RowCount + 1;
        Break;
      end;
    end;
  end;
end;
end;
end;
end;
end;
end;
end;
end;
end;

```

```

procedure TFrmCadDocumentos.ClicarLinhaGrid(Sender : TObject);

```

```

var

```

```

  Marca, x, Linha : Integer;

```

```

  Texto, TextoStr : String;

```

```

  Tamanho, Ant, Prox, t : Integer;

```

```

begin

```

```

  for x := 1 to (FrmCadDocumentos.ComponentCount - 1) do begin

```

```

    if Components[x] is TStringGrid then begin

```

```

      If TStringGrid(Components[x]).Focused then begin

```

```

        Marca := TStringGrid(Components[x]).Tag;

```

```

        Linha := TStringGrid(Components[x]).Row;

```

```

        if Linha = 0 then begin

```

```

          TStringGrid(Components[x]).RowCount := TStringGrid(Components[x]).RowCount -
(TStringGrid(Components[x]).RowCount - 1);

```

```

        end else begin

```

```

          Texto := TStringGrid(Components[x]).Rows[Linha].Text;

```

```

          Tamanho := Length(Texto);

```

```

          Texto := Copy(Texto,1,Tamanho-2);

```

```

          for t := Linha to (TStringGrid(Components[x]).RowCount - 1) do begin

```

```

            Prox := t + 1;

```

```

            TextoStr := TStringGrid(Components[x]).Rows[Prox].Text;

```

```

            Tamanho := Length(TextoStr);

```

```

            TextoStr := Copy(TextoStr,1,Tamanho-2);

```

```

            TStringGrid(Components[x]).Cells[0,t] := TextoStr;

```

```

          end;

```

```

          TStringGrid(Components[x]).RowCount := TStringGrid(Components[x]).RowCount -

```

```

1;

```

```

        end;
        Break;
    end;
end;
end;
for x := 1 to (FrmCadDocumentos.ComponentCount - 1) do begin
    if Components[x] is TEdit then begin
        if TEdit(Components[x]).Tag = Marca then begin
            TEdit(Components[x]).Text := Texto;
        end;
    end;
end;
end;
end;

//Procedure para habilitar ou desabilitar os botões dependendo da ação
procedure TFrmCadDocumentos.Hab_Desab_Botoes(var Item : Smallint);
begin
    if (Item = 1) or (Item = 2) then begin
        EdtCodigo.Enabled := False;
        CbTipoDoc.Enabled := False;
        RgOrdenar.Enabled := False;
        BtnNovo.Enabled := False;
        BtnEditar.Enabled := False;
        BtnSalvar.Enabled := True;
        BtnExcluir.Enabled := False;
        BtnCancelar.Enabled := True;
        BtnSair.Enabled := False;
    end else begin
        EdtCodigo.Enabled := True;
        CbTipoDoc.Enabled := True;
        RgOrdenar.Enabled := True;
        BtnNovo.Enabled := True;
        BtnEditar.Enabled := False;
        BtnSalvar.Enabled := False;
        BtnExcluir.Enabled := False;
        BtnCancelar.Enabled := False;
        BtnSair.Enabled := True;
    end;
end;

procedure TFrmCadDocumentos.BtnCancelarClick(Sender: TObject);
begin
    Item := 3;
    Hab_Desab_Botoes(Item);
    LimparCampos;
    Desativar_Campos;
    CbTipoDocChange(Self);
    VerificarFormulario;
end;

```

```

    BtnPesqDocumentos.Enabled := True;
end;

```

```

procedure TFrmCadDocumentos.Esc_TipoDoc;
begin
    if (CbTipoDoc.ItemIndex = 0) and (EdtCodigo.Text = "") then begin
        BtnNovo.Enabled := False;
        BtnEditar.Enabled := False;
        BtnExcluir.Enabled := False;
    end else begin
        if EdtCodigo.Text = "" then begin
            BtnNovo.Enabled := True;
            BtnEditar.Enabled := False;
            BtnExcluir.Enabled := False;
        end else begin
            BtnEditar.Enabled := True;
            BtnNovo.Enabled := False;
            BtnExcluir.Enabled := True;
        end;
    end;
end;

```

```

procedure TFrmCadDocumentos.BtnSalvarClick(Sender: TObject);
begin
    if VerificarObrig then begin
        ConteudoDoc := TConteudoDoc.Create;
        ConteudoDoc.NewInstance;
        ConteudoDoc.Fregistro := MontarRegistro;
        if BtnSalvar.Tag = 0 then begin
            if DMFireBird.MensagemInclusao then begin
                if ConteudoDoc.IncluiConteudo then begin
                    MessageDlg('Registro ' + IntToStr(ConteudoDoc.FMfn) + ' incluído com
sucesso.', mtInformation, [mbOK], 0);
                    if MessageDlg('Deseja digitalizar o documento?', mtConfirmation, [mbYes, mbNo], 0) =
6 then begin
                        if Assigned (FrmlImagensDoc) then begin
                            FrmlImagensDoc.BringToFront;
                            FrmlImagensDoc.Show;
                            FrmlImagensDoc.WindowState := wsNormal;
                        end else begin
                            FrmlImagensDoc := TFrmImagensDoc.Create(Self);
                            FrmlImagensDoc.Show;
                        end;
                        FrmlImagensDoc.EdtCodigo.Text := IntToStr(ConteudoDoc.FMfn);
                        FrmlImagensDoc.BtnAplicar.Click;
                        BtnCancelar.Click;
                    end else begin
                        BtnCancelar.Click;

```

```

        BtnNovo.Click;
    end;
end else begin
    DMFireBird.MensagemErroBanco;
end;
end;
end else begin
    if DMFireBird.MensagemAlteracao then begin
        ConteudoDoc.FMfn := StrToInt(EdtCodigo.Text);
        if ConteudoDoc.AlterarConteudo then begin
            MessageDlg('Registro ' + IntToStr(ConteudoDoc.FMfn) + ' alterado com
sucesso.',mtInformation,[mbOK],0);
            BtnCancelar.Click;
        end else begin
            DMFireBird.MensagemErroBanco;
        end;
    end;
end;
end;
end else begin
    MessageDlg('Favor preencher todos os campos obrigatórios.',mtError,[mbOK],0);
end;
end;

function TFrmCadDocumentos.MontarRegistro : PChar;
var
    x, Tag, Linha, Tamanho : Integer;
    Registro, Texto : String;
begin
    Registro := '';
    Registro := 'a1!' + Cod + '!';
    for x := 1 to (FrmCadDocumentos.ComponentCount - 1) do begin
        if (Components[x] is TEdit) then begin
            if (TEdit(Components[x]).Text <> '') then begin
                if (TEdit(Components[x]).Tag <> 0) then begin
                    Registro := Registro + 'a' + IntToStr(TEdit(Components[x]).Tag) + '!' +
TEdit(Components[x]).Text + '!';
                end;
            end;
        end else begin
            if Components[x] is TStringGrid then begin
                if TStringGrid(Components[x]).RowCount > 1 then begin
                    for Linha := 1 to (TStringGrid(Components[x]).RowCount - 1) do begin
                        Texto := TStringGrid(Components[x]).Rows[Linha].Text;
                        Tamanho := Length(Texto);
                        Texto := Copy(Texto,1,Tamanho-2);
                        Registro := Registro + 'a' + IntToStr(TStringGrid(Components[x]).Tag) + '!' + Texto
+ '!';
                    end;
                end;
            end;
        end;
    end;
end;

```

```

        end;
    end;
end;
end;
Result := PChar(Registro);
end;

function TFrmCadDocumentos.VerificarObrig : Boolean;
var
    Tag : Integer;
    x : Integer;
begin
    ConteudoDoc := TConteudoDoc.Create;
    ConteudoDoc.NewInstance;
    ConteudoDoc.FDoc.FtipoDoc.Fcod := Cod;
    QryTemp := ConteudoDoc.FDoc.VerificarObrig;
    Verificacao := 0;
    if not QryTemp.Eof then begin
        QryTemp.First;
        while not QryTemp.Eof do begin
            Tag := QryTemp.FieldName('TAG_CAMPO').Value;
            for x := 1 to (FrmCadDocumentos.ComponentCount - 1) do begin
                if Components[x] is TEdit then begin
                    if TEdit(Components[x]).Tag = Tag then begin
                        if (TEdit(Components[x]).Text = "") and (QryTemp.FieldName('Repetitivo').Value
= 'Não') then begin
                            Verificacao := Verificacao + 1;
                        end;
                    end;
                end else begin
                    if Components[x] is TStringGrid then begin
                        if TStringGrid(Components[x]).Tag = Tag then begin
                            if TStringGrid(Components[x]).RowCount = 1 then begin
                                Verificacao := Verificacao + 1;
                            end;
                        end;
                    end;
                end;
            end;
        end;
        QryTemp.Next;
    end;
    if Verificacao = 0 then begin
        Result := True;
    end else begin
        Result := False;
    end;
end else begin
    Result := True;
end;

```

```

end;
QryTemp.Close;
FreeAndNil(QryTemp);
ConteudoDoc.Destroy;
end;

```

```

function TFrmCadDocumentos.CodigoDocumento : String;
var
  Posicao : Integer;
begin
  Posicao := Pos('-',CbTipoDoc.Text);
  Result := Copy(CbTipoDoc.Text,1,Posicao-2);
end;

```

```

procedure TFrmCadDocumentos.LimparCampos;
var
  x : Integer;
begin
  for x := 1 to FrmCadDocumentos.ComponentCount - 1 do begin
    if Components[x] is TEdit then begin
      TEdit(Components[x]).Text := "";
    end else begin
      if Components[x] is TStringGrid then begin
        TStringGrid(Components[x]).RowCount := 1;
      end;
    end;
  end;
end;

```

```

procedure TFrmCadDocumentos.LimparEditRepet;
var
  x, Tag : Integer;
begin
  ConteudoDoc := TConteudoDoc.Create;
  ConteudoDoc.NewInstance;
  QryTemp := ConteudoDoc.FDoc.VerificaRepet;
  if not QryTemp.Eof then begin
    QryTemp.First;
    while not QryTemp.Eof do begin
      Tag := QryTemp.FieldName('TAG_CAMPO').Value;
      for x := 1 to (FrmCadDocumentos.ComponentCount - 1) do begin
        if Components[x] is TEdit then begin
          if TEdit(Components[x]).Tag = Tag then begin
            TEdit(Components[x]).Text := "";
          end;
        end;
      end;
    end;
  end;
  QryTemp.Next;

```

```

    end;
end;
QryTemp.Close;
FreeAndNil(QryTemp);
ConteudoDoc.Destroy;
end;

```

```

procedure TFrmCadDocumentos.CarregarComboTipoDoc;
var
  x : Integer;
  Ord : Smallint;
begin
  for x := CbTipoDoc.Items.Count downto 0 do begin
    CbTipoDoc.Items.Delete(x);
  end;
  Ord := RgOrdenar.ItemIndex;
  TipoDoc := TTipoDoc.Create;
  TipoDoc.NewInstance;
  CbTipoDoc.Items.Add('Selecione...');
  QryTemp := TipoDoc.CarregaTipoDoc(Ord);
  if not QryTemp.Eof then begin
    QryTemp.First;
    while not QryTemp.Eof do begin
      CbTipoDoc.Items.Add(QryTemp.FieldName('COD_TIPO').Value + ' - ' +
        QryTemp.FieldName('DESC_TIPO').Value);
      QryTemp.Next;
    end;
    CbTipoDoc.Refresh;
    CbTipoDoc.ItemIndex := 0;
  end;
  QryTemp.Close;
  FreeAndNil(QryTemp);
  TipoDoc.Destroy;
end;

```

```

procedure TFrmCadDocumentos.RgOrdenarClick(Sender: TObject);
begin
  CarregarComboTipoDoc;
  CbTipoDocChange(Self);
end;

```

```

procedure TFrmCadDocumentos.Desativar_Campos;
var
  x : Integer;
begin
  For X := (FrmCadDocumentos.ComponentCount - 1) downto 0 do begin
    if (Components[X] is TEdit) then begin
      TEdit(Components[x]).Enabled := false;
    end;
  end;
end;

```

```

end else begin
  if Components[x] is TStringGrid then begin
    TStringGrid(Components[x]).Enabled := false;
  end
  else begin
    if Components[x] is TButton and (Components[x].Tag <> 0) then
      TButton(Components[x]).Enabled := false;
    end;
  end;
end;
end;
end;

procedure TFrmCadDocumentos.Formar_Campos(Var Cod : String);
var X      : integer;
    Edit   : TEdit;
    lbl    : TLabel;
    toplble : integer;
    toplbld : integer;
    topedte : integer;
    topedtd : integer;
    coluna : boolean;
    posicao : integer;
    StrGrid : TStringGrid;
    botao : TButton;

begin
  Esc_TipoDoc;
  toplble := 3;
  toplbld := 3;
  topedte := 20;
  topedtd := 20;
  coluna := false;

  ConteudoDoc := TConteudoDoc.Create;
  ConteudoDoc.NewInstance;
  ConteudoDoc.FDoc.FtipoDoc.Fcod := Cod;
  QryTemp := ConteudoDoc.FDoc.PegarCampos;
  if not QryTemp.Eof then begin
    while not QryTemp.Eof do begin
      lbl := TLabel.Create(self);
      lbl.Name := 'lbl' + IntToStr(QryTemp.FieldName('TAG').Value);
      if Coluna = False then begin
        lbl.Left := 30;
        lbl.Top := toplble + 10;
      end
      else begin
        lbl.Left := 300;
        lbl.Top := toplbld + 10;
      end
    end
  end
end

```

```

end;
lbl.Width := 200;
lbl.Tag := 0;
lbl.Font.Style := [fsBold];
lbl.AutoSize := True;
lbl.Caption := QryTemp.FieldName('Campo').Value;
if QryTemp.FieldName('OBRIGATORIO').Value = 'Sim' then begin
  lbl.Caption := lbl.Caption + ' (Obrig.)';
end;
lbl.Alignment := taLeftJustify;
lbl.Parent := FrmCadDocumentos.SBoxCampo;

Edit := TEdit.Create(self);
Edit.Name := 'edt' + IntToStr(QryTemp.FieldName('TAG').Value);
if Coluna = False then begin
  edit.Left := 30;
  edit.Top := topedte + 10;
  toplble := toplble + edit.Height + 23;
  topedte := topedte + Edit.Height + 23;
end
else begin
  edit.Left := 300;
  edit.Top := topedtd + 10;
  toplbld := toplbld + edit.Height + 23;
  topedtd := topedtd + Edit.Height + 23;
end;
Edit.Width := 200;
Edit.CharCase := ecUpperCase;
Edit.Tag := QryTemp.FieldName('TAG').Value;
Edit.Text := '';
Edit.MaxLength := QryTemp.FieldName('Tamanho').Value;
if QryTemp.FieldName('TIPO').Value = 'Numérico' then begin
  Edit.OnKeyPress := VerificarCampoNumerico;
end;
Edit.Enabled := False;
Edit.Parent := FrmCadDocumentos.SBoxCampo;

if QryTemp.FieldName('Repetitivo').Value = 'Sim' then begin
  botao := TButton.Create(Self);
  botao.Left := Edit.Left + Edit.Width + 5;
  botao.Height := 25;
  botao.Width := 37;
  botao.Caption := '+';
  botao.Tag := QryTemp.FieldName('Tag').Value;
  botao.Top := Edit.Top;
  botao.Enabled := False;
  botao.OnClick := ClicarBotao;
  botao.Parent := FrmCadDocumentos.SBoxCampo;

```

```

StrGrid := TStringGrid.Create(Self);
StrGrid.Name := 'strGrid' + IntToStr(QryTemp.FieldName('TAG').Value);
if coluna = False then begin
    StrGrid.Left := 30;
    toplble := lbl.Top + (StrGrid.Height - Edit.Height) + 10;
    topedte := Edit.Top + (StrGrid.Height - Edit.Height) + 10;
end
else begin
    StrGrid.Left := 300;
    toplble := lbl.Top + (StrGrid.Height - Edit.Height) + 10;
    topedtd := edit.Top + (StrGrid.Height - Edit.Height) + 10;
end;
StrGrid.Top := edit.Top + Edit.Height + 2;
StrGrid.Width := 200;
StrGrid.Tag := QryTemp.FieldName('TAG').Value;
StrGrid.DefaultRowHeight := 22;
StrGrid.DefaultColWidth := 175;
StrGrid.RowCount := 1;
StrGrid.ColCount := 0;
StrGrid.FixedRows := 0;
StrGrid.FixedCols := 0;
StrGrid.Height := 73;
StrGrid.Enabled := False;
StrGrid.OnDblClick := ClicarLinhaGrid;
StrGrid.Parent := FrmCadDocumentos.SBoxCampo;
StrGrid.Cells[0,1] := "";
end;
coluna := not Coluna;
QryTemp.Next;
end;
end
else begin
    showmessage('Não há campos vinculados para este tipo de documento');
    CbTipoDoc.ItemIndex := 0;
    CbTipoDocChange(Self);
end;
QryTemp.Close;
FreeAndNil(QryTemp);
ConteudoDoc.Destroy;
end;

procedure TFrmCadDocumentos.BtnAplicarClick(Sender: TObject);
var
    Area : PChar;
    Campo, Ocor : Smallint;
begin
    Area := StrAlloc(6);
    if EdtCodigo.Text <> " then begin

```

```

Campo := 1;
Ocor := 1;
ConteudoDoc := TConteudoDoc.Create;
ConteudoDoc.NewInstance;
ConteudoDoc.FMfn := StrToInt(EdtCodigo.Text);
Area := ConteudoDoc.PegarDadosIsis(Campo,Ocor);
if Area > '0000' then begin
    Cod := area;
    CbTipoDocChange(Self);
    MontReg(ConteudoDoc.FMfn);
    Esc_TipoDoc;
    EdtCodigo.Enabled := False;
    BtnCancelar.Enabled := True;
    BtnAplicar.Enabled := False;
    CbTipoDoc.Enabled := False;
    RgOrdenar.Enabled := False;
end else begin
    LimparFormulario;
    MessageDlg('Registro não encontrado.',mtError,[mbOK],0);
end;
end;
end;

```

```

procedure TFrmCadDocumentos.LimparFormulario;
var
    x : Integer;
begin
    for X := (FrmCadDocumentos.ComponentCount - 1) downto 0 do begin
        if (Components[X] is TEdit) and (Components[x].Tag <> 0) then begin
            Components[X].Free;
        end else begin
            if (Components[X] is TLabel) and (Components[X].Tag = 0) then begin
                Components[x].Free;
            end else begin
                if Components[x] is TStringGrid then begin
                    Components[x].Free;
                end else begin
                    if Components[x] is TButton and (Components[x].Tag <> 0) then begin
                        Components[x].Free;
                    end;
                end;
            end;
        end;
    end;
end;
end;
end;
end;

```

```

procedure TFrmCadDocumentos.MontReg(var Codigo : Longint);
var

```

```

    Campo, Cont, TotCampos, TotComp, x : Smallint;
    Area : PChar;
begin
    Area := StrAlloc(8000);
    x := 1;
    TotComp := 0;
    Cont := 0;
    Campo := 0;
    ConteudoDoc := TConteudoDoc.Create;
    ConteudoDoc.NewInstance;
    ConteudoDoc.FMfn := StrToInt(EdtCodigo.Text);
    ConteudoDoc.FDoc.FtipoDoc.Fcod := Cod;
    TotCampos := ConteudoDoc.NumCamposReg;
    while x <= TotCampos do begin
        if ConteudoDoc.FtagReg[x] <> 1 then begin
            if Campo <> ConteudoDoc.FtagReg[x] then begin
                Campo := ConteudoDoc.FtagReg[x];
                cont := 1;
            end else begin
                cont := cont + 1;
            end;
            Area := ConteudoDoc.PegarDadosIsis(Campo,Cont);
            ConteudoDoc.FDoc.Fcampo.Ftag := Campo;
            if ConteudoDoc.FDoc.VerificarCampoRepet then begin
                for TotComp := 1 to FrmCadDocumentos.ComponentCount -1 do begin
                    if Components[TotComp] is TStringGrid then begin
                        if TStringGrid(Components[TotComp]).Tag = Campo then begin
                            TStringGrid(Components[TotComp]).Cells[0,TStringGrid(Components[TotComp]).
RowCount] := area;
                            TStringGrid(Components[TotComp]).RowCount :=
TStringGrid(Components[TotComp]).RowCount + 1;
                        end;
                    end;
                end;
            end else begin
                for TotComp := 1 to FrmCadDocumentos.ComponentCount -1 do begin
                    if Components[TotComp] is TEdit then begin
                        if TEdit(Components[TotComp]).Tag = Campo then begin
                            TEdit(Components[TotComp]).Text := area;
                        end;
                    end;
                end;
            end;
            x := x + 1;
        end;
        ConteudoDoc.Destroy;
    end;
end;

```

```

procedure TFrmCadDocumentos.VerificarFormulario;
begin
  if Assigned(FrmPesquisaDoc) then begin
    BtnSair.Click;
  end;
end;

```

```

procedure TFrmCadDocumentos.EdtCodigoKeyPress(Sender: TObject;
  var Key: Char);
begin
  if not (Key in ['0'..'9',Chr(8)]) then begin
    Key:= #0;
  end;
end;

```

```

procedure TFrmCadDocumentos.VerificarCampoNumerico(Sender : TObject; var Key:
  Char);
begin
  if not (Key in ['0'..'9',Chr(8)]) then begin
    Key:= #0;
  end;
end;

```

```

procedure TFrmCadDocumentos.BtnEditarClick(Sender: TObject);
begin
  EdtCodigo.Enabled := False;
  BtnAplicar.Enabled := False;
  Ativar_Campos;
  Item := 2;
  Hab_Desab_Botoes(Item);
  BtnSalvar.Tag := 1;
  BtnPesqDocumentos.Enabled := False;
end;

```

```

procedure TFrmCadDocumentos.BtnExcluirClick(Sender: TObject);
begin
  ConteudoDoc := TConteudoDoc.Create;
  ConteudoDoc.NewInstance;
  ConteudoDoc.FMfn := StrToInt(EdtCodigo.Text);
  if ConteudoDoc.VerificaImagem then begin
    if MessageDlg('O documento possui imagens, confirma exclusão mesmo
    assim?',mtConfirmation,[mbYes,mbNo],0) = 6 then begin
      Imagem := TImagem.Create;
      Imagem.NewInstance;
      Imagem.Fdoc.FMfn := ConteudoDoc.FMfn;
      Imagem.PegaImagem.Open;
      Imagem.PegaImagem.First;
    end;
  end;
end;

```

```

While not Imagem.Pegalmagem.Eof do begin
  Imagem.Fitem := Imagem.Pegalmagem.FieldName('Item').Value;
  Imagem.ExcluiImagem;
  Imagem.Pegalmagem.Next;
end;
Imagem.Pegalmagem.Close;
Imagem.Destroy;
if ConteudoDoc.ExcluiConteudo then begin
  DMFireBird.ConfirmacaoExclusao;
  BtnCancelar.Click;
end else begin
  DMFireBird.MensagemErroBanco;
end;
end;
end else begin
  if DMFireBird.MensagemExclusao then begin
    if ConteudoDoc.ExcluiConteudo then begin
      DMFireBird.ConfirmacaoExclusao;
      BtnCancelar.Click;
    end else begin
      DMFireBird.MensagemErroBanco;
    end;
  end;
end;
end;
ConteudoDoc.Destroy;
end;

procedure TFrmCadDocumentos.BtnPesqDocumentosClick(Sender: TObject);
begin
  if Assigned(FrmPesquisaDoc) then begin
    FrmPesquisaDoc.Show;
    FrmPesquisaDoc.BringToFront;
    FrmPesquisaDoc.WindowState := wsNormal;
  end else begin
    FrmPesquisaDoc := TFrmPesquisaDoc.Create(Self);
    FrmPesquisaDoc.Show;
  end;
  FrmCadDocumentos.Enabled := False;
end;

end.

```

```
unit UClasse;
```

```
interface
```

```
uses
```

```
  Forms, DBTables, Isis32, Isis001, Dialogs, DelphiTwain, Twain, DelphiTwainUtils,  
  Graphics, QExtCtrls, Classes, Jpeg, clipbrd, DB, Spin, ExtCtrls;
```

```
//-----Classe TUsuario-----
```

```
type
```

```
  TUsuario = class(TObject)
```

```
    Flogin : string;
```

```
    FNome : string;
```

```
    Fperfil : String;
```

```
    Fsetor : String;
```

```
    Fsenha : String;
```

```
    Fstatus : String;
```

```
  private
```

```
  public
```

```
    Constructor Create;
```

```
    Destructor Destroy; Override;
```

```
    function IncluiUsuario : Boolean;
```

```
    function AlteraUsuario : Boolean;
```

```
    function ExcluiUsuario : Boolean;
```

```
    function CarregaPerfilUsuario : String;
```

```
    function VerifLogin : Boolean;
```

```
    function CarregaUsuario : Boolean;
```

```
    function AlteraSenha : Boolean;
```

```
end;
```

```
//-----Classe TCampo-----
```

```
type
```

```
  TCampo = class(TObject)
```

```
    Ftag : Integer;
```

```
    Fnome : String;
```

```
    Ftamanho : Byte;
```

```
    Ftipo : String;
```

```
  private
```

```
    procedure AtualTipoIndexCampo;
```

```
  public
```

```
    Constructor Create;
```

```
    Destructor Destroy; Override;
```

```
    function IncluiCampo : Boolean;
```

```
    function AlteraCampo : Boolean;
```

```
    function ExcluiCampo : Boolean;
```

```
    function Campolsis : Boolean;
```

```
    function CampoVinculado : Boolean;
```

```

    function RetornaCampo : String;
end;

```

```

//-----Classe TTipoDoc-----
type
  TTipoDoc = class(TObject)
    Fcod : String;
    Fnome : String;
  public
    Constructor Create;
    Destructor Destroy; Override;
    function IncluiTipoDoc : Boolean;
    function AlteraTipoDoc : Boolean;
    function ExcluiTipoDoc : Boolean;
    function CamposVinculados : Boolean;
    function CodTipoDoc : Boolean;
    function CarregaTipoDoc(var Ord : Smallint) : TQuery;
    function DescTipo : String;
  end;

```

```

//-----Classe TDocumento-----
type
  TDocumento = class(TObject)
    FtipoDoc : TTipoDoc;
    Fcampo : TCampo;
    Fid : Smallint;
  public
    Constructor Create;
    Destructor Destroy; Override;
    function VincCampos : Boolean;
    function VincCamposT : Boolean;
    function DesVincCampos : Boolean;
    function DesVincCamposT : Boolean;
    function CampoRepet : Boolean;
    function CampoObrig : Boolean;
    function VerificarObrig : TQuery;
    function VerificaRepet : TQuery;
    function PegarCampos : TQuery;
    function VerificarCampoRepet : Boolean;
  end;

```

```

//-----Classe TConteudoDoc-----
type
  TConteudoDoc = class(TObject)
    FDoc : TDocumento;
    FMfn : Longint;
    Fr : Longint;

```

```

Fregistro : String;
FtagReg : array[1..MAXMFRL] of Smallint;
Fexp : PChar;
private
  procedure AtualizaIndex;
public
  Constructor Create;
  Destructor Destroy; Override;
  function IncluiConteudo : Boolean;
  function AlteraConteudo : Boolean;
  function ExcluiConteudo : Boolean;
  function VerificaImagem : Boolean;
  function PegarDadosIsis(var Campo : Smallint; Ocor : Smallint) : PChar;
  function NumCamposReg : Smallint;
  function Pesquisaisis(var i : Integer) : Boolean;
  function QtdeRegistro : Longint;
end;

```

```
//-----Classe TImagem-----
```

```
type
```

```

TImagem = class(TObject)
  Ftwain : TDelphiTwain;
  FcodSelecionado : Integer;
  Fcor : TTwainPixelType;
  Fresolucao : Integer;
  Fimagem : TImage;
  Fthumb : TImage;
  Fdoc : TConteudoDoc;
  Fitem : Integer;
  Fdir : String;
private
  procedure MakeThumbNail(src, dest: TBitmap; ThumbSize: Word);
  procedure FTwainTwainAcquire(Sender: TObject;
    const Index: Integer; Image: TBitmap; var Cancel: Boolean);
  function Novoltem : Integer;
public
  Constructor Create;
  Destructor Destroy; Override;
  procedure CapturaImagem;
  function IncluiImagem : Boolean;
  function AlteraImagem : Boolean;
  function ExcluiImagem : Boolean;
  function PegarImagem : TQuery;
  function CarregaImagem : String;
end;

```

```
implementation
```

```

uses
  UDMFireBird, Variants, SysUtils;

var
  QryTemp : TQuery;

//-----#Classe Usuario#-----
Constructor TUsuario.Create;
begin
  Inherited Create;
  Flogin := '';
  Flogin := StrAlloc(10);
  FNome := '';
  FNome := StrAlloc(30);
  Fperfil := '';
  Fperfil := StrAlloc(1);
  Fsetor := '';
  Fsetor := StrAlloc(30);
  Fsenha := '';
  Fsenha := StrAlloc(15);
  Fstatus := '';
  Fstatus := StrAlloc(15);
end;

Destructor TUsuario.Destroy;
begin
  Inherited Destroy;
end;

//Método para incluir um usuário no sistema
function TUsuario.IncluiUsuario : Boolean;
begin
  DMFireBird.Comp_Dados(QryTemp);
  QryTemp.SQL.Text := 'INSERT INTO USERS (LOGIN,NOME,PERFIL,SETOR,'+
    ' DATA_CRIACAO,PASSWD,STATUS)+'+
    'VALUÊS
(:LOGIN,;NOME,;PERFIL,;SETOR,;DTCRIACAO,;PASSWD,;STATUS)';
  QryTemp.ParamByName('Login').Value := Flogin;
  QryTemp.ParamByName('Nome').Value := FNome;
  QryTemp.ParamByName('Perfil').Value := Fperfil;
  QryTemp.ParamByName('Setor').Value := Fsetor;
  QryTemp.ParamByName('DtCriacao').Value := Now;
  QryTemp.ParamByName('Passwd').Value := Fsenha;
  QryTemp.ParamByName('Status').Value := Fstatus;
  DMFireBird.DBDigidoc.StartTransaction;
  try
    QryTemp.ExecSQL;
    DMFireBird.DBDigidoc.Commit;
  
```

```

    Result := True;
except
    DMFireBird.DBDigidoc.Rollback;
    Result := False;
end;
QryTemp.Close;
FreeAndNil(QryTemp);
end;

//Método para alterar um usuário no sistema
function TUsuario.AlterarUsuario : Boolean;
begin
    DMFireBird.Comp_Dados(QryTemp);
    QryTemp.SQL.Text := 'UPDATE USERS SET'+
        ' NOME = :NOME, PERFIL = :PERFIL,'+
        ' SETOR = :SETOR, DATA_MODIFICACAO = :DTMODIFIC,'+
        ' STATUS = :STATUS'+
        ' WHERE LOGIN = :LOGIN';
    QryTemp.ParamByName('Login').Value := Flogin;
    QryTemp.ParamByName('Nome').Value := FNome;
    QryTemp.ParamByName('Perfil').Value := Fperfil;
    QryTemp.ParamByName('Setor').Value := Fsetor;
    QryTemp.ParamByName('DtModific').Value := Now;
    QryTemp.ParamByName('Status').Value := Fstatus;
    DMFireBird.DBDigidoc.StartTransaction;
    try
        QryTemp.ExecSQL;
        DMFireBird.DBDigidoc.Commit;
        Result := True;
    except
        DMFireBird.DBDigidoc.Rollback;
        Result := False;
    end;
    QryTemp.Close;
    FreeAndNil(QryTemp);
end;

//Método para excluir um usuário no sistema
function TUsuario.ExcluirUsuario : Boolean;
begin
    DMFireBird.Comp_Dados(QryTemp);
    QryTemp.SQL.Text := 'DELETE FROM USERS'+
        ' WHERE LOGIN = :LOGIN';
    QryTemp.ParamByName('Login').Value := Flogin;
    DMFireBird.DBDigidoc.StartTransaction;
    try
        QryTemp.ExecSQL;
        DMFireBird.DBDigidoc.Commit;
    end;
end;

```

```

    Result := True;
except
    DMFireBird.DBDigidoc.Rollback;
    Result := False;
end;
QryTemp.Close;
FreeAndNil(QryTemp);
end;

//Método para carregar o perfil do usuário
function TUsuario.CarregaPerfilUsuario : String;
begin
    DMFireBird.Comp_Dados(QryTemp);
    QryTemp.SQL.Text := 'SELECT PERFIL FROM USERS'+
        ' WHERE LOGIN = :LOGIN';
    QryTemp.ParamByName('Login').Value := Flogin;
    QryTemp.Open;
    Result := QryTemp.FieldByName('Perfil').Value;
    QryTemp.Close;
    FreeAndNil(QryTemp);
end;

//Método para verificar se o login está cadastrado
function TUsuario.VerifLogin : Boolean;
begin
    DMFireBird.Comp_Dados(QryTemp);
    QryTemp.SQL.Text := 'SELECT * FROM USERS'+
        ' WHERE LOGIN = :LOGIN';
    QryTemp.ParamByName('Login').Value := Flogin;
    QryTemp.Open;
    if not QryTemp.Eof then begin
        Result := False;
    end else begin
        Result := True;
    end;
    QryTemp.Close;
    FreeAndNil(QryTemp);
end;

//Método para carregar senha e status do usuário
function TUsuario.CarregaUsuario : Boolean;
begin
    DMFireBird.Comp_Dados(QryTemp);
    QryTemp.SQL.Text := 'SELECT STATUS, PASSWD FROM USERS'+
        ' WHERE LOGIN = :LOGIN';
    QryTemp.ParamByName('Login').Value := Flogin;
    QryTemp.Open;
    if not QryTemp.Eof then begin

```

```

    Fstatus := QryTemp.FieldName('Status').Value;
    Fsenha := QryTemp.FieldName('Passwd').Value;
    Result := True;
end else begin
    Result := False;
end;
QryTemp.Close;
FreeAndNil(QryTemp);
end;

//Método para alteração de senha do usuário
function TUsuario.AlterarSenha : Boolean;
begin
    DMFireBird.Comp_Dados(QryTemp);
    QryTemp.SQL.Text := 'UPDATE USERS SET'+
        ' PASSWD = :PASSWD, DATA_MODIFICACAO = :DtModific'+
        ' WHERE LOGIN = :LOGIN';
    QryTemp.ParamByName('Login').Value := Flogin;
    QryTemp.ParamByName('Passwd').Value := Fsenha;
    QryTemp.ParamByName('DtModific').Value := Now;
    DMFireBird.DBDigidoc.StartTransaction;
    try
        QryTemp.ExecSQL;
        DMFireBird.DBDigidoc.Commit;
        Result := True;
    except
        DMFireBird.DBDigidoc.Rollback;
        Result := False;
    end;
    QryTemp.Close;
    FreeAndNil(QryTemp);
end;

//-----#Classe TCampo#-----
Constructor TCampo.Create;
begin
    Inherited Create;
    Ftag := 0;
    Fnome := StrAlloc(20);
    Fnome := '';
    Ftamanho := 0;
    Ftipo := StrAlloc(8);
    Ftipo := '';
end;

Destructor TCampo.Destroy;
begin
    Inherited Destroy;

```

end;

//Método para inclusão de dados na tabela Campo

function TCampo.IncluiCampo;

begin

DMFireBird.Comp_Dados(QryTemp);

QryTemp.Close;

QryTemp.SQL.Text := 'SELECT MAX(TAG) + 1 AS NovoCodigo'+
' FROM CAMPO';

QryTemp.Open;

if (QryTemp.FieldByName('NovoCodigo').Value <> Null) then begin

Ftag := QryTemp.FieldByName('NovoCodigo').Value;

end

else begin

Ftag := 2;

end;

QryTemp.Close;

QryTemp.SQL.Text := 'INSERT INTO CAMPO (TAG,CAMPO,TAMANHO,TIPO)'+
' VALUES (:TAG,:CAMPO,:TAMANHO,:TIPO)';

QryTemp.ParamByName('Tag').Value := Ftag;

QryTemp.ParamByName('Campo').Value := Fnome;

QryTemp.ParamByName('Tamanho').Value := Ftamanho;

QryTemp.ParamByName('Tipo').Value := Ftipo;

DMFireBird.DBDigidoc.StartTransaction;

try

QryTemp.ExecSQL;

DMFireBird.DBDigidoc.Commit;

AtualTipoIndexCampo;

Result := True;

except

DMFireBird.DBDigidoc.Rollback;

Result := False;

end;

QryTemp.Close;

FreeAndNil(QryTemp);

end;

//Método para alteração de dados na tabela Campo

function TCampo.AlterarCampo;

begin

DMFireBird.Comp_Dados(QryTemp);

QryTemp.Close;

QryTemp.SQL.Text := 'UPDATE CAMPO SET'+
' CAMPO=:CAMPO, TAMANHO=:TAMANHO, TIPO=:TIPO'+
' WHERE TAG=:TAG';

QryTemp.ParamByName('Tag').Value := Ftag;

QryTemp.ParamByName('Campo').Value := Fnome;

QryTemp.ParamByName('Tamanho').Value := Ftamanho;

```

QryTemp.ParamByName('Tipo').Value := Ftipo;
DMFireBird.DBDigidoc.StartTransaction;
try
  QryTemp.ExecSQL;
  DMFireBird.DBDigidoc.Commit;
  Result := True;
except
  DMFireBird.DBDigidoc.Rollback;
  Result := False;
end;
QryTemp.Close;
FreeAndNil(QryTemp);
end;

```

//Método para exclusão de dados na tabela Campo

```

function TCampo.ExcluiCampo;
begin
  DMFireBird.Comp_Dados(QryTemp);
  QryTemp.Close;
  QryTemp.SQL.Text := 'DELETE FROM CAMPO'+
    ' WHERE TAG=:TAG';
  QryTemp.ParamByName('Tag').Value := Ftag;
  DMFireBird.DBDigidoc.StartTransaction;
  try
    QryTemp.ExecSQL;
    DMFireBird.DBDigidoc.Commit;
    Result := True;
  except
    DMFireBird.DBDigidoc.Rollback;
    Result := False;
  end;
  QryTemp.Close;
  FreeAndNil(QryTemp);
end;

```

//Método para verificação se há algum dado no campo referenciado

```

function TCampo.Campolsis;
var
  Control : IsisRecControl;
  r, x : Longint;
  Area, f : PChar;
  Resultado : String;
begin
  x := 1;
  Area := StrAlloc(8000);
  Result := True;
  DMFireBird.ConexaoBdWinisis;
  r := IsisRecControlMap(DMFireBird.h,Control);

```

```

f := Pchar('if p(v' + PChar(IntToStr(Ftag)) + ') then mfn fi');
while (x < Control.nxtmfn) or (Resultado <> "") do begin
  r := IsisRecRead(DMFireBird.h,0,x);
  r := IsisSpaPft(DMFireBird.h,f);
  r := IsisRecFormat(DMFireBird.h,0,Area,8000);
  Resultado := Area;
  if (Resultado <> "") then begin
    Result := False;
  end;
  x := x + 1;
end;
r := IsisSpaDelete(DMFireBird.h);
end;

//Método para verificação se o campo referenciado
//tem vínculo com algum tipo de documento
function TCampo.CampoVinculado;
begin
  DMFireBird.Comp_Dados(QryTemp);
  QryTemp.Close;
  QryTemp.SQL.Text := 'SELECT * FROM CAMPO_DOCUMENTO'+
    ' WHERE TAG_CAMPO = :Tag';
  QryTemp.ParamByName('Tag').Value := Ftag;
  QryTemp.Open;
  if QryTemp.Eof then begin
    Result := True;
  end else begin
    Result := False;
  end;
  QryTemp.Close;
  FreeAndNil(QryTemp);
end;

//Método para atualização do arquivo .fst do banco Isis
//tipo de indexação para o campo
procedure TCampo.AtualTipoIndexCampo;
var
  data : TextFile;
  Caminho : String;
  Linha : String;
begin
  Linha := IntToStr(Ftag) + ' 4 mhl,v' + IntToStr(Ftag) + '|%|';
  Caminho := DMFireBird.CaminhoIsis + '\DB_DDC.fst';
  AssignFile(data,Caminho);
  Append(data);
  Writeln(data,Linha);
  CloseFile(data);
end;

```

```

//Método que retorna a descrição do campo
function TCampo.RetornaCampo : String;
begin
  DMFireBird.Comp_Dados(QryTemp);
  QryTemp.Close;
  QryTemp.SQL.Text := 'SELECT CAMPO FROM CAMPO'+
    ' WHERE TAG = :Tag';
  QryTemp.ParamByName('Tag').Value := Ftag;
  QryTemp.Open;
  Result := QryTemp.FieldByName('Campo').Value;
  FreeAndNil(QryTemp);
end;

//-----#Classe TTipoDoc#-----
Constructor TTipoDoc.Create;
begin
  Inherited Create;
  Fcod := StrAlloc(4);
  Fcod := "";
  Fnome := StrAlloc(30);
  Fnome := "";
end;

Destructor TTipoDoc.Destroy;
begin
  Inherited Destroy;
end;

//Método para inclusão de dados na tabela Tipo_Documento
function TTipoDoc.IncluiTipoDoc : Boolean;
begin
  DMFireBird.Comp_Dados(QryTemp);
  QryTemp.Close;
  QryTemp.SQL.Text := 'INSERT INTO TIPO_DOCUMENTO (COD_TIPO, DESC_TIPO)'+
    ' VALUES (:Cod,:Descricao)';
  QryTemp.ParamByName('Cod').Value := Fcod;
  QryTemp.ParamByName('Descricao').Value := Fnome;
  DMFireBird.DBDigidoc.StartTransaction;
  try
    QryTemp.ExecSQL;
    DMFireBird.DBDigidoc.Commit;
    Result := True;
  except
    DMFireBird.DBDigidoc.Rollback;
    Result := False;
  end;
  QryTemp.Close;

```

```

    FreeAndNil(QryTemp);
end;

//Método para alteração de dados na tabela Tipo_Documento
function TTipoDoc.AlterarTipoDoc : Boolean;
begin
    DMFireBird.Comp_Dados(QryTemp);
    QryTemp.Close;
    QryTemp.SQL.Text := 'UPDATE TIPO_DOCUMENTO SET'+
        ' DESC_TIPO=:DESCRICAO'+
        ' WHERE COD_TIPO=:COD';
    QryTemp.ParamByName('Cod').Value := Fcod;
    QryTemp.ParamByName('Descricao').Value := Fnome;
    DMFireBird.DBDigidoc.StartTransaction;
    try
        QryTemp.ExecSQL;
        DMFireBird.DBDigidoc.Commit;
        Result := True;
    except
        DMFireBird.DBDigidoc.Rollback;
        Result := False;
    end;
    QryTemp.Close;
    FreeAndNil(QryTemp);
end;

//Método para exclusão de dados na tabela Tipo_Documento
function TTipoDoc.ExcluirTipoDoc : Boolean;
begin
    DMFireBird.Comp_Dados(QryTemp);
    QryTemp.Close;
    QryTemp.SQL.Text := 'DELETE FROM TIPO_DOCUMENTO'+
        ' WHERE COD_TIPO=:COD';
    QryTemp.ParamByName('Cod').Value := Fcod;
    DMFireBird.DBDigidoc.StartTransaction;
    try
        QryTemp.ExecSQL;
        DMFireBird.DBDigidoc.Commit;
        Result := True;
    except
        DMFireBird.DBDigidoc.Rollback;
        Result := False;
    end;
    QryTemp.Close;
    FreeAndNil(QryTemp);
end;

```

//Método para verificação se não há campos vinculados para o tipo de documento

```

function TTipoDoc.CamposVinculados : Boolean;
begin
  DMFireBird.Comp_Dados(QryTemp);
  QryTemp.Close;
  QryTemp.SQL.Text := 'SELECT * FROM CAMPO_DOCUMENTO'+
    ' WHERE COD_DOC = :Codigo';
  QryTemp.ParamByName('Codigo').Value := Fcod;
  QryTemp.Open;
  if QryTemp.Eof then begin
    Result := True;
  end else begin
    Result := False;
  end;
  QryTemp.Close;
  FreeAndNil(QryTemp);
end;

```

//Método que verifica se o código para o tipo de documento já não está cadastrado

```

function TTipoDoc.CodTipoDoc : Boolean;
begin
  DMFireBird.Comp_Dados(QryTemp);
  QryTemp.Close;
  QryTemp.SQL.Text := 'SELECT * FROM TIPO_DOCUMENTO'+
    ' WHERE COD_TIPO = :Cod';
  QryTemp.ParamByName('Cod').Value := Fcod;
  QryTemp.Open;
  if QryTemp.Eof then begin
    Result := True;
  end else begin
    Result := False;
  end;
  QryTemp.Close;
  FreeAndNil(QryTemp);
end;

```

//Método para carregar os tipos de documento

```

function TTipoDoc.CarregaTipoDoc(var Ord : Smallint) : TQuery;
begin
  DMFireBird.Comp_Dados(Result);
  Result.SQL.Text := 'SELECT COD_TIPO, DESC_TIPO FROM TIPO_DOCUMENTO';
  if Ord = 0 then begin
    Result.Sql.Text := Result.Sql.Text + ' ORDER BY DESC_TIPO';
  end else begin
    Result.Sql.Text := Result.Sql.Text + ' ORDER BY COD_TIPO';
  end;
  Result.Open;
end;

```

```

//Método que retorno a descrição do tipo de documento
function TTipoDoc.DescTipo : String;
begin
  DMFireBird.Comp_Dados(QryTemp);
  QryTemp.SQL.Text := 'SELECT DESC_TIPO FROM TIPO_DOCUMENTO'+
    ' WHERE COD_TIPO = :Cod';
  QryTemp.ParamByName('Cod').Value := Fcod;
  QryTemp.Open;
  Result := QryTemp.FieldByName('DESC_TIPO').Value;
  QryTemp.Close;
  FreeAndNil(QryTemp);
end;

//-----#Classe TDocumento#-----
Constructor TDocumento.Create;
begin
  Inherited Create;
  Fcampo := TCampo.Create;
  FtipoDoc := TTipoDoc.Create;
  Fid := 0;
end;

Destructor TDocumento.Destroy;
begin
  Inherited Destroy;
end;

//Método para vincular campos
function TDocumento.VincCampos : Boolean;
var
  Qry : TQuery;
begin
  DMFireBird.Comp_Dados(QryTemp);
  QryTemp.Close;
  QryTemp.SQL.Text := 'INSERT INTO CAMPO_DOCUMENTO'+
    ' (COD_DOC, TAG_CAMPO, OBRIGATORIO, REPETITIVO)+'
    ' VALUES (:Codigo, :Tag, :Obrig, :Repet)';
  QryTemp.ParamByName('Codigo').Value := FtipoDoc.Fcod;
  QryTemp.ParamByName('Tag').Value := Fcampo.Ftag;
  QryTemp.ParamByName('Obrig').Value := 'Não';
  QryTemp.ParamByName('Repet').Value := 'Não';
  DMFireBird.DBDigidoc.StartTransaction;
  try
    QryTemp.ExecSQL;
    DMFireBird.DBDigidoc.Commit;
    Result := True;
  except
    DMFireBird.DBDigidoc.Rollback;
  end;
end;

```

```

    Result := False;
end;
QryTemp.Close;
FreeAndNil(QryTemp);
end;

//Método para vincular todos os campos no tipo de documento
function TDocumento.VincCamposT : Boolean;
var
    Qry : TQuery;
begin
    DMFireBird.Comp_Dados(Qry);
    Qry.Close;
    Qry.SQL.Text := 'SELECT * FROM CAMPO'+
        ' WHERE NOT EXISTS(SELECT * FROM CAMPO_DOCUMENTO'+
        ' WHERE CAMPO.TAG = CAMPO_DOCUMENTO.TAG_CAMPO AND
COD_DOC = :Cod)';
    Qry.ParamByName('Cod').Value := FtipoDoc.Fcod;
    Qry.Open;
    if not Qry.Eof then begin
        Qry.First;
        while not Qry.Eof do begin
            Fcampo.Ftag := Qry.FieldName('TAG').Value;
            DMFireBird.Comp_Dados(QryTemp);
            QryTemp.Close;
            QryTemp.SQL.Text := 'INSERT INTO CAMPO_DOCUMENTO'+
                ' (COD_DOC, TAG_CAMPO, OBRIGATORIO, REPETITIVO)'+
                ' VALUES (:Codigo, :Tag, :Obrig, :Repet)';
            QryTemp.ParamByName('Codigo').Value := FtipoDoc.Fcod;
            QryTemp.ParamByName('Tag').Value := Fcampo.Ftag;
            QryTemp.ParamByName('Obrig').Value := 'Não';
            QryTemp.ParamByName('Repet').Value := 'Não';
            DMFireBird.DBDigidoc.StartTransaction;
            try
                QryTemp.ExecSQL;
                DMFireBird.DBDigidoc.Commit;
                Result := True;
                Qry.Next;
            except
                DMFireBird.DBDigidoc.Rollback;
                Result := False;
            end;
        end;
    end;
    QryTemp.Close;
    FreeAndNil(QryTemp);
end;
Qry.Close;
FreeAndNil(Qry);

```

end;

//Método para retirar campo vinculado

function TDocumento.DesVincCampos : Boolean;

begin

DMFireBird.Comp_Dados(QryTemp);

QryTemp.Close;

QryTemp.SQL.Text := 'DELETE FROM CAMPO_DOCUMENTO'+
 ' WHERE COD_DOC = :Codigo AND TAG_CAMPO = :Tag';

QryTemp.ParamByName('Codigo').Value := FtipoDoc.Fcod;

QryTemp.ParamByName('Tag').Value := Fcampo.Ftag;

DMFireBird.DBDigidoc.StartTransaction;

try

QryTemp.ExecSQL;

DMFireBird.DBDigidoc.Commit;

Result := True;

except

DMFireBird.DBDigidoc.Rollback;

Result := False;

end;

QryTemp.Close;

FreeAndNil(QryTemp);

end;

//Método para retirar todos os campos vinculados

function TDocumento.DesVincCamposT : Boolean;

begin

DMFireBird.Comp_Dados(QryTemp);

QryTemp.Close;

QryTemp.SQL.Text := 'DELETE FROM CAMPO_DOCUMENTO'+
 ' WHERE COD_DOC = :Codigo';

QryTemp.ParamByName('Codigo').Value := FtipoDoc.Fcod;

DMFireBird.DBDigidoc.StartTransaction;

try

QryTemp.ExecSQL;

DMFireBird.DBDigidoc.Commit;

Result := True;

except

DMFireBird.DBDigidoc.Rollback;

Result := False;

end;

QryTemp.Close;

FreeAndNil(QryTemp);

end;

//Método para marcar o campo como obrigatório ou não obrigatório para o tipo de documento

function TDocumento.CampoObrig : Boolean;

```

begin
  DMFireBird.Comp_Dados(QryTemp);
  QryTemp.SQL.Text := 'UPDATE CAMPO_DOCUMENTO SET'+
    ' OBRIGATORIO=:Obrig'+
    ' WHERE COD_DOC=:Codigo AND TAG_CAMPO=:Tag';
  if Fid = 0 then begin
    QryTemp.ParamByName('Obrig').Value := 'Sim';
  end else begin
    QryTemp.ParamByName('Obrig').Value := 'Não';
  end;
  QryTemp.ParamByName('Codigo').Value := FtipoDoc.Fcod;
  QryTemp.ParamByName('Tag').Value := Fcampo.Ftag;
  DMFireBird.DBDigidoc.StartTransaction;
  try
    QryTemp.ExecSQL;
    DMFireBird.DBDigidoc.Commit;
    Result := True;
  except
    DMFireBird.DBDigidoc.Rollback;
    Result := False;
  end;
  QryTemp.Close;
  FreeAndNil(QryTemp);
end;

```

//Método para marcar o campo como repetitivo ou não repetitivo para o tipo de documento
function TDocumento.CampoRepet : Boolean;

```

begin
  DMFireBird.Comp_Dados(QryTemp);
  QryTemp.SQL.Text := 'UPDATE CAMPO_DOCUMENTO SET'+
    ' REPETITIVO= :Repet'+
    ' WHERE COD_DOC=:Codigo AND TAG_CAMPO=:Tag';
  if Fid = 0 then begin
    QryTemp.ParamByName('Repet').Value := 'Sim';
  end else begin
    QryTemp.ParamByName('Repet').Value := 'Não';
  end;
  QryTemp.ParamByName('Codigo').Value := FtipoDoc.Fcod;
  QryTemp.ParamByName('Tag').Value := Fcampo.Ftag;
  DMFireBird.DBDigidoc.StartTransaction;
  try
    QryTemp.ExecSQL;
    DMFireBird.DBDigidoc.Commit;
    Result := True;
  except
    DMFireBird.DBDigidoc.Rollback;
    Result := False;
  end;
end;

```

```

    QryTemp.Close;
    FreeAndNil(QryTemp);
end;

```

```

//Método para carregar os campos obrigatórios
function TDocumento.VerificarObrig : TQuery;
begin
    DMFireBird.Comp_Dados(Result);
    Result.SQL.Text := 'SELECT * FROM CAMPO_DOCUMENTO'+
        ' WHERE COD_DOC = :Codigo AND OBRIGATORIO = :Obrig';
    Result.ParamByName('Codigo').Value := FtipoDoc.Fcod;
    Result.ParamByName('Obrig').Value := 'Sim';
    Result.Open;
end;

```

```

//Método para carregar os campos repetitivos
function TDocumento.VerificaRepet : TQuery;
begin
    DMFireBird.Comp_Dados(Result);
    Result.SQL.Text := 'SELECT * FROM CAMPO_DOCUMENTO'+
        ' WHERE COD_DOC = :Codigo AND REPETITIVO = :Repet';
    Result.ParamByName('Codigo').Value := FtipoDoc.Fcod;
    Result.ParamByName('Repet').Value := 'Sim';
    Result.Open;
end;

```

```

//Método para carregar os campos para um tipo de documento
function TDocumento.PegarCampos : TQuery;
begin
    DMFireBird.Comp_Dados(Result);
    Result.Close;
    Result.Sql.Text := 'SELECT CAMPO.TAG, CAMPO.CAMPO, CAMPO.TIPO,
CAMPO.TAMANHO,'+
        ' CAMPO_DOCUMENTO.OBRIGATORIO,
CAMPO_DOCUMENTO.REPETITIVO'+
        ' FROM CAMPO'+
        ' INNER JOIN CAMPO_DOCUMENTO ON
CAMPO_DOCUMENTO.TAG_CAMPO=CAMPO.TAG'+
        ' WHERE CAMPO_DOCUMENTO.COD_DOC=:DOCUMENTO'+
        ' ORDER BY REPETITIVO';
    Result.ParamByName('DOCUMENTO').Value := FtipoDoc.Fcod;
    Result.Open;
end;

```

```

//Método para verificar se o campo é repetitivo para o tipo de documento
function TDocumento.VerificarCampoRepet : Boolean;
begin
    Result := False;

```

```

DMFireBird.Comp_Dados(QryTemp);
QryTemp.Close;
QryTemp.SQL.Text := 'SELECT * FROM CAMPO_DOCUMENTO'+
    ' WHERE COD_DOC = :Doc AND TAG_CAMPO = :Tag'+
    ' AND REPETITIVO = :Repet';
QryTemp.ParamByName('Doc').Value := FtipoDoc.Fcod;
QryTemp.ParamByName('Tag').Value := Fcampo.Ftag;
QryTemp.ParamByName('Repet').Value := 'Sim';
QryTemp.Open;
if not QryTemp.Eof then begin
    Result := True;
end;
QryTemp.Close;
FreeAndNil(QryTemp);
end;

```

```
//-----#Classe TConteudoDoc#-----
```

```

Constructor TConteudoDoc.Create;
begin
    FDoc := TDocumento.Create;
    FDoc.NewInstance;
    FMfn := 0;
    Fr := 0;
    Fregistro := "";
    Fexp := StrAlloc(255);
end;

```

```

Destructor TConteudoDoc.Destroy;
begin
    FDoc.Destroy;
    Inherited Destroy;
end;

```

```

//Método para inclusão de registros no WinIsis
function TConteudoDoc.IncluiConteudo : Boolean;
begin
    Result := False;
    DMFireBird.ConexaoBdWinisis;
    FMfn := IsisRecNewLock(DMFireBird.h,0);
    Fr := IsisRecFieldUpdate(DMFireBird.h,0,Pchar(Fregistro));
    Fr := IsisRecWrite(DMFireBird.h,0);
    AtualizaIndex;
    Result := True;
    Fr := IsisSpaDelete(DMFireBird.h);
end;

```

```
//Método para alteração de registros no WinIsis
```

```

function TConteudoDoc.AlterarConteudo : Boolean;
begin
  if ExcluirConteudo then begin
    Result := False;
    DMFireBird.ConexaoBdWinisis;
    Fr := IsisRecReadLock(DMFireBird.h,0,FMfn);
    Fr := IsisRecFieldUpdate(DMFireBird.h,0,PChar(Fregistro));
    Fr := IsisRecWrite(DMFireBird.h,0);
    AtualizaIndex;
    Result := True;
    Fr := IsisSpaDelete(DMFireBird.h);
  end;
end;

//Método para exclusão de registros no Winisis
function TConteudoDoc.ExcluirConteudo : Boolean;
begin
  Result := False;
  DMFireBird.ConexaoBdWinisis;
  Fr := IsisRecReadLock(DMFireBird.h,0,FMfn);
  Fr := IsisRecFieldUpdate(DMFireBird.h,0,'d*');
  Fr := IsisRecWriteUnlock(DMFireBird.h,0);
  AtualizaIndex;
  Result := True;
  Fr := IsisSpaDelete(DMFireBird.h);
end;

//Método para atualizar a indexação do registro
procedure TConteudoDoc.AtualizaIndex;
var
  Arquivo : PChar;
begin
  Arquivo := PChar('@' + DMFireBird.CaminhoHsis + '\DB_DDC');
  Fr := IsisSpaFst(DMFireBird.h,Arquivo);
  Fr := IsisRecIfUpdate(DMFireBird.h,FMfn);
  Fr := IsisRecUnlockForce(DMFireBird.h,FMfn);
end;

//Método para verificar se o documento possui imagens
function TConteudoDoc.VerificaImagem : Boolean;
begin
  DMFireBird.Comp_Dados(QryTemp);
  QryTemp.Close;
  QryTemp.SQL.Text := 'SELECT * FROM IMG_DOCUMENTO'+
    ' WHERE COD_DOC = :Cod';
  QryTemp.ParamByName('Cod').Value := FMfn;
  QryTemp.Open;
  if not QryTemp.Eof then begin

```

```

    Result := True;
end else begin
    Result := False;
end;
QryTemp.Close;
FreeAndNil(QryTemp);
end;

```

//Método para carregar os dados do Isis

```

function TConteudoDoc.PegarDadosIsis(var Campo : Smallint; Ocor : Smallint) : PChar;
var
    Area : PChar;
begin
    DMFireBird.ConexaoBdWinisis;
    Fr := IsisRecRead(DMFireBird.h,0,FMfn);
    Area := StrAlloc(8000);
    Fr := IsisRecField(DMFireBird.h,0,Campo,Ocor, Area,8000);
    if Fr > 0 then begin
        Result := Area;
    end else begin
        Result := '0000';
    end;
    Fr := IsisSpaDelete(DMFireBird.h);
end;

```

//Método para verificar quantos campos tem no registro do Isis

```

function TConteudoDoc.NumCamposReg : Smallint;
var
    Leader : IsisRecLeader;
    TotCampo, x : Smallint;
    Dir : IsisRecDir;
begin
    DMFireBird.ConexaoBdWinisis;
    Fr := IsisRecRead(DMFireBird.h,0,FMfn);
    Fr := IsisRecLeaderMap(DMFireBird.h,0,Leader);
    for x := 1 to Leader.nvf do begin
        Fr := IsisRecDirMap(DMFireBird.h,0,x,x,Dir);
        FtagReg[x] := Dir.tag;
    end;
    Result := Leader.nvf;
    Fr := IsisSpaDelete(DMFireBird.h);
end;

```

//Método para retornar a quantidade de registros com a expressão de pesquisa

```

function TConteudoDoc.QtdeRegistro : Longint;
var
    scrHeader : IsisSrcHeader;
begin

```

```

DMFireBird.ConexaoBdWinisis;
Fr := IsisSrcSearch(DMFireBird.h, 0, Fexp, scrHeader);
if (Fr >= 0) then begin
  if scrHeader.recs > 0 then begin
    Result := scrHeader.recs;
  end else begin
    Result := 0;
  end;
end;
Fr := IsisSpaDelete(DMFireBird.h);
end;

//Método para retornar os registros com os termos da pesquisa
function TConteudoDoc.Pesquisasis(var i : Integer): Boolean;
var
  srcMfn : IsisSrcMfn;
begin
  DMFireBird.ConexaoBdWinisis;
  Fr := IsisSrcMfnMap(DMFireBird.a, 0, 0, i, i,srcMfn);
  if Fr >= 0 then begin
    FMfn := srcMfn.mfn;
    Result := True;
  end else begin
    Result := False;
  end;
  Fr := IsisSpaDelete(DMFireBird.h);
end;

//-----#Classe TImagem#-----
Constructor TImagem.Create;
begin
  Inherited Create;
  Ftwain := TDelphiTwain.Create(Ftwain);
  Ftwain.NewInstance;
  Ftwain.OnTwainAcquire := FtwainTwainAcquire;
  Fimagem := TImage.Create(Fthumb);
  Fimagem.AutoSize := True;
  Fthumb := TImage.Create(Fthumb);
  Fthumb.AutoSize := True;
  FDoc := TConteudoDoc.Create;
  FDoc.NewInstance;
  Fdir := DMFireBird.CaminhoImagens + '\';
end;

Destructor TImagem.Destroy;
begin
  Ftwain.Destroy;
  Fdoc.Destroy;

```

```

    Inherited Destroy;
end;

//Método que cria um thumbnail da imagem digitalizada
procedure TImagem.MakeThumbNail(src, dest: TBitmap; ThumbSize: Word);
type
    PRGB24 = ^TRGB24;
    TRGB24 = packed record
        B: Byte;
        G: Byte;
        R: Byte;
    end;
var
    x, y, ix, iy: integer;
    x1, x2, x3: integer;

    xscale, yscale: single;
    iRed, iGrn, iBlu, iRatio: Longword;
    p, c1, c2, c3, c4, c5: tRGB24;
    pt, pt1: pRGB24;
    iSrc, iDst, s1: integer;
    i, j, r, g, b, tmpY: integer;

    RowDest, RowSource, RowSourceStart: integer;
    w, h: integer;
    dxmin, dymin: integer;
    ny1, ny2, ny3: integer;
    dx, dy: integer;
    lutX, lutY: array of integer;
begin
    if src.PixelFormat <> pf24bit then src.PixelFormat := pf24bit;
    if dest.PixelFormat <> pf24bit then dest.PixelFormat := pf24bit;
    dest.Width := ThumbSize;
    dest.Height := ThumbSize;
    w := ThumbSize;
    h := ThumbSize;

    if (src.Width <= ThumbSize) and (src.Height <= ThumbSize) then
    begin
        dest.Assign(src);
        exit;
    end;

    iDst := (w * 24 + 31) and not 31;
    iDst := iDst div 8; //BytesPerScanline
    iSrc := (Src.Width * 24 + 31) and not 31;
    iSrc := iSrc div 8;

```

```

xscale := 1 / (w / src.Width);
yscale := 1 / (h / src.Height);

```

```

SetLength(lutX, w);
x1 := 0;
x2 := trunc(xscale);
for x := 0 to w - 1 do
begin
  lutX[x] := x2 - x1;
  x1 := x2;
  x2 := trunc((x + 2) * xscale);
end;

```

```

SetLength(lutY, h);
x1 := 0;
x2 := trunc(yscale);
for x := 0 to h - 1 do
begin
  lutY[x] := x2 - x1;
  x1 := x2;
  x2 := trunc((x + 2) * yscale);
end;

```

```

dec(w);
dec(h);
RowDest := integer(Dest.Scanline[0]);
RowSourceStart := integer(Src.Scanline[0]);
RowSource := RowSourceStart;
for y := 0 to h do
begin
  dy := lutY[y];
  x1 := 0;
  x3 := 0;
  for x := 0 to w do
  begin
    dx := lutX[x];
    iRed := 0;
    iGrn := 0;
    iBlu := 0;
    RowSource := RowSourceStart;
    for iy := 1 to dy do
    begin
      pt := PRGB24(RowSource + x1);
      for ix := 1 to dx do
      begin
        iRed := iRed + pt.R;
        iGrn := iGrn + pt.G;

```

```

    iBlu := iBlu + pt.B;
    inc(pt);
end;
RowSource := RowSource - iSrc;
end;
iRatio := 65535 div (dx * dy);
pt1 := PRGB24(RowDest + x3);
pt1.R := (iRed * iRatio) shr 16;
pt1.G := (iGrn * iRatio) shr 16;
pt1.B := (iBlu * iRatio) shr 16;
x1 := x1 + 3 * dx;
inc(x3, 3);
end;
RowDest := RowDest - iDst;
RowSourceStart := RowSource;
end;

if dest.Height < 3 then exit;

s1 := integer(dest.ScanLine[0]);
iDst := integer(dest.ScanLine[1]) - s1;
ny1 := Integer(s1);
ny2 := ny1 + iDst;
ny3 := ny2 + iDst;
for y := 1 to dest.Height - 2 do
begin
  for x := 0 to dest.Width - 3 do
  begin
    x1 := x * 3;
    x2 := x1 + 3;
    x3 := x1 + 6;

    c1 := pRGB24(ny1 + x1)^;
    c2 := pRGB24(ny1 + x3)^;
    c3 := pRGB24(ny2 + x2)^;
    c4 := pRGB24(ny3 + x1)^;
    c5 := pRGB24(ny3 + x3)^;

    r := (c1.R + c2.R + (c3.R * -12) + c4.R + c5.R) div -8;
    g := (c1.G + c2.G + (c3.G * -12) + c4.G + c5.G) div -8;
    b := (c1.B + c2.B + (c3.B * -12) + c4.B + c5.B) div -8;

    if r < 0 then r := 0 else if r > 255 then r := 255;
    if g < 0 then g := 0 else if g > 255 then g := 255;
    if b < 0 then b := 0 else if b > 255 then b := 255;

    pt1 := pRGB24(ny2 + x2);
    pt1.R := r;

```

```

    pt1.G := g;
    pt1.B := b;
end;
inc(ny1, iDst);
inc(ny2, iDst);
inc(ny3, iDst);
end;
end;

//Método para capturar a imagem do scanner
procedure TImagem.CapturaImagem;
var
    x,y : Extended;
begin
    if Ftwain.LoadLibrary then begin //Carregando Biblioteca
        Ftwain.SourceManagerLoaded := True;
        FcodSelecioneado := Ftwain.SelectSource; //Escolhendo o Scanner
        if FcodSelecioneado <> -1 then begin
            Ftwain.Source[FcodSelecioneado].Loaded := True; //Carregando o Scanner
            if Ftwain.Source[FcodSelecioneado].Loaded = True then begin
                Ftwain.Source[FcodSelecioneado].SetIPixelType(Fcor);
                Ftwain.Source[FcodSelecioneado].GetIPhysicalHeight(y,rcGet);
                Ftwain.Source[FcodSelecioneado].GetIPhysicalWidth(x,rcGet);
                Ftwain.Source[FcodSelecioneado].SetIBitDepth(8);
                Ftwain.Source[FcodSelecioneado].SetIXResolution(Fresolucao);
                Ftwain.Source[FcodSelecioneado].SetIYResolution(Fresolucao);
                Ftwain.Source[FcodSelecioneado].SetImagelayoutFrame(0,0,x,y);
                Ftwain.Source[FcodSelecioneado].TransferMode := TTwainTransferMode(1);
                Ftwain.Source[FcodSelecioneado].EnableSource(False,False);
                While Ftwain.Source[FcodSelecioneado].Enabled do begin
                    Application.ProcessMessages;
                end;
                Ftwain.UnloadLibrary;
            end else begin
                MessageDlg('Impossível localizar o Scanner.',mtError,[mbOK],0);
            end;
        end;
    end else begin
        ShowMessage('Driver Twain não esta Instalado');
    end;
end;

//Método para adicionar a imagem ao componente
procedure TImagem.FtwainTwainAcquire(Sender: TObject;
    const Index: Integer; Image: TBitmap; var Cancel: Boolean);
var
    dest: TBitmap;
begin

```

```

Fimagem.Picture.Assign(Image);
Cancel := True;
dest := TBitmap.Create;
try
  MakeThumbNail(Fimagem.Picture.Bitmap, dest, 100);
  Fthumb.Picture.Bitmap.Assign(dest);
finally
  dest.Free;
end;
end;

//Método para gravar imagem e thumbnail
function TImagem.IncluiImagem : Boolean;
var
  ImgBlob : TMemoryStream;
  ImagemBMP : TBitmap;
  ImagemJPG : TJPEGImage;
  Diretorio : String;
begin
  ImgBlob := TMemoryStream.Create;
  Fitem := Novoltem;
  DMFireBird.Comp_Dados(QryTemp);
  QryTemp.Close;
  QryTemp.SQL.Text := 'INSERT INTO IMG_DOCUMENTO'+
    '(COD_DOC, ITEM, CAMINHO_IMG, THUMB)+'
    ' VALUÉS (:Cod, :Item, :Caminho, :Imagem)';
  QryTemp.ParamByName('Cod').Value := FDoc.FMfn;
  QryTemp.ParamByName('Item').Value := Fitem;
  QryTemp.ParamByName('Caminho').Value := Fdir;
  Fthumb.Picture.Graphic.SaveToStream(ImgBlob);
  QryTemp.ParamByName('Imagem').LoadFromStream(ImgBlob,ftBlob);
  DMFireBird.DBDigidoc.StartTransaction;
  try
    QryTemp.ExecSQL;
    DMFireBird.DBDigidoc.Commit;
    ImagemBMP := TBitmap.Create;
    ImagemBMP.Assign(Fimagem.Picture);
    ImagemJPG := TJPEGImage.Create;
    ImagemJPG.Assign(ImagemBMP);
    Diretorio := Fdir + '\' + IntToStr(Fdoc.Fmfn) + '_IMG';
    if not DirectoryExists(Diretorio) then begin
      ForceDirectories(Diretorio)
    end;
    ImagemJPG.SaveToFile(Diretorio + '\' + IntToStr(Fdoc.Fmfn) + '_IMG_' +
IntToStr(Fitem) + '.jpeg');
    Result := True;
  except
    DMFireBird.DBDigidoc.Rollback;
  end;
end;

```

```

    Result := False;
end;
end;

//Método para alterar imagem e thumbnail
function TImagem.AlterarImagem : Boolean;
var
    ImgBlob : TMemoryStream;
    ImagemBMP : TBitmap;
    ImagemJPG : TJPEGImage;
    Diretorio : String;
begin
    ImgBlob := TMemoryStream.Create;
    DMFireBird.Comp_Dados(QryTemp);
    QryTemp.Close;
    QryTemp.SQL.Text := 'UPDATE IMG_DOCUMENTO'+
        ' SET THUMB = :Imagem'+
        ' WHERE COD_DOC = :Cod AND'+
        ' ITEM = :Item';
    QryTemp.ParamByName('Cod').Value := FDoc.FMfn;
    QryTemp.ParamByName('Item').Value := Fitem;
    Fthumb.Picture.Graphic.SaveToStream(ImgBlob);
    QryTemp.ParamByName('Imagem').LoadFromStream(ImgBlob,ftBlob);
    DMFireBird.DBDigidoc.StartTransaction;
    try
        QryTemp.ExecSQL;
        DMFireBird.DBDigidoc.Commit;
        ImagemBMP := TBitmap.Create;
        ImagemBMP.Assign(Fimagem.Picture);
        ImagemJPG := TJPEGImage.Create;
        ImagemJPG.Assign(ImagemBMP);
        Diretorio := Fdir + '\' + IntToStr(Fdoc.Fmfn) + '_IMG';
        ImagemJPG.SaveToFile(Diretorio + '\' + IntToStr(Fdoc.Fmfn) + '_IMG_' +
IntToStr(Fitem) + '.jpeg');
        Result := True;
        ImagemBMP.Free;
        ImagemJPG.Free;
    except
        DMFireBird.DBDigidoc.Rollback;
        Result := False;
    end;
    ImgBlob.Destroy;
    QryTemp.Close;
    FreeAndNil(QryTemp);
end;

//Método para excluir imagem e thumbnail
function TImagem.ExcluirImagem : Boolean;

```

```

var
  F : TextFile;
  Diretorio : String;
begin
  DMFireBird.Comp_Dados(QryTemp);
  QryTemp.SQL.Text := 'DELETE FROM IMG_DOCUMENTO'+
    ' WHERE COD_DOC = :Cod AND'+
    ' ITEM = :Item';
  QryTemp.ParamByName('Cod').Value := Fdoc.FMfn;
  QryTemp.ParamByName('Item').Value := Fitem;
  DMFireBird.DBDigidoc.StartTransaction;
  Try
    QryTemp.ExecSQL;
    Diretorio := Fdir + '\' + IntToStr(Fdoc.FMfn) + '_IMG';
    AssignFile(F,Diretorio + '\' + IntToStr(Fdoc.FMfn) + '_IMG_' + IntToStr(Fitem) + '.jpeg');
    Reset(F);
    CloseFile(F);
    Erase(F);
    DMFireBird.DBDigidoc.Commit;
    Result := True;
  except
    DMFireBird.DBDigidoc.Rollback;
    Result := False;
  end;
end;

//Método para verificar qual a sequência da próxima imagem
function TImagem.Novoltem : Integer;
begin
  DMFireBird.Comp_Dados(QryTemp);
  QryTemp.Close;
  QryTemp.SQL.Text := 'SELECT MAX(ITEM) + 1 AS Proxltem'+
    ' FROM IMG_DOCUMENTO'+
    ' WHERE COD_DOC =:Cod';
  QryTemp.ParamByName('Cod').Value := Fdoc.FMfn;
  QryTemp.Open;
  if QryTemp.FieldByName('Proxltem').Value <> null then begin
    Result := QryTemp.FieldByName('Proxltem').Value;
  end else begin
    Result := 1;
  end;
  QryTemp.Close;
  FreeAndNil(QryTemp);
end;

function TImagem.PegalImagem : TQuery;
begin
  DMFireBird.Comp_Dados(QryTemp);

```

```
QryTemp.SQL.Text := 'SELECT * FROM IMG_DOCUMENTO'+  
    ' WHERE COD_DOC = :Cod';  
QryTemp.ParamByName('Cod').Value := Fdoc.FMfn;  
QryTemp.Open;  
Result := QryTemp;  
end;  
  
//Método para carregar a imagem do arquivo  
function TImagem.Carregalmagem : String;  
var  
    Diretorio : String;  
begin  
    Diretorio := Fdir + '\' + IntToStr(Fdoc.Fmfn) + '_IMG';  
    Result := Diretorio + '\' + IntToStr(Fdoc.FMfn) + '_IMG_' + IntToStr(Fitem) + '.jpeg';  
end;  
  
end.
```

```
unit UConfiguracao;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, StdCtrls, IniFiles, Buttons, Isis001, Isis32, ExtCtrls,  
OleServer, Outlook2000, FileCtrl;
```

```
type
```

```
TFrmConfig = class(TForm)  
    Dialog: TOpenDialog;  
    Panel1: TPanel;  
    EdtDirAtual: TEdit;  
    Label1: TLabel;  
    Panel2: TPanel;  
    Label2: TLabel;  
    EdtNovoDir: TEdit;  
    Panel3: TPanel;  
    BtnNovo: TBitBtn;  
    BtnConfirmar: TBitBtn;  
    BtnCancelar: TBitBtn;  
    procedure FormCreate(Sender: TObject);  
    procedure FormClose(Sender: TObject; var Action: TCloseAction);  
    procedure BtnNovoClick(Sender: TObject);  
    procedure BtnCancelarClick(Sender: TObject);  
    procedure BtnConfirmarClick(Sender: TObject);  
private  
    procedure AtualizarCaminho;  
public  
    { Public declarations }  
end;
```

```
var
```

```
FrmConfig: TFrmConfig;
```

```
implementation
```

```
uses UDMFireBird;
```

```
{ $R *.dfm }
```

```
procedure TFrmConfig.FormCreate(Sender: TObject);  
begin  
    AtualizarCaminho;  
end;
```

```

procedure TFrmConfig.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  FrmConfig := nil;
  Action := caFree;
end;

procedure TFrmConfig.BtnNovoClick(Sender: TObject);
var
  S: string;
begin
  SelectDirectory('Selecionar Pasta','',S);
  if S <> '' then begin
    EdtNovoDir.Text := S;
  end;
end;

procedure TFrmConfig.BtnCancelarClick(Sender: TObject);
begin
  EdtNovoDir.Text := '';
  Close;
end;

procedure TFrmConfig.BtnConfirmarClick(Sender: TObject);
var
  Arquivo : TIniFile;
  Caminho : String;
begin
  Caminho := ExtractFilePath(ParamStr(0))+ 'Conexao.ini';
  Arquivo := TIniFile.Create(Caminho);
  Arquivo.WriteString('Banco_IMAGENS','Caminho',EdtNovoDir.Text);
  Arquivo.UpdateFile;
  Arquivo.Free;
  AtualizarCaminho;
end;

procedure TFrmConfig.AtualizarCaminho;
var
  Arquivo : TIniFile;
  Caminho : String;
begin
  EdtDirAtual.Text := '';
  EdtNovoDir.Text := '';
  Caminho := ExtractFilePath(ParamStr(0))+ 'Conexao.ini';
  Arquivo := TIniFile.Create(Caminho);
  EdtDirAtual.Text := Arquivo.ReadString('Banco_IMAGENS','Caminho','');
  Arquivo.Free;
end;

```

end.

```
unit UDigitaliza;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, StdCtrls, Buttons, ExtCtrls, DelphiTwain, Twain, DelphiTwainUtils,  
Spin, jpeg, DB, DBTables, DBXpress, clipbrd, UClasse;
```

```
type
```

```
TFrmDigitaliza = class(TForm)  
    Panel2: TPanel;  
    Panel4: TPanel;  
    BtnSalvar: TSpeedButton;  
    BtnCancelar: TSpeedButton;  
    BtnSair: TSpeedButton;  
    BtnCapturar: TSpeedButton;  
    LbNova: TLabel;  
    LbAnterior: TLabel;  
    ContainImage: TScrollBar;  
    ImageHolder: TImage;  
    Label1: TLabel;  
    CbCor: TComboBox;  
    Label2: TLabel;  
    Label3: TLabel;  
    CbResolucao: TComboBox;  
    Panel1: TPanel;  
    ImgThumb: TImage;  
    Panel3: TPanel;  
    ImgAnterior: TImage;  
    procedure FormClose(Sender: TObject; var Action: TCloseAction);  
    procedure BtnCapturarClick(Sender: TObject);  
    procedure FormCreate(Sender: TObject);  
    procedure BtnSairClick(Sender: TObject);  
    procedure BtnSalvarClick(Sender: TObject);  
    procedure ImageHolderMouseMove(Sender: TObject; Shift: TShiftState; X,  
        Y: Integer);  
    procedure ImageHolderMouseDown(Sender: TObject; Button: TMouseButton;  
        Shift: TShiftState; X, Y: Integer);  
    procedure BtnCancelarClick(Sender: TObject);  
private  
    procedure CarregarImagemAnterior;  
    function InclusaoImagem : Boolean;  
    function AlteracaoImagem : Boolean;  
    procedure HabDesab_Botoes;  
public  
    { Public declarations }
```

end;

var

 ImgDig : TImage;
 FrmDigitaliza: TFrmDigitaliza;
 QryTemp : TQuery;
 ClickPos: TPoint;
 ImgBlob : TStream;

implementation

uses UDMFireBird, UlmagemDoc;

{ \$R *.dfm }

procedure TFrmDigitaliza.FormClose(Sender: TObject;

 var Action: TCloseAction);

begin

 ImgDig.Destroy;
 FrmImagensDoc.Enabled := True;
 FrmImagensDoc.BtnAplicar.Click;
 FrmDigitaliza := nil;
 Action := CaFree;

end;

procedure TFrmDigitaliza.BtnCapturarClick(Sender: TObject);

begin

 ImgDig.NewInstance;
 ImgDig.Fdoc.FMfn := StrToInt(FrmImagensDoc.EdtCodDoc.Text);
 if CbCor.ItemIndex = 0 then begin
 ImgDig.Fcor := tbdGray;
 end else begin
 if CbCor.ItemIndex = 1 then begin
 ImgDig.Fcor := tbdBw;
 end else begin
 ImgDig.Fcor := tbdRgb;
 end;
 end;

end;

if CbResolucao.ItemIndex = 0 then begin

 ImgDig.Fresolucao := 25

end else begin

 if CbResolucao.ItemIndex = 1 then begin

 ImgDig.Fresolucao := 50

 end else begin

 if CbResolucao.ItemIndex = 2 then begin

 ImgDig.Fresolucao := 75

 end else begin

```

    ImgDig.Fresolucao := 150;
  end;
end;
end;
ImgDig.CapturaImagem;
ImageHolder.Picture.Assign(ImgDig.Fimagem.Picture);
ImgThumb.Picture.Assign(ImgDig.Fthumb.Picture);
HabDesab_Botoes;
end;

procedure TFrmDigitaliza.FormCreate(Sender: TObject);
begin
  ImgDig := TImagem.Create;
  if FrmImagensDoc.Novo = True then begin
    LbAnterior.Visible := False;
  end else begin
    LbAnterior.Visible := True;
    CarregarImagemAnterior;
  end;
  //count := -15;
  //ileft := -120;
end;

procedure TFrmDigitaliza.BtnSairClick(Sender: TObject);
begin
  Close;
end;

procedure TFrmDigitaliza.BtnSalvarClick(Sender: TObject);
begin
  if FrmImagensDoc.Novo = True then begin
    if InclusaoImagem then begin
      if ImgDig.IncluiImagem then begin
        MessageDlg('Imagem incluída com sucesso.',mtInformation,[mbOK],0);
        BtnCancelar.Click;
      end else begin
        DMFireBird.MensagemErroBanco;
      end;
    end;
  end else begin
    if AlteracaoImagem then begin
      ImgDig.Fitem := StrToInt(FrmImagensDoc.LblItem.Caption);
      if ImgDig.AlterarImagem then begin
        MessageDlg('Imagem alterado com sucesso.',mtInformation,[mbOK],0);
        BtnCancelar.Click;
      end else begin
        DMFireBird.MensagemErroBanco;
      end;
    end;
  end;
end;

```

```

    end;
  end;
end;

```

```

procedure TFrmDigitaliza.CarregarImagemAnterior;
begin
  ImgDig.NewInstance;
  ImgDig.Fdoc.FMfn := StrToInt(FrmImagensDoc.EdtCodDoc.Text);
  ImgDig.Fitem := StrToInt(FrmImagensDoc.LblItem.Caption);
  QryTemp := ImgDig.Pegalmagem;
  if not QryTemp.Eof then begin
    QryTemp.First;
    while QryTemp.FieldName('Item').Value <> ImgDig.Fitem do begin
      QryTemp.Next;
    end;
    ImgBlob := QryTemp.CreateBlobStream(QryTemp.FieldName('THUMB'),bmRead);
    ImgAnterior.Picture.Bitmap.LoadFromStream(ImgBlob);
  end;
  QryTemp.Close;
  FreeAndNil(QryTemp);
end;

```

```

procedure TFrmDigitaliza.ImageHolderMouseMove(Sender: TObject;
  Shift: TShiftState; X, Y: Integer);
var
  NewPos: TPoint;
begin
  if ssLeft in Shift then
    begin
      NewPos.X := ImageHolder.Left + x - ClickPos.x;
      NewPos.Y := ImageHolder.Top + y - ClickPos.y;
      if NewPos.x + ImageHolder.Width < ContainImage.Width then
        NewPos.x := ContainImage.Width - ImageHolder.Width;
      if NewPos.y + ImageHolder.Height < ContainImage.Height then
        NewPos.y := ContainImage.Height - ImageHolder.Height;
      if NewPos.X > 0 then NewPos.X := 0;
      if NewPos.Y > 0 then NewPos.Y := 0;

      ImageHolder.Top := NewPos.Y;
      ImageHolder.Left := NewPos.X;
    end
  end;
end;

```

```

procedure TFrmDigitaliza.ImageHolderMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  ClickPos.x := X;
  ClickPos.y := Y;

```

end;

function TFrmDigitaliza.InclusaoImagem;

begin

if MessageDlg('Confirma inclusão da imagem?',mtConfirmation,[mbYes,mbNo],0) = 6
then

Result := True

else

Result := False;

end;

function TFrmDigitaliza.AlteracaoImagem;

begin

if MessageDlg('Confirma alteração da imagem?',mtConfirmation,[mbYes,mbNo],0) = 6
then

Result := True

else

Result := False;

end;

procedure TFrmDigitaliza.BtnCancelarClick(Sender: TObject);

begin

ImageHolder.Picture := nil;

ImgThumb.Picture := nil;

HabDesab_Botoes;

end;

procedure TFrmDigitaliza.HabDesab_Botoes;

begin

BtnCapturar.Enabled := not BtnCapturar.Enabled;

BtnSalvar.Enabled := not BtnSalvar.Enabled;

BtnCancelar.Enabled := not BtnCancelar.Enabled;

BtnSair.Enabled := not BtnSair.Enabled;

end;

end.

```
unit UDMFireBird;
```

```
interface
```

```
uses
```

```
    SysUtils, Classes, DB, DBTables, ISIS32, ISIS001, Messages, Dialogs;
```

```
type
```

```
    TDMFireBird = class(TDataModule)
        DBDigidoc: TDatabase;
        DsCampo: TDataSource;
        QryCampo: TQuery;
        QryCampoTAG: TIntegerField;
        QryCampoCAMPO: TStringField;
        QryCampoTIPO: TStringField;
        QryCampoTAMANHO: TIntegerField;
        QryTipoDoc: TQuery;
        DsTipoDoc: TDataSource;
        QryUsuario: TQuery;
        DsUsuario: TDataSource;
        QryPesqCampo: TQuery;
        DsPesqCampo: TDataSource;
        QryPesqCampoTAG: TIntegerField;
        QryPesqCampoCAMPO: TStringField;
        QryPesqCampoTIPO: TStringField;
        QryPesqCampoTAMANHO: TIntegerField;
        QryVincCamposG: TQuery;
        DsVincCamposG: TDataSource;
        QryCampVinc: TQuery;
        DsCampVinc: TDataSource;
        QryCampVincTAG_CAMPO: TIntegerField;
        QryCampVincCAMPO: TStringField;
        QryCampVincOBRIGATORIO: TStringField;
        QryCampVincREPETITIVO: TStringField;
        QryTipoDocCOD_TIPO: TStringField;
        QryTipoDocDESC_TIPO: TStringField;
        QryVincCamposGTAG: TIntegerField;
        QryVincCamposGCAMPO: TStringField;
        QryVincCamposGTAMANHO: TIntegerField;
        QryVincCamposGTIPO: TStringField;
        QryUsuarioLOGIN: TStringField;
        QryUsuarioNOME: TStringField;
        QryUsuarioPERFIL: TStringField;
        QryUsuarioSETOR: TStringField;
        QryUsuarioDATA_CRIACAO: TDateTimeField;
        QryUsuarioPASSWD: TStringField;
        QryUsuarioDATA_MODIFICACAO: TDateTimeField;
```

```

    QryUsuarioSTATUS: TStringField;
    procedure DataModuleCreate(Sender: TObject);
    procedure DataModuleDestroy(Sender: TObject);
private
    { Private declarations }
public
    a, h : Longint;
    CodCadastro : Longint;
    CaminhoImagens : String;
    Caminholsis : String;
    CaminhoFB : String;
    procedure Comp_Dados (Var Qry : TQuery);
    procedure Cria_Ds (Var DtS : TDataSource);
    procedure ConexaoBdWinisis;
    function MensagemInclusao : Boolean;
    function MensagemAlteracao : Boolean;
    function MensagemExclusao : Boolean;
    procedure ConfirmacaoInclusao;
    procedure ConfirmacaoExclusao;
    procedure ConfirmacaoAlteracao;
    procedure MensagemErroBanco;
    function VerificarCampoNumerico(var Texto : String) : Boolean;
    procedure FundoTelaNormal;
    procedure FundoTelaBranco;
    procedure AtualizarArqInvertido(var Mfn : Longint);
    procedure HabilitarSistema;
end;

var
    DMFireBird: TDMFireBird;
    r : Longint;

implementation

uses UCadCampos, UTipoDoc, IniFiles, UPrincipal;

{$R *.dfm}

procedure TDMFireBird.DataModuleCreate(Sender: TObject);
var
    Caminho : TIniFile;
    Diretorio : String;
begin
    a := IsisAppNew();
    Diretorio := ExtractFilePath(ParamStr(0))+ 'Conexao.ini';
    Caminho := TIniFile.Create(Diretorio);

    //Caminho do Banco de Dados FireBird

```

```

CaminhoFB := Caminho.ReadString('Banco_FB','Caminho','');
DBDigidoc.Params[0] := 'SERVER NAME=' + CaminhoFB;
DBDigidoc.Connected := True;

//Caminho do Banco Isis
Caminholisis := Caminho.ReadString('Banco_ISIS','Caminho','');

//Caminho do diretório das imagens
Caminholimagens := Caminho.ReadString('Banco_IMAGENS','Caminho','');

Caminho.Free;
end;

procedure TDMFireBird.DataModuleDestroy(Sender: TObject);
begin
  r := IsisAppDelete(a);
end;

procedure TDMFireBird.Comp_Dados (Var Qry : TQuery);
begin
  Qry := TQuery.Create(Self);
  Qry.DatabaseName := DMFireBird.DBDigidoc.DatabaseName;
end;

procedure TDMFireBird.Cria_Ds(Var DtS : TDataSource);
begin
  DtS := TDataSource.Create(Self);
end;

procedure TDMFireBird.ConexaoBdWinisis;
var
  Arq : PAnsiChar;
  Control : IsisRecControl;
begin
  h := IsisSpaNew(a);
  Arq := PChar(Caminholisis + '\DB_DDC');
  r := IsisSpaMf(DMFireBird.h, Arq);
  r := IsisSpalf(DMFireBird.h, Arq);
  r:= IsisAppDebug(a, DEBUG_LIGHT); //Desligando as mensagens de erros fatais
end;

function TDMFireBird.MensagemInclusao : Boolean;
begin
  if MessageDlg('Confirma inclusão do registro?',mtConfirmation,[mbYes,mbNo],0) = 6
  then
    Result := True
  else
    Result := False;
end;

```

```

end;

function TDMFireBird.MensagemAlteracao : Boolean;
begin
  if MessageDlg('Confirma alteração do registro?',mtConfirmation,[mbYes,mbNo],0) = 6
  then
    Result := True
  else
    Result := False;
  end;

function TDMFireBird.MensagemExclusao : Boolean;
begin
  if MessageDlg('Confirma exclusão do registro?',mtConfirmation,[mbYes,mbNo],0) = 6
  then
    Result := True
  else
    Result := False;
  end;

procedure TDMFireBird.ConfirmacaoInclusao;
begin
  MessageDlg('Registro incluído com sucesso.',mtInformation,[mbOK],0);
end;

procedure TDMFireBird.ConfirmacaoAlteracao;
begin
  MessageDlg('Registro alterado com sucesso.',mtInformation,[mbOK],0);
end;

procedure TDMFireBird.ConfirmacaoExclusao;
begin
  MessageDlg('Registro excluído com sucesso.',mtInformation,[mbOK],0);
end;

procedure TDMFireBird.MensagemErroBanco;
begin
  MessageDlg('Problemas de acesso ao banco de dados.' + #13 + 'Contacte o
administrador do sistema.',mtError,[mbOK],0);
end;

function TDMFireBird.VerificarCampoNumerico(var Texto : String) : Boolean;
var
  Tamanho, x : Integer;
begin
  Result := True;
  Tamanho := Length(Texto);
  for x := 1 to Tamanho do begin

```

```

    if not (Copy(Texto,x,1) = '0') and (Copy(Texto,x,1) <= '9') then begin
        Result := False;
    end;
end;
end;

procedure TDMFireBird.FundoTelaNormal;
begin
    FrmPrincipal.Color := $006A6A68;
end;

procedure TDMFireBird.FundoTelaBranco;
begin
    FrmPrincipal.Color := $00FFFFFF;
end;

procedure TDMFireBird.AtualizarArqInvertido(var Mfn : Longint);
var
    Arquivo : PChar;
begin
    Arquivo := PChar('@' + DMFireBird.Caminholis + '\DB_DDC');
    r := IsisSpaFst(DMFireBird.h,Arquivo);
    r := IsisReclfUpdate(DMFireBird.h,Mfn);
    r := IsisRecUnlockForce(DMFireBird.h,0);
end;

procedure TDMFireBird.HabilitaSistema;
var
    Totltens, x : Integer;
begin
    Totltens := FrmPrincipal.MenuPrincipal.Items.Count;
    for x := 0 to (Totltens - 1) do begin
        FrmPrincipal.MenuPrincipal.Items[x].Enabled := not
FrmPrincipal.MenuPrincipal.Items[x].Enabled;
    end;
    FrmPrincipal.BtnCadCampos.Enabled := not FrmPrincipal.BtnCadCampos.Enabled;
    FrmPrincipal.BtnTipoDoc.Enabled := not FrmPrincipal.BtnTipoDoc.Enabled;
    FrmPrincipal.BtnCadDocumentos.Enabled := not
FrmPrincipal.BtnCadDocumentos.Enabled;
    FrmPrincipal.BtnDigDocumentos.Enabled := not
FrmPrincipal.BtnDigDocumentos.Enabled;
    FrmPrincipal.BtnPesqDocumentos.Enabled := not
FrmPrincipal.BtnPesqDocumentos.Enabled;
end;

end.

```

```
unit UlmagemDoc;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, StdCtrls, Buttons, ExtCtrls, DBXpress, DB, DBTables, Isis001, Isis32,  
ComCtrls, UClasse, Jpeg;
```

```
type
```

```
TFrmImagensDoc = class(TForm)
```

```
Panel2: TPanel;
```

```
Label1: TLabel;
```

```
Label2: TLabel;
```

```
BtnAplicar: TSpeedButton;
```

```
EdtCodigo: TEdit;
```

```
EdtTipoDoc: TEdit;
```

```
Panel1: TPanel;
```

```
SbThumb: TScrollBar;
```

```
Panel3: TPanel;
```

```
BtnNovo: TSpeedButton;
```

```
BtnSair: TSpeedButton;
```

```
BtnEditar: TSpeedButton;
```

```
BtnExcluir: TSpeedButton;
```

```
EdtCodDoc: TEdit;
```

```
Label3: TLabel;
```

```
LblItem: TLabel;
```

```
SbDados: TScrollBar;
```

```
Label5: TLabel;
```

```
EdtImagem: TEdit;
```

```
LbRegistro: TLabel;
```

```
LbSelecao: TLabel;
```

```
Panel4: TPanel;
```

```
ImgCar: TImage;
```

```
BtnPesqDocumentos: TBitBtn;
```

```
procedure BtnAplicarClick(Sender: TObject);
```

```
procedure BtnSairClick(Sender: TObject);
```

```
procedure FormClose(Sender: TObject; var Action: TCloseAction);
```

```
procedure BtnExcluirClick(Sender: TObject);
```

```
procedure BtnNovoClick(Sender: TObject);
```

```
procedure FormCreate(Sender: TObject);
```

```
procedure BtnEditarClick(Sender: TObject);
```

```
procedure EdtCodigoKeyPress(Sender: TObject; var Key: Char);
```

```
procedure ImgCarDbClick(Sender: TObject);
```

```
procedure BtnPesqDocumentosClick(Sender: TObject);
```

```
private
```

```
procedure ClicarImagem(Sender : TObject);
```

```

    procedure DuploClickImagem(Sender : TObject);
    procedure LimparTela;
    procedure BuscarDadosRegistro;
    function AcessoPerfil : Boolean;
public
    Novo : Boolean;
end;

```

```

var
    FrmImagensDoc: TFrmImagensDoc;
    ConteudoDoc : TConteudoDoc;
    ImgDir : TImagem;

```

implementation

uses UDMFireBird, ShellApi, UDigitaliza, UPesquisaDoc, Math, UPrincipal;

{ \$R *.dfm }

```

procedure TFrmImagensDoc.BtnAplicarClick(Sender: TObject);
var
    ImgBlob : TStream;
    Campo : Smallint;
    Ocor : Smallint;
    Esq : Boolean;
    x : Integer;
    ImgTh : TImage;
    Area : PChar;
begin
    Esq := True;
    x := 0;
    LimparTela;
    LblItem.Caption := '';
    Area := StrAlloc(8000);
    if not (EdtCodigo.Text = '') then begin
        ImgDir := TImagem.Create;
        ImgDir.NewInstance;
        ImgDir.Fdoc.FMfn := StrToInt(EdtCodigo.Text);
        Campo := 1;
        Ocor := 1;
        Area := ImgDir.Fdoc.PegarDadosIsis(Campo,Ocor);
        if Area <> '0000' then begin
            LbRegistro.Visible := True;
            LbSelecao.Caption := EdtCodigo.Text;
            BuscarDadosRegistro;
            ImgDir.Fdoc.FDoc.FtipoDoc.Fcod := Area;
            EdtTipoDoc.Text := ImgDir.Fdoc.FDoc.FtipoDoc.DescTipo;
            QryTemp := ImgDir.PegalImagem;

```

```

if not QryTemp.Eof then begin
  QryTemp.First;
  ImgBlob := TStream.Create;
  while not QryTemp.Eof do begin
    ImgTh := TImage.Create(Self);
    ImgTh.Width := 5;
    ImgTh.Height := 5;
    ImgTh.AutoSize := True;
    ImgTh.Top := 9;
    ImgTh.Hint := IntToStr(QryTemp.FieldName('ITEM').Value);
    ImgTh.ShowHint := True;
    if Esq then begin
      ImgTh.Left := 7;
      if x > 0 then begin
        ImgTh.Top := 9 + (50 * x) + (5 * x);
      end else begin
        ImgTh.Top := 9;
      end;
    end else begin
      ImgTh.Left := 7 + (50 * 2) + (5 * 2);
      if x > 0 then begin
        ImgTh.Top := 9 + (50 * (x-1)) + (5 * (x-1));
      end else begin
        ImgTh.Top := 9;
      end;
    end;
    ImgTh.Tag := QryTemp.FieldName('ITEM').Value;
    ImgTh.OnClick := ClicarImagem;
    ImgTh.OnDblClick := DuploClickImagem;
    ImgBlob :=
QryTemp.CreateBlobStream(QryTemp.FieldName('THUMB'),bmRead);
    ImgTh.Picture.Bitmap.LoadFromStream(ImgBlob);
    ImgTh.Parent := FrmImagensDoc.SbThumb;
    QryTemp.Next;
    Esq := not Esq;
    x := x + 1;
  end;
end;
QryTemp.Close;
FreeAndNil(QryTemp);
EdtCodDoc.Text := EdtCodigo.Text;
end else begin
  MessageDlg('Registro não encontrado.',mtError,[mbOK],0);
end;
end else begin
  MessageDlg('É preciso informar um registro para poder visualizar suas
imagens.',mtError,[mbOK],0);
end;

```

end;

```
procedure TFrmImagensDoc.BtnSairClick(Sender: TObject);
begin
  Close;
end;
```

```
procedure TFrmImagensDoc.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  if Assigned(FrmPesquisaDoc) then begin
    FrmPesquisaDoc.Enabled := True;
  end;
  FrmImagensDoc := nil;
  Action := caFree;
end;
```

```
procedure TFrmImagensDoc.ClicarImagem(Sender : TObject);
var
  ImgTh : TImage;
begin
  LblItem.Caption := ImgTh.Hint;
end;
```

```
procedure TFrmImagensDoc.DuploClickImagem(Sender : TObject);
begin
  EdtImagem.Text := LblItem.Caption;
  ImgDir := TImagem.Create;
  ImgDir.Fitem := StrToInt(LblItem.Caption);
  ImgDir.Fdoc.FMfn := StrToInt(EdtCodDoc.Text);
  ImgCar.Picture.LoadFromFile(ImgDir.CarregalImagem);
  ImgDir.Destroy;
end;
```

```
procedure TFrmImagensDoc.LimparTela;
var
  x : Integer;
begin
  for x := FrmImagensDoc.ComponentCount -1 Downto 1 do begin
    if (Components[x] is TImage) and (TImage(Components[x]).Tag <> 0) then begin
      TImage(Components[x]).Destroy;
    end else begin
      if (Components[x] is TLabel) and (TLabel(Components[x]).Tag = 0) then begin
        TLabel(Components[x]).Destroy;
      end;
    end;
  end;
  LbRegistro.Visible := False;
```

```

LbSelecao.Caption := '';
EdtTipoDoc.Text := '';
EdtCodDoc.Text := '';
EdtImagem.Text := '';
ImgCar.Picture := nil;
end;

procedure TFrmImagensDoc.BtnExcluirClick(Sender: TObject);
begin
  if AcessoPerfil then begin
    if EdtCodDoc.Text <> '' then begin
      if LblItem.Caption <> '' then begin
        if DMFireBird.MensagemExclusao then begin
          ImgDir := TImagem.Create;
          ImgDir.Fdoc.FMfn := StrToInt(EdtCodDoc.Text);
          ImgDir.Fitem := StrToInt(LblItem.Caption);
          if ImgDir.ExcluirImagem then begin
            MessageDlg('Imagem excluída com sucesso.', mtInformation, [mbOK], 0);
          end else begin
            DMFireBird.MensagemErroBanco;
          end;
          ImgDir.Destroy;
        end;
        BtnAplicar.Click;;
      end else begin
        MessageDlg('É preciso informar uma imagem para poder excluí-la.', mtError, [mbOK], 0);
      end;
    end else begin
      MessageDlg('Não há nenhum documento selecionado.', mtError, [mbOK], 0);
    end;
  end;
end;

procedure TFrmImagensDoc.BtnNovoClick(Sender: TObject);
begin
  if AcessoPerfil then begin
    if EdtTipoDoc.Text <> '' then begin
      Novo := True;
      FrmImagensDoc.Enabled := False;
      if Assigned(FrmDigitaliza) then begin
        FrmDigitaliza.BringToFront;
        FrmDigitaliza.WindowState := wsNormal;
      end else begin
        FrmDigitaliza := TFrmDigitaliza.Create(Self);
        FrmDigitaliza.Show;
      end;
    end else begin

```

```

    MessageDlg('É preciso selecionar um documento para incluir uma imagem à
ele.',mtError,[mbOK],0);
    end;
    end;
end;

```

```

procedure TFrmImagensDoc.FormCreate(Sender: TObject);
var
    I : Integer;
begin
    Self.Left := 1;
    Self.Top := 57;
    if Assigned(FrmPesquisaDoc) then begin
        I := FrmPesquisaDoc.Linha;
        EdtCodigo.Text := FrmPesquisaDoc.SgDados.Cells[0,I];
    end;
    Novo := True;
end;

```

```

procedure TFrmImagensDoc.BtnEditarClick(Sender: TObject);
begin
    if AcessoPerfil then begin
        if EdtTipoDoc.Text <> " then begin
            if (LblItem.Caption <> ") then begin
                ImgDir.Fitem := StrToInt(LblItem.Caption);
                Novo := False;
                FrmImagensDoc.Enabled := False;
                if Assigned(FrmDigitaliza) then begin
                    FrmDigitaliza.BringToFront;
                    FrmDigitaliza.WindowState := wsNormal;
                end else begin
                    FrmDigitaliza := TFrmDigitaliza.Create(Self);
                    FrmDigitaliza.Show;
                end;
            end else begin
                MessageDlg('É preciso selecionar uma imagem para alterá-la.',mtError,[mbOK],0);
            end;
        end else begin
            MessageDlg('É preciso selecionar um documento para alterar uma
imagem.',mtError,[mbOK],0);
        end;
    end;
end;

```

```

procedure TFrmImagensDoc.EdtCodigoKeyPress(Sender: TObject; var Key: Char);
begin
    if not (Key in ['0'..'9',Chr(8)]) then begin
        Key:= #0;
    end;
end;

```

```

end;
end;

procedure TFrmImagensDoc.BuscarDadosRegistro;
var
  Campo, Cont, TotCampos, TotComp, x : Smallint;
  Area : PChar;
  Caption : String;
  Dir, Sup : Smallint;
  LbC, LbD : TLabel;
begin
  Area := StrAlloc(8000);
  x := 1;
  TotComp := 0;
  Cont := 0;
  Campo := 0;
  Dir := 104;
  Sup := 5;
  TotCampos := ImgDir.Fdoc.NumCamposReg;
  while x <= TotCampos do begin
    if ImgDir.Fdoc.FtagReg[x] <> 1 then begin
      if Campo <> ImgDir.Fdoc.FtagReg[x] then begin
        Campo := ImgDir.Fdoc.FtagReg[x];
        ImgDir.Fdoc.FDoc.Fcampo.Ftag := ImgDir.Fdoc.FtagReg[x];
        cont := 1;
      end else begin
        cont := cont + 1;
      end;
    end;
    Area := ImgDir.Fdoc.PegarDadosIsis(Campo,Cont);
    if Cont = 1 then begin
      LbC := TLabel.Create(Self);
      LbC.Name := 'Campo_' + IntToStr(x);
      LbC.AutoSize := True;
      LbC.Caption := ImgDir.Fdoc.FDoc.Fcampo.RetornaCampo;
      LbC.Left := 5;
      LbC.Top := Sup;
      LbC.Height := 13;
      LbC.Font.Style := [fsBold];
      LbC.Font.Color := clWhite;
      LbC.Alignment := taLeftJustify;
      LbC.Parent := FrmImagensDoc.SbDados;
      Sup := LbC.Height + Sup + 5;
    end;
    LbD := TLabel.Create(Self);
    LbD.Name := 'Dados_' + IntToStr(x);
    LbD.AutoSize := True;
    LbD.Caption := Area;
    LbD.Left := 39;
  end;
end;

```

```

    LbD.Top := Sup;
    LbD.Height := 13;
    LbD.Font.Color := clWhite;
    LbD.Alignment := taLeftJustify;
    LbD.Parent := FrmImagensDoc.SbDados;
    LbD.Parent := FrmImagensDoc.SbDados;
    Sup := LbD.Height + Sup + 5;
end;
x := x + 1;
end;
end;

procedure TFrmImagensDoc.ImgCarDbClick(Sender: TObject);
begin
    if EdtImagem.Text <> '' then begin
        ImgDir := TImagem.Create;
        ImgDir.Fitem := StrToInt(LblItem.Caption);
        ImgDir.Fdoc.FMfn := StrToInt(EdtCodDoc.Text);
        ImgCar.Picture.LoadFromFile(ImgDir.Carregalmagem);
        shellexecute(0,pchar('open'),PChar(ImgDir.Carregalmagem),nil,PChar(ImgDir.Fdir),SW
        _SHOW);
        ImgDir.Destroy;
    end;
end;

end;

procedure TFrmImagensDoc.BtnPesqDocumentosClick(Sender: TObject);
begin
    if Assigned(FrmPesquisaDoc) then begin
        FrmPesquisaDoc.Show;
        FrmPesquisaDoc.BringToFront;
        FrmPesquisaDoc.WindowState := wsNormal;
    end else begin
        FrmPesquisaDoc := TFrmPesquisaDoc.Create(Self);
        FrmPesquisaDoc.Show;
    end;
    FrmImagensDoc.Enabled := False;
end;

function TFrmImagensDoc.AcessoPerfil : Boolean;
begin
    Result := True;
    if FrmPrincipal.Perfil = 'U' then begin
        Result := False;
        MessageDlg('Seu perfil não permite acesso à esta função.',mtInformation,[mbOK],0);
    end;
end;
end;

```

end.

```
unit ULogin;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, StdCtrls, DBCtrls, Buttons, ExtCtrls, DBTables, Grids, UClasse;
```

```
type
```

```
TFrmLogin = class(TForm)
```

```
Panel1: TPanel;
```

```
EdtLogin: TEdit;
```

```
EdtSenha: TEdit;
```

```
Label1: TLabel;
```

```
Label2: TLabel;
```

```
BtnOK: TBitBtn;
```

```
BtnCancelar: TBitBtn;
```

```
Label3: TLabel;
```

```
EdSenhatual: TEdit;
```

```
EdNovasenha: TEdit;
```

```
EdConfNovaSenha: TEdit;
```

```
Label4: TLabel;
```

```
Label5: TLabel;
```

```
Label6: TLabel;
```

```
BtnOkSenha: TBitBtn;
```

```
BtnAlterarSenha: TBitBtn;
```

```
procedure BtnCancelarClick(Sender: TObject);
```

```
procedure BtnOKClick(Sender: TObject);
```

```
procedure BtnOkSenhaClick(Sender: TObject);
```

```
procedure BtnAlterarSenhaClick(Sender: TObject);
```

```
procedure FormClose(Sender: TObject; var Action: TCloseAction);
```

```
procedure EdSenhatualExit(Sender: TObject);
```

```
procedure EdNovasenhaExit(Sender: TObject);
```

```
procedure EdConfNovaSenhaExit(Sender: TObject);
```

```
procedure BtnOKEnter(Sender: TObject);
```

```
procedure BtnAlterarSenhaEnter(Sender: TObject);
```

```
private
```

```
vAS : boolean;
```

```
public
```

```
end;
```

```
var
```

```
FrmLogin: TFrmLogin;
```

```
Usuario : TUsuario;
```

```
implementation
```

```
uses UPrincipal, UCadDoc, USelCampos, UDMFireBird;
```

```
{ $R *.dfm }
```

```
procedure TFrmLogin.BtnCancelarClick(Sender: TObject);
```

```
begin
```

```
  Application.Terminate;
```

```
end;
```

```
procedure TFrmLogin.BtnOKClick(Sender: TObject);
```

```
begin
```

```
  Usuario := TUsuario.Create;
```

```
  Usuario.Flogin := EdtLogin.Text;
```

```
  if Usuario.CarregaUsuario then begin
```

```
    if EdtSenha.Text <> Usuario.Fsenha then begin
```

```
      ShowMessage('Login/Senha incorreta!');
```

```
      EdtSenha.SelectAll;
```

```
    end else begin
```

```
      if Usuario.Fstatus = 'I' then begin
```

```
        ShowMessage('Usuário Inativo/Bloqueado.' + #13 + 'Contate o administrador!');
```

```
        Application.Terminate;
```

```
      end else begin
```

```
        if Usuario.Flogin = Usuario.Fsenha then begin
```

```
          vAS := True;
```

```
          ShowMessage('Alteração de Senha obrigatória!');
```

```
        end;
```

```
        if vAS then begin
```

```
          Label3.Visible := True;
```

```
          Label4.Visible := True;
```

```
          Label5.Visible := True;
```

```
          Label6.Visible := True;
```

```
          EdSenhatual.Visible := True;
```

```
          EdNovaSenha.Visible := True;
```

```
          EdConfNovaSenha.Visible := True;
```

```
          BtnOKSenha.Visible := True;
```

```
          Label1.Visible := False;
```

```
          Label2.Visible := False;
```

```
          EdtLogin.Visible := False;
```

```
          EdtSenha.Visible := False;
```

```
          BtnOk.Visible := False;
```

```
          BtnCancelar.Visible := False;
```

```
          BtnAlterarSenha.Visible := False;
```

```
          EdSenhatual.SetFocus;
```

```
        end else begin
```

```
          FrmPrincipal.Perfil := Usuario.CarregaPerfilUsuario;
```

```
          FrmPrincipal.Login := False;
```

```
          FrmPrincipal.Enabled := True;
```

```

        FrmLogin.Close;
    end;
end;
end;
end;
Usuario.Destroy;
end;

procedure TFrmLogin.BtnOkSenhaClick(Sender: TObject);
begin
    Usuario := TUsuario.Create;
    Usuario.Flogin := EdtLogin.Text;
    if Usuario.CarregaUsuario then begin
        if Usuario.Fsenha <> EdtSenha.Text then begin
            ShowMessage('Senha atual incorreta!');
            EdSenhatual.Clear;
            EdNovaSenha.Clear;
            EdConfNovaSenha.Clear;
            EdSenhatual.SetFocus;
        end else begin
            if EdNovaSenha.Text <> EdConfNovaSenha.Text then begin
                ShowMessage('Nova senha e confirmação não conferem!');
                EdNovaSenha.Clear;
                EdConfNovaSenha.Clear;
                EdNovaSenha.SetFocus;
            end else begin
                Usuario.Fsenha := EdNovasenha.Text;
                if Usuario.AlterarSenha then begin
                    ShowMessage('Senha alterada com sucesso!');
                    FrmPrincipal.Perfil := Usuario.CarregaPerfilUsuario;
                    FrmPrincipal.Login := False;
                    FrmPrincipal.Enabled := True;
                    FrmLogin.Close;
                end else begin
                    DMFireBird.MensagemErroBanco;
                end;
            end;
        end;
    end;
    Usuario.Destroy;
end;

procedure TFrmLogin.BtnAlterarSenhaClick(Sender: TObject);
begin
    vAS := True;
    BtnOK.Click;
end;

```

```
procedure TFrmLogin.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    FrmLogin := nil;
    Action := CaFree;
end;
```

```
procedure TFrmLogin.EdSenhatualExit(Sender: TObject);
begin
    if EdSenhatual.Text = " then begin
        ShowMessage('Informe a sua senha atual!');
        EdSenhatual.SetFocus;
    end;
end;
```

```
procedure TFrmLogin.EdNovasenhaExit(Sender: TObject);
begin
    if EdNovasenha.Text = " then begin
        ShowMessage('Informe a sua nova senha!');
        EdNovasenha.SetFocus;
    end;
end;
```

```
procedure TFrmLogin.EdConfNovaSenhaExit(Sender: TObject);
begin
    if EdConfNovaSenha.Text = " then begin
        ShowMessage('Confirme a sua nova senha!');
        EdConfNovaSenha.SetFocus;
    end;
end;
```

```
procedure TFrmLogin.BtnOKEnter(Sender: TObject);
begin
    if EdtLogin.Text = " then begin
        ShowMessage('Informe seu Login!');
        EdtLogin.SetFocus;
    end;
    if EdtSenha.Text = " then begin
        ShowMessage('Informe sua Senha!');
        EdtSenha.SetFocus;
    end;
end;
```

```
procedure TFrmLogin.BtnAlterarSenhaEnter(Sender: TObject);
begin
    if EdtLogin.Text = " then begin
        ShowMessage('Informe seu Login!');
        EdtLogin.SetFocus;
    end;
```

```
if EdtSenha.Text = " then begin
  ShowMessage('Informe sua Senha!');
  EdtSenha.SetFocus;
end;
end;

end.
```

```
unit UPesquisaCampo;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, StdCtrls, Buttons, ExtCtrls, Grids, DBGrids, DBTables, DB;
```

```
type
```

```
TFrmPesquisaCampos = class(TForm)
```

```
Panel1: TPanel;
```

```
BtnSair: TSpeedButton;
```

```
Panel2: TPanel;
```

```
DbCampos: TDBGrid;
```

```
RgTipoPesquisa: TRadioGroup;
```

```
EdPesquisa: TEdit;
```

```
RgTipo: TRadioGroup;
```

```
procedure BtnSairClick(Sender: TObject);
```

```
procedure FormClose(Sender: TObject; var Action: TCloseAction);
```

```
procedure FormCreate(Sender: TObject);
```

```
procedure EdPesquisaChange(Sender: TObject);
```

```
procedure RgTipoPesquisaClick(Sender: TObject);
```

```
procedure RgTipoClick(Sender: TObject);
```

```
procedure DbCamposDbClick(Sender: TObject);
```

```
private
```

```
{ Private declarations }
```

```
public
```

```
{ Public declarations }
```

```
end;
```

```
var
```

```
FrmPesquisaCampos: TFrmPesquisaCampos;
```

```
implementation
```

```
uses UDMFireBird, UCadCampos;
```

```
{ $R *.dfm }
```

```
procedure TFrmPesquisaCampos.BtnSairClick(Sender: TObject);
```

```
begin
```

```
Close;
```

```
end;
```

```
procedure TFrmPesquisaCampos.FormClose(Sender: TObject;
```

```
var Action: TCloseAction);
```

```
begin
```

```

FrmCadCampos.Enabled := True;
FrmPesquisaCampos := nil;
Action := caFree;
end;

```

```

procedure TFrmPesquisaCampos.FormCreate(Sender: TObject);
begin
  DMFireBird.QryPesqCampo.SQL.Text := 'SELECT * FROM CAMPO ORDER BY
CAMPO';
  DMFireBird.QryPesqCampo.Open;
end;

```

```

procedure TFrmPesquisaCampos.EdPesquisaChange(Sender: TObject);
begin
  DMFireBird.QryPesqCampo.Close;
  DMFireBird.QryPesqCampo.SQL.Text := 'SELECT * FROM CAMPO';
  if RgTipoPesquisa.ItemIndex = 0 then begin
    DMFireBird.QryPesqCampo.SQL.Text := DMFireBird.QryPesqCampo.SQL.Text + '
WHERE CAMPO.campo LIKE :Campo';
    DMFireBird.QryPesqCampo.ParamByName('Campo').Value := '%' + EdPesquisa.Text +
'%';
  end;
  DMFireBird.QryPesqCampo.Open;
end;

```

```

procedure TFrmPesquisaCampos.RgTipoPesquisaClick(Sender: TObject);
begin
  if RgTipoPesquisa.ItemIndex = 1 then begin
    EdPesquisa.Visible := False;
    RgTipo.Visible := True;
  end
  else begin
    EdPesquisa.Visible := True;
    RgTipo.ItemIndex := -1;
    RgTipo.Visible := False;
  end;
  DMFireBird.QryPesqCampo.Close;
  DMFireBird.QryPesqCampo.SQL.Text := 'SELECT * FROM CAMPO'+
' ORDER BY CAMPO';
  DMFireBird.QryPesqCampo.Open;
end;

```

```

procedure TFrmPesquisaCampos.RgTipoClick(Sender: TObject);
begin
  DMFireBird.QryPesqCampo.Close;
  DMFireBird.QryPesqCampo.SQL.Text := 'SELECT * FROM CAMPO';
  if RgTipo.ItemIndex = 0 then begin

```

```

        DMFireBird.QryPesqCampo.SQL.Text := DMFireBird.QryPesqCampo.SQL.Text + '
WHERE CAMPO.TIPO = :Tipo';
        DMFireBird.QryPesqCampo.ParamByName('Tipo').Value := 'Alfanum.';
        end
    else begin
        DMFireBird.QryPesqCampo.SQL.Text := DMFireBird.QryPesqCampo.SQL.Text + '
WHERE CAMPO.TIPO = :Tipo';
        DMFireBird.QryPesqCampo.ParamByName('Tipo').Value := 'Numérico';
        end;
        DMFireBird.QryPesqCampo.Open;

end;

```

```

procedure TFrmPesquisaCampos.DbCamposDbClick(Sender: TObject);
begin
    FrmCadCampos.Enabled := True;
    if assigned(FrmCadCampos) and (FrmCadCampos.BtnNovo.Enabled) and
DMFireBird.QryPesqCampo.FieldByName('Tag').Value <> null then begin
        FrmCadCampos.EdNome.Text :=
DMFireBird.QryPesqCampo.FieldByName('Campo').Value;
        FrmCadCampos.EdCodigo.Text :=
DMFireBird.QryPesqCampo.FieldByName('Tag').Value;
        FrmCadCampos.EdTamMax.Text :=
IntToStr(DMFireBird.QryPesqCampo.FieldByName('TAMANHO').Value);
        if DMFireBird.QryPesqCampo.FieldByName('TIPO').Value = 'Numérico' then
            FrmCadCampos.RgTipo.ItemIndex := 0
        else
            FrmCadCampos.RgTipo.ItemIndex := 1;
        FrmCadCampos.BtnEditar.Click;
    end;
    FrmPesquisaCampos.Close;
end;

end.

```

```
unit UPesquisaDoc;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, StdCtrls, ExtCtrls, DBTables, Isis001, Isis32, Grids, Buttons,  
UClasse;
```

```
type
```

```
TFrmPesquisaDoc = class(TForm)
```

```
Panel1: TPanel;
```

```
Label1: TLabel;
```

```
CbTipoDoc: TComboBox;
```

```
RgOrdenar: TRadioGroup;
```

```
Panel2: TPanel;
```

```
BtnSair: TSpeedButton;
```

```
Panel3: TPanel;
```

```
SgDados: TStringGrid;
```

```
GbCritérios: TGroupBox;
```

```
EdtPesquisa: TEdit;
```

```
EdCodigoReg: TEdit;
```

```
BtnPesquisar: TSpeedButton;
```

```
BtnLimparCritérios: TSpeedButton;
```

```
BtnEditar: TSpeedButton;
```

```
BtnDigitalizar: TSpeedButton;
```

```
RgTipo: TRadioGroup;
```

```
RgContem: TRadioGroup;
```

```
procedure FormClose(Sender: TObject; var Action: TCloseAction);
```

```
procedure FormCreate(Sender: TObject);
```

```
procedure RgOrdenarClick(Sender: TObject);
```

```
procedure BtnSairClick(Sender: TObject);
```

```
procedure BtnPesquisarClick(Sender: TObject);
```

```
procedure CbTipoDocChange(Sender: TObject);
```

```
procedure BtnLimparCritériosClick(Sender: TObject);
```

```
procedure SgDadosDbClick(Sender: TObject);
```

```
procedure SgDadosSelectCell(Sender: TObject; ACol, ARow: Integer;  
var CanSelect: Boolean);
```

```
procedure BtnEditarClick(Sender: TObject);
```

```
procedure BtnDigitalizarClick(Sender: TObject);
```

```
procedure EdCodigoRegKeyPress(Sender: TObject; var Key: Char);
```

```
private
```

```
procedure CarregarComboTipoDoc;
```

```
procedure LimparStringGrid;
```

```
procedure MontarGrid;
```

```
procedure Limpar_Campos;
```

```
procedure MontarRegistro;
```

```

    function CodigoDocumento : String;
    function MontarPesquisa : String;
public
    Linha : Integer;
end;

var
    FrmPesquisaDoc: TFrmPesquisaDoc;
    ConteudoDoc : TConteudoDoc;

    QryTemp : TQuery;
    VetCampo : array[1..MAXMFRL] of Integer;
implementation

uses UDMFireBird, UCadDoc, UlmagemDoc, UPrincipal;

{$R *.dfm}

procedure TFrmPesquisaDoc.FormClose(Sender: TObject;
    var Action: TCloseAction);
begin
    if Assigned(FrmImagensDoc) then begin
        FrmImagensDoc.Enabled := True;
    end;
    if Assigned(FrmCadDocumentos) then begin
        FrmCadDocumentos.Enabled := True;
    end;
    FrmPesquisaDoc := nil;
    Action := CaFree;
end;

procedure TFrmPesquisaDoc.FormCreate(Sender: TObject);
begin
    CarregarComboTipoDoc;
end;

procedure TFrmPesquisaDoc.CarregarComboTipoDoc;
var
    x : Integer;
    Ord : Smallint;
begin
    for x := CbTipoDoc.Items.Count downto 0 do begin
        CbTipoDoc.Items.Delete(x);
    end;
    Ord := RgOrdenar.ItemIndex;
    TipoDoc := TTipoDoc.Create;
    TipoDoc.NewInstance;
    CbTipoDoc.Items.Add('Selecione...');

```

```

QryTemp := TipoDoc.CarregaTipoDoc(Ord);
if not QryTemp.Eof then begin
  QryTemp.First;
  while not QryTemp.Eof do begin
    CbTipoDoc.Items.Add(QryTemp.FieldName('COD_TIPO').Value + ' - ' +
      QryTemp.FieldName('DESC_TIPO').Value);
    QryTemp.Next;
  end;
  CbTipoDoc.Refresh;
  CbTipoDoc.ItemIndex := 0;
end;
QryTemp.Close;
FreeAndNil(QryTemp);
TipoDoc.Destroy;
end;

```

```

procedure TFrmPesquisaDoc.RgOrdenarClick(Sender: TObject);
begin
  CarregarComboTipoDoc;
  CbTipoDocChange(Self);
end;

```

```

procedure TFrmPesquisaDoc.BtnSairClick(Sender: TObject);
begin
  Close;
end;

```

```

procedure TFrmPesquisaDoc.BtnPesquisarClick(Sender: TObject);
var
  Campo, Ocor : Smallint;
  i : integer;
  TotalRegistro : Integer;
  Area : Pchar;
begin
  ConteudoDoc := TConteudoDoc.Create;
  ConteudoDoc.NewInstance;
  area := StrAlloc(6);
  LimparStringGrid;
  if CbTipoDoc.ItemIndex = 0 then begin
    if EdCodigoReg.Text <> '' then begin
      ConteudoDoc.FMfn := StrToInt(EdCodigoReg.Text);
      Campo := 1;
      Ocor := 1;
      Area := ConteudoDoc.PegarDadosIsis(Campo,Ocor);
      if Area > '0000' then begin
        ConteudoDoc.FDoc.FtipoDoc.Fcod := Area;
        MontarGrid;
        MontarRegistro;
      end;
    end;
  end;
end;

```

```

    SgDados.RowCount := SgDados.RowCount - 1;
end else begin
    MessageDlg('Este registro não existe.', mtError, [mbOK], 0);
end;
end;
end else begin
    ConteudoDoc.Fexp := PChar(MontarPesquisa);
    ConteudoDoc.FDoc.FtipoDoc.Fcod := CodigoDocumento;
    TotalRegistro := ConteudoDoc.QtdeRegistro;
    if TotalRegistro > 0 then begin
        MontarGrid;
        For i:= 1 to TotalRegistro do begin
            if ConteudoDoc.Pesquisasis(i) then begin
                MontarRegistro;
            end;
        end;
        SgDados.RowCount := SgDados.RowCount - 1;
    end;
end;
end;

```

```

procedure TFrmPesquisaDoc.LimparStringGrid;

```

```

var

```

```

    x, y : Integer;

```

```

begin

```

```

    for x := 0 to SgDados.ColCount do begin

```

```

        for y := 0 to SgDados.RowCount do begin

```

```

            SgDados.Cells[x,y] := '';

```

```

        end;

```

```

    end;

```

```

    SgDados.ColCount := 1;

```

```

    SgDados.RowCount := 2;

```

```

    SgDados.FixedRows := 1;

```

```

    SgDados.FixedCols := 0;

```

```

end;

```

```

procedure TFrmPesquisaDoc.MontarGrid;

```

```

var

```

```

    x : Integer;

```

```

begin

```

```

    //Limpando o vetor VetCampo

```

```

    for x := 1 to MAXMFRL do begin

```

```

        VetCampo[x] := 0;

```

```

    end;

```

```

    LimparStringGrid;

```

```

    SgDados.Cells[0,0] := 'Código';

```

```

    QryTemp := ConteudoDoc.FDoc.PegarCampos;

```

```

if not QryTemp.Eof then begin
  QryTemp.First;
  while not QryTemp.Eof do begin
    SgDados.ColCount := SgDados.ColCount + 1;
    SgDados.Cells[SgDados.ColCount - 1,0] := QryTemp.FieldName('Campo').Value;
    x := QryTemp.FieldName('Tag').Value;
    VetCampo[x] := SgDados.ColCount - 1;
    QryTemp.Next;
  end;
end;
QryTemp.Close;
FreeAndNil(QryTemp);
end;

```

```

procedure TFrmPesquisaDoc.CbTipoDocChange(Sender: TObject);
begin
  if CbTipoDoc.ItemIndex = 0 then begin
    EdtPesquisa.Visible := False;
    EdCodigoReg.Visible := True;
    GbCritérios.Caption := 'Código do Documento';
    BtnLimparCritérios.Visible := False;
    RgTipo.Visible := False;
    RgContem.Visible := False;
  end else begin
    EdtPesquisa.Visible := True;
    EdCodigoReg.Visible := False;
    GbCritérios.Caption := 'Critérios Pesquisa';
    BtnLimparCritérios.Visible := True;
    RgTipo.Visible := True;
    RgContem.Visible := True;
  end;
  Limpar_Campos;
end;

```

```

procedure TFrmPesquisaDoc.Limpar_Campos;
begin
  EdtPesquisa.Text := '';
  EdCodigoReg.Text := '';
end;

```

```

procedure TFrmPesquisaDoc.BtnLimparCritériosClick(Sender: TObject);
begin
  Limpar_Campos;
end;

```

```

procedure TFrmPesquisaDoc.MontarRegistro;
var
  Campo, cont : Smallint;

```

```

Area: PChar;
x, TotCampos : Integer;
MesmoCampo : Boolean;
begin
  x := 2;
  cont := 0;
  Campo := 0;
  Area := StrAlloc(8000);
  SgDados.Cells[0,SgDados.RowCount - 1] := IntToStr(ConteudoDoc.FMfn);
  TotCampos := ConteudoDoc.NumCamposReg;
  while x <= TotCampos do begin
    if ConteudoDoc.FtagReg[x] <> 1 then begin
      if Campo <> ConteudoDoc.FtagReg[x] then begin
        Campo := ConteudoDoc.FtagReg[x];
        cont := 1;
        MesmoCampo := False;
      end else begin
        cont := cont + 1;
        MesmoCampo := True;
      end;
      Area := ConteudoDoc.PegarDadosIsis(Campo,Cont);
      if MesmoCampo then begin
        SgDados.Cells[VetCampo[campo],(SgDados.RowCount - 1)] :=
SgDados.Cells[VetCampo[campo],(SgDados.RowCount - 1)] + ' | ' + area;
      end else begin
        SgDados.Cells[VetCampo[campo],(SgDados.RowCount - 1)] := area;
      end;
    end;
    x := x + 1;
  end;
  SgDados.RowCount := SgDados.RowCount + 1;
end;

```

```

function TFrmPesquisaDoc.CodigoDocumento : String;
var
  Posicao : Integer;
begin
  Posicao := Pos('-',CbTipoDoc.Text);
  Result := Copy(CbTipoDoc.Text,1,Posicao-2);
end;

```

```

procedure TFrmPesquisaDoc.SgDadosDbClick(Sender: TObject);
begin
  Linha := SgDados.Row;
  if SgDados.Cells[0,linha] <> " then begin;
    BtnEditar.Visible := True;
    BtnDigitalizar.Visible := True;
  end;

```

end;

```
procedure TFrmPesquisaDoc.SgDadosSelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
begin
  BtnEditar.Visible := False;
  BtnDigitalizar.Visible := False;
end;
```

```
procedure TFrmPesquisaDoc.BtnEditarClick(Sender: TObject);
begin
  if FrmPrincipal.Perfil <> 'U' then begin
    If Assigned(FrmCadDocumentos) then begin
      FrmCadDocumentos.EdtCodigo.Text := SgDados.Cells[0,Linha];
      FrmPesquisaDoc.BtnSair.Click;
      FrmCadDocumentos.BtnAplicar.Click;
      FrmCadDocumentos.BtnEditar.Click;
    end else begin
      FrmPesquisaDoc.Enabled := False;
      FrmCadDocumentos := TFrmCadDocumentos.Create(Self);
      FrmCadDocumentos.BtnAplicar.Click;
      FrmCadDocumentos.BtnEditar.Click;
    end;
  end;
end;
```

```
procedure TFrmPesquisaDoc.BtnDigitalizarClick(Sender: TObject);
begin
  If Assigned(FrmImagensDoc) then begin
    FrmImagensDoc.EdtCodigo.Text := SgDados.Cells[0,Linha];
    FrmPesquisaDoc.BtnSair.Click;
    FrmImagensDoc.BtnAplicar.Click;
  end else begin
    FrmPesquisaDoc.Enabled := False;
    FrmImagensDoc := TFrmImagensDoc.Create(Self);
    FrmImagensDoc.BtnAplicar.Click;
  end;
end;
```

```
function TFrmPesquisaDoc.MontarPesquisa : String;
var
  Tamanho, PosF : Integer;
  Documento, Termo, ParPesq, Palavra : String;
  StrTipo, StrContem : String;
begin
  Documento := CodigoDocumento;
  if RgTipo.ItemIndex = 1 then
    StrTipo := '$'
```

```

else
    StrTipo := "";

if RgContem.ItemIndex = 1 then
    StrContem := '*'
else
    StrContem := '+';

if EdtPesquisa.Text <> " then begin
    Termo := EdtPesquisa.Text;
    Tamanho := Length(Termo);
    PosF := Pos(' ', Termo);
    while (PosF > 0) do begin
        Palavra := Copy(Termo, 1, (PosF - 1));
        if ParPesq = " then begin
            if StrTipo = " then begin
                ParPesq := Palavra;
            end else begin
                ParPesq := Palavra + StrTipo;
            end;
        end else begin
            if StrTipo = " then begin
                ParPesq := ParPesq + Palavra;
            end else begin
                ParPesq := ParPesq + Palavra + StrTipo;
            end;
        end;
        ParPesq := ParPesq + StrContem;
        Termo := Copy(Termo, PosF + 1, Tamanho - PosF);
        Tamanho := Length(Termo);
        PosF := Pos(' ', Termo);
    end;
    if Termo <> " then begin
        if ParPesq = " then begin
            if StrTipo = " then begin
                ParPesq := Termo;
            end else begin
                ParPesq := Termo + StrTipo;
            end;
        end else begin
            if StrTipo = " then begin
                ParPesq := ParPesq + Termo;
            end else begin
                ParPesq := ParPesq + Termo + StrTipo;
            end;
        end;
    end;
    Result := '(' + ParPesq + ')' + Documento + '/'(1) ;

```

```
end else begin
  Result := '$*' + Documento + '/'(1))' ;
end;
end;

procedure TFrmPesquisaDoc.EdCodigoRegKeyPress(Sender: TObject;
  var Key: Char);
begin
  if not (Key in ['0'..'9',Chr(8)]) then begin
    Key:= #0;
  end;
end;

end.
```

```
unit UPrincipal;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, Menus, StdCtrls, Buttons, ComCtrls, ExtCtrls, Isis001, Isis32;
```

```
type
```

```
TFrmPrincipal = class(TForm)  
    MenuPrincipal: TMainMenu;  
    MenuDocumentos: TMenuItem;  
    D1: TMenuItem;  
    CadastrarDocumentos1: TMenuItem;  
    Digitalizar1: TMenuItem;  
    PesquisarDocumentos1: TMenuItem;  
    MenuAdmin: TMenuItem;  
    Configuraes1: TMenuItem;  
    Usurios1: TMenuItem;  
    Help1: TMenuItem;  
    Sair1: TMenuItem;  
    StatusBar1: TStatusBar;  
    Panel1: TPanel;  
    BtnCadCampos: TBitBtn;  
    BtnCadDocumentos: TBitBtn;  
    BtnDigDocumentos: TBitBtn;  
    BtnPesqDocumentos: TBitBtn;  
    BtnTipoDoc: TBitBtn;  
    CadastrarTipoDocumento1: TMenuItem;  
    BtnSair: TBitBtn;  
    procedure Sair1Click(Sender: TObject);  
    procedure FormDestroy(Sender: TObject);  
    procedure FormCreate(Sender: TObject);  
    procedure BtnCadCamposClick(Sender: TObject);  
    procedure D1Click(Sender: TObject);  
    procedure BtnCadDocumentosClick(Sender: TObject);  
    procedure CadastrarDocumentos1Click(Sender: TObject);  
    procedure BtnDigDocumentosClick(Sender: TObject);  
    procedure Usurios1Click(Sender: TObject);  
    procedure Digitalizar1Click(Sender: TObject);  
    procedure PesquisarDocumentos1Click(Sender: TObject);  
    procedure BtnTipoDocClick(Sender: TObject);  
    procedure Configuraes1Click(Sender: TObject);  
    procedure BtnPesqDocumentosClick(Sender: TObject);  
    procedure CadastrarTipoDocumento1Click(Sender: TObject);  
    procedure BtnSairClick(Sender: TObject);  
    procedure FormActivate(Sender: TObject);
```

```

    procedure Help1Click(Sender: TObject);
private
    { Private declarations }
public
    Login : Boolean;
    Perfil : String;
    { Public declarations }
end;

var
    FrmPrincipal: TFrmPrincipal;

implementation

uses UCadCampos, ULogin, USelCampos, UDigitaliza, UUsuarios, UPesquisaDoc,
    UTipoDoc, UCadDoc,
    UConfiguracao, UlimagemDoc, UDMFireBird, USobre;

{$R *.dfm}

procedure TFrmPrincipal.Sair1Click(Sender: TObject);
begin
    BtnSair.Click;
end;

procedure TFrmPrincipal.FormDestroy(Sender: TObject);
begin
    Application.Terminate;
end;

procedure TFrmPrincipal.FormCreate(Sender: TObject);
begin
    Perfil := '';
    Login := True;
    StatusBar1.Panels[1].Text := DateToStr(Date);
end;

procedure TFrmPrincipal.BtnCadCamposClick(Sender: TObject);
begin
    if (FrmPrincipal.Perfil = 'A') or (FrmPrincipal.Perfil = 'C') then begin
        if Assigned(FrmCadCampos) then
            begin
                FrmCadCampos.Show;
                FrmCadCampos.BringToFront;
                FrmCadCampos.WindowState := wsNormal;
            end
        else
            begin

```

```

        FrmCadCampos := TFrmCadCampos.Create(Self);
        FrmCadCampos.Show;
    end;
end else begin
    MessageDlg('Seu perfil não permite acesso à esta transação.',mtInformation,[mbOK],0);
end;
end;

procedure TFrmPrincipal.D1Click(Sender: TObject);
begin
    FrmPrincipal.BtnCadCampos.Click;
end;

procedure TFrmPrincipal.BtnCadDocumentosClick(Sender: TObject);
begin
    if FrmPrincipal.Perfil <> 'U' then begin
        if Assigned(FrmCadDocumentos) then
            begin
                FrmCadDocumentos.Show;
                FrmCadDocumentos.BringToFront;
                FrmCadDocumentos.WindowState := wsNormal;
            end
        else
            begin
                FrmCadDocumentos := TFrmCadDocumentos.Create(Self);
                FrmCadDocumentos.Show;
            end;
        end else begin
            MessageDlg('Seu perfil não permite acesso à esta transação.',mtInformation,[mbOK],0);
        end;
    end;

procedure TFrmPrincipal.CadastrarDocumentos1Click(Sender: TObject);
begin
    FrmPrincipal.BtnCadDocumentos.Click;
end;

procedure TFrmPrincipal.BtnDigDocumentosClick(Sender: TObject);
begin
    if Assigned(FrmImagensDoc) then
        begin
            FrmImagensDoc.Show;
            FrmImagensDoc.BringToFront;
            FrmImagensDoc.WindowState := wsNormal;
        end
    else
        begin
            FrmImagensDoc := TFrmImagensDoc.Create(Self);

```

```

        FrmImagensDoc.Show;
    end;
end;

procedure TFrmPrincipal.Usuarios1Click(Sender: TObject);
begin
    if Perfil = 'A' then begin
        if Assigned(FrmUsuarios) then begin
            FrmUsuarios.Show;
            FrmUsuarios.BringToFront;
            FrmUsuarios.WindowState := wsNormal;
        end else begin
            FrmUsuarios := TFrmUsuarios.Create(Self);
            FrmUsuarios.Show;
        end;
    end else begin
        MessageDlg('Seu perfil não permite acesso à esta transação.',mtInformation,[mbOK],0);
    end;
end;

procedure TFrmPrincipal.Digitalizar1Click(Sender: TObject);
begin
    FrmPrincipal.BtnDigDocumentos.Click;
end;

procedure TFrmPrincipal.PesquisarDocumentos1Click(Sender: TObject);
begin
    BtnPesqDocumentos.Click;
end;

procedure TFrmPrincipal.BtnTipoDocClick(Sender: TObject);
begin
    if (FrmPrincipal.Perfil = 'A') or (FrmPrincipal.Perfil = 'C') then begin
        if Assigned(FrmTipoDoc) then begin
            FrmTipoDoc.Show;
            FrmTipoDoc.BringToFront;
            FrmTipoDoc.WindowState := wsNormal;
        end
        else begin
            FrmTipoDoc := TFrmTipoDoc.Create(Self);
            FrmTipoDoc.Show;
        end;
    end else begin
        MessageDlg('Seu perfil não permite acesso à esta transação.',mtInformation,[mbOK],0);
    end;
end;

procedure TFrmPrincipal.Configuraes1Click(Sender: TObject);

```

```

begin
  if Perfil = 'A' then begin
    if Assigned(FrmConfig) then begin
      FrmConfig.BringToFront;
      FrmConfig.WindowState := wsNormal;
      FrmConfig.Show;
    end else begin
      FrmConfig := TFrmConfig.Create(Self);
      FrmConfig.Show;
    end;
  end else begin
    MessageDlg('Seu perfil não permite acesso à esta transação.', mtInformation, [mbOK], 0);
  end;
end;

procedure TFrmPrincipal.BtnPesqDocumentosClick(Sender: TObject);
begin
  if Assigned(FrmPesquisaDoc) then begin
    FrmPesquisaDoc.Show;
    FrmPesquisaDoc.BringToFront;
    FrmPesquisaDoc.WindowState := wsNormal;
  end else begin
    FrmPesquisaDoc := TFrmPesquisaDoc.Create(Self);
    FrmPesquisaDoc.Show;
  end;
end;

procedure TFrmPrincipal.CadastrarTipoDocumento1Click(Sender: TObject);
begin
  BtnTipoDoc.Click;
end;

procedure TFrmPrincipal.BtnSairClick(Sender: TObject);
begin
  Application.Terminate;
end;

procedure TFrmPrincipal.FormActivate(Sender: TObject);
begin
  if Login then begin
    FrmPrincipal.Enabled := False;
    FrmLogin := TFrmLogin.Create(Self);
    FrmLogin.Show;
    FrmLogin.BringToFront;
  end;
end;

procedure TFrmPrincipal.Help1Click(Sender: TObject);

```

```
begin
  if Assigned(FrmSobre) then begin
    FrmSobre.BringToFront;
    FrmSobre.Show;
    FrmSobre.WindowState := wsNormal;
  end else begin
    FrmSobre := TFrmSobre.Create(Self);
    FrmSobre.Show;
  end;
end;

end.
```

```
unit UTipoDoc;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, Buttons, StdCtrls, Grids, DBGrids, DBTables, ExtCtrls, DB, UClasse;
```

```
type
```

```
TFrmTipoDoc = class(TForm)
```

```
Panel2: TPanel;
```

```
Label1: TLabel;
```

```
Panel1: TPanel;
```

```
BtnNovo: TSpeedButton;
```

```
BtnEditar: TSpeedButton;
```

```
BtnSair: TSpeedButton;
```

```
BtnCancelar: TSpeedButton;
```

```
BtnSalvar: TSpeedButton;
```

```
BtnVincular: TBitBtn;
```

```
EdtDocumento: TEdit;
```

```
DbDocumentos: TDBGrid;
```

```
BtnExcluir: TSpeedButton;
```

```
Label2: TLabel;
```

```
EdtCodigo: TEdit;
```

```
procedure BtnSairClick(Sender: TObject);
```

```
procedure FormClose(Sender: TObject; var Action: TCloseAction);
```

```
procedure DbDocumentosDbClick(Sender: TObject);
```

```
procedure BtnNovoClick(Sender: TObject);
```

```
procedure BtnCancelarClick(Sender: TObject);
```

```
procedure BtnEditarClick(Sender: TObject);
```

```
procedure BtnSalvarClick(Sender: TObject);
```

```
procedure BtnExcluirClick(Sender: TObject);
```

```
procedure BtnVincularClick(Sender: TObject);
```

```
procedure FormCreate(Sender: TObject);
```

```
private
```

```
procedure Habilitar_Desabilitar;
```

```
procedure LimparCampos;
```

```
procedure Atualiza_Grid;
```

```
public
```

```
{ Public declarations }
```

```
end;
```

```
var
```

```
FrmTipoDoc: TFrmTipoDoc;
```

```
TipoDoc : TTipoDoc;
```

```
implementation
```

```
uses UDmFireBird, UVincCampos;
```

```
{ $R *.dfm }
```

```
procedure TFrmTipoDoc.BtnSairClick(Sender: TObject);
begin
  Close;
end;
```

```
procedure TFrmTipoDoc.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  FrmTipoDoc := nil;
  Action := CaFree;
end;
```

```
procedure TFrmTipoDoc.DbDocumentosDbClick(Sender: TObject);
begin
  EdtDocumento.Text := DMFireBird.QryTipoDoc.FieldByName('DESC_TIPO').Value;
  EdtCodigo.Text := DMFireBird.QryTipoDoc.FieldByName('COD_TIPO').Value;
end;
```

```
procedure TFrmTipoDoc.Habilitar_Desabilitar;
begin
  BtnNovo.Enabled := not BtnNovo.Enabled;
  BtnEditar.Enabled := not BtnEditar.Enabled;
  BtnSalvar.Enabled := not BtnSalvar.Enabled;
  BtnExcluir.Enabled := not BtnExcluir.Enabled;
  BtnSair.Enabled := not BtnSair.Enabled;
  BtnCancelar.Enabled := not BtnCancelar.Enabled;
  BtnVincular.Enabled := not BtnVincular.Enabled;
  if BtnSalvar.Tag = 0 then begin
    EdtCodigo.Enabled := not EdtCodigo.Enabled;
  end;
  EdtDocumento.Enabled := not EdtDocumento.Enabled;
  DbDocumentos.Enabled := not DbDocumentos.Enabled;
  if BtnNovo.Enabled = False then begin
    EdtCodigo.Color := clWindow;
    EdtDocumento.Color := clWindow;
  end else begin
    EdtCodigo.Color := clScrollBar;
    EdtDocumento.Color := clScrollBar;
  end;
end;
```

```
procedure TFrmTipoDoc.BtnNovoClick(Sender: TObject);
begin
  LimparCampos;
  Habilitar_Desabilitar;
```

```

EdtCodigo.SetFocus;
BtnSalvar.Tag := 0;
end;

procedure TFrmTipoDoc.BtnCancelarClick(Sender: TObject);
begin
    LimparCampos;
    Habilitar_Desabilitar;
end;

procedure TFrmTipoDoc.LimparCampos;
begin
    EdtCodigo.Text := '';
    EdtDocumento.Text := '';
end;

procedure TFrmTipoDoc.BtnEditarClick(Sender: TObject);
begin
    if (EdtCodigo.Text = '') then begin
        MessageDlg('Não há nenhum registro selecionado.', mtError, [mbOK], 0);
    end else begin
        BtnSalvar.Tag := 1;
        Habilitar_Desabilitar;
        EdtDocumento.SetFocus;
    end;
end;

procedure TFrmTipoDoc.BtnSalvarClick(Sender: TObject);
begin
    if (EdtCodigo.Text <> '') and (EdtDocumento.Text <> '') then begin
        TipoDoc := TTipoDoc.Create;
        TipoDoc.NewInstance;
        TipoDoc.Fcod := EdtCodigo.Text;
        TipoDoc.Fnome := EdtDocumento.Text;
        if BtnSalvar.Tag = 0 then begin
            if TipoDoc.CodTipoDoc then begin
                if DMFireBird.MensagemInclusao then begin
                    if TipoDoc.IncluiTipoDoc then begin
                        DMFireBird.ConfirmacaoInclusao;
                        if MessageDlg('Deseja vincular os campos para este tipo de documento?', mtConfirmation, [mbYes, mbNo], 0) = 6 then begin
                            Habilitar_Desabilitar;
                            BtnVincular.Click;
                        end else begin
                            LimparCampos;
                            EdtCodigo.SetFocus;
                        end;
                    end else begin
                        LimparCampos;
                    end;
                end else begin
                    LimparCampos;
                end;
            end else begin
                LimparCampos;
            end;
        end;
    end;
end;

```

```

        DMFireBird.MensagemErroBanco;
    end;
end;
end else begin
    MessageDlg('Este código já existe.', mtError, [mbOK], 0);
end;
end else begin
    if DMFireBird.MensagemAlteracao then begin
        TipoDoc.Fcod := EdtCodigo.Text;
        if TipoDoc.AlterarTipoDoc then begin
            DMFireBird.ConfirmacaoAlteracao;
            BtnCancelar.Click;
        end else begin
            DMFireBird.MensagemErroBanco;
        end;
    end;
end;
TipoDoc.Destroy;
Atualiza_Grid;
end;
end;

procedure TFrmTipoDoc.Atualiza_Grid;
begin
    DMFireBird.QryTipoDoc.Close;
    DMFireBird.QryTipoDoc.Open;
end;

procedure TFrmTipoDoc.BtnExcluirClick(Sender: TObject);
begin
    if EdtCodigo.Text <> '' then begin
        TipoDoc := TTipoDoc.Create;
        TipoDoc.NewInstance;
        TipoDoc.Fcod := EdtCodigo.Text;
        if TipoDoc.CamposVinculados then begin
            if DMFireBird.MensagemExclusao then begin
                if TipoDoc.ExcluirTipoDoc then begin
                    DMFireBird.ConfirmacaoExclusao;
                    LimparCampos;
                end else begin
                    DMFireBird.MensagemErroBanco;
                end;
            end;
        end else begin
            MessageDlg('Para este tipo de documento há campos vinculados, impossível excluir.', mtInformation, [mbOK], 0);
        end;
        TipoDoc.Destroy;
    end;
end;

```

```
    Atualiza_Grid;
end else begin
    MessageDlg('Não há nenhum registro selecionado para exclusão.',mtError,[mbOK],0);
end;
end;

procedure TFrmTipoDoc.BtnVincularClick(Sender: TObject);
begin
    if EdtCodigo.Text = " " then begin
        MessageDlg('Não há nenhum Documento selecionado para poder vincular os
campos',mtError,[mbOK],0);
    end else begin
        FrmTipoDoc.Enabled := False;
        FrmVincularCampos := TFrmVincularCampos.Create(Self);
        FrmVincularCampos.Show;
    end;
end;

procedure TFrmTipoDoc.FormCreate(Sender: TObject);
begin
    DMFireBird.QryTipoDoc.Active := True;
end;

end.
```

```
unit UUsuarios;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, Grids, DBGrids, StdCtrls, Buttons, ExtCtrls, DB, DBTables, DBXpress,  
Spin, Menus, CheckLst, Mask, UClasse;
```

```
type
```

```
TFrmUsuarios = class(TForm)
```

```
    Panel1: TPanel;
```

```
    Label1: TLabel;
```

```
    Label2: TLabel;
```

```
    Label4: TLabel;
```

```
    EdLogin: TEdit;
```

```
    EdNome: TEdit;
```

```
    EdSetor: TEdit;
```

```
    RgTipoPerfil: TRadioGroup;
```

```
    Panel2: TPanel;
```

```
    DbUsuarios: TDBGrid;
```

```
    Panel3: TPanel;
```

```
    BtnNovo: TSpeedButton;
```

```
    BtnSalvar: TSpeedButton;
```

```
    BtnCancelar: TSpeedButton;
```

```
    BtnSair: TSpeedButton;
```

```
    BtnAlterar: TSpeedButton;
```

```
    BtnExcluir: TSpeedButton;
```

```
    CbStatus: TCheckBox;
```

```
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
```

```
    procedure BtnSairClick(Sender: TObject);
```

```
    procedure BtnNovoClick(Sender: TObject);
```

```
    procedure BtnAlterarClick(Sender: TObject);
```

```
    procedure BtnExcluirClick(Sender: TObject);
```

```
    procedure BtnSalvarClick(Sender: TObject);
```

```
    procedure BtnCancelarClick(Sender: TObject);
```

```
    procedure FormCreate(Sender: TObject);
```

```
    procedure DbUsuariosDbClick(Sender: TObject);
```

```
private
```

```
    procedure LimparCampos;
```

```
    function VerificaCampos : Boolean;
```

```
    procedure AtualizaGrid;
```

```
public
```

```
    { Public declarations }
```

```
end;
```

```
var
```

```
FrmUsuarios: TFrmUsuarios;
Usuario : TUsuario;
```

```
Procedure StatusEnable;
Procedure StatusDisable;
```

implementation

```
uses UDMFireBird, UCadCampos;
```

```
{ $R *.dfm }
```

```
procedure TFrmUsuarios.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  FrmUsuarios := nil;
  Action := CaFree;
end;
```

```
procedure TFrmUsuarios.BtnSairClick(Sender: TObject);
begin
  Close;
end;
```

```
procedure StatusEnable;
begin
  if (FrmUsuarios.EdLogin.Enabled = False) then begin
    begin
      FrmUsuarios.EdLogin.Enabled := True;
      FrmUsuarios.EdNome.Enabled := True;
      FrmUsuarios.EdSetor.Enabled := True;
      FrmUsuarios.RgTipoPerfil.Enabled := True;
      FrmUsuarios.EdLogin.Color := clWindow;
      FrmUsuarios.EdNome.Color := clWindow;
      FrmUsuarios.EdSetor.Color := clWindow;
      FrmUsuarios.CbStatus.Enabled := True;
      FrmUsuarios.BtnAlterar.Enabled := False;
      FrmUsuarios.BtnExcluir.Enabled := False;
      FrmUsuarios.BtnNovo.Enabled := False;
      FrmUsuarios.BtnAlterar.Enabled := False;
      FrmUsuarios.BtnExcluir.Enabled := False;
      FrmUsuarios.BtnSalvar.Enabled := True;
      FrmUsuarios.BtnCancelar.Enabled := True;
      FrmUsuarios.EdLogin.SetFocus;
      FrmUsuarios.DbUsuarios.Enabled := False;
    end;
  end;
end;
```

```

procedure StatusDisable;
begin
  if (FrmUsuarios.EdLogin.Enabled = True) then begin
    FrmUsuarios.EdLogin.Enabled := False;
    FrmUsuarios.EdNome.Enabled := False;
    FrmUsuarios.EdSetor.Enabled := False;
    FrmUsuarios.RgTipoPerfil.Enabled := False;
    FrmUsuarios.CbStatus.Enabled := False;
    FrmUsuarios.EdLogin.Color := clScrollBar;
    FrmUsuarios.EdNome.Color := clScrollBar;
    FrmUsuarios.EdSetor.Color := clScrollBar;
    FrmUsuarios.BtnAlterar.Enabled := True;
    FrmUsuarios.BtnExcluir.Enabled := True;
    FrmUsuarios.BtnNovo.Enabled := True;
    FrmUsuarios.BtnSalvar.Enabled := False;
    FrmUsuarios.BtnCancelar.Enabled := False;
    FrmUsuarios.BtnSalvar.Tag := 0;
    FrmUsuarios.DbUsuarios.Enabled := True;
  end;
end;

```

```

procedure TFrmUsuarios.BtnNovoClick(Sender: TObject);
begin
  LimparCampos;
  StatusEnable;
  BtnSalvar.Tag := 0;
  FrmUsuarios.RgTipoPerfil.ItemIndex := 3;
end;

```

```

procedure TFrmUsuarios.BtnAlterarClick(Sender: TObject);
begin
  if EdLogin.Text <> '' then begin
    StatusEnable;
    BtnSalvar.Tag := 1;
  end else begin
    MessageDlg('Não há nenhum usuário selecionado para edição.', mtError, [mbOK], 0);
  end;
end;

```

```

procedure TFrmUsuarios.BtnExcluirClick(Sender: TObject);
begin
  if EdLogin.Text <> '' then begin
    if EdLogin.Text <> 'admin' then begin
      if DMFireBird.MensagemExclusao then begin
        Usuario := TUsuario.Create;
        Usuario.Flogin := EdLogin.Text;
        if Usuario.ExcluiUsuario then begin

```

```

        DMFireBird.ConfirmacaoExclusao;
    end else begin
        DMFireBird.MensagemErroBanco;
    end;
    Usuario.Destroy;
end;
end else begin
    MessageDlg('Impossível excluir, administrador do sistema.', mtError, [mbOK], 0);
end;
end else begin
    MessageDlg('Não há nenhum usuário selecionado para exclusão.', mtError, [mbOK], 0);
end;
AtualizaGrid;
end;

```

```

procedure TFrmUsuarios.BtnSalvarClick(Sender: TObject);

```

```

begin
    if VerificaCampos then begin
        Usuario := TUsuario.Create;
        Usuario.Flogin := EdLogin.Text;
        Usuario.FNome := EdNome.Text;
        Usuario.Fsetor := EdSetor.Text;
        if RgTipoPerfil.ItemIndex = 0 then begin
            Usuario.Fperfil := 'A';
        end else begin
            if RgTipoPerfil.ItemIndex = 1 then begin
                Usuario.Fperfil := 'C';
            end else begin
                if RgTipoPerfil.ItemIndex = 2 then begin
                    Usuario.Fperfil := 'D';
                end else begin
                    Usuario.Fperfil := 'U';
                end;
            end;
        end;
    end;
    if CbStatus.Checked then begin
        Usuario.Fstatus := 'I';
    end else begin
        Usuario.Fstatus := 'A';
    end;
    if BtnSalvar.Tag = 0 then begin
        if Usuario.VerifLogin then begin
            if DMFireBird.MensagemInclusao then begin
                Usuario.Fsenha := EdLogin.Text;
                if Usuario.IncluiUsuario then begin
                    DMFireBird.ConfirmacaoInclusao;
                    BtnCancelar.Click;
                end else begin

```

```

        DMFireBird.MensagemErroBanco;
    end;
end;
end else begin
    MessageDlg('Login já existe.', mtError, [mbOK], 0);
end;
end else begin
    if DMFireBird.MensagemAlteracao then begin
        if Usuario.AlterarUsuario then begin
            DMFireBird.ConfirmacaoAlteracao;
            BtnCancelar.Click;
        end else begin
            DMFireBird.MensagemErroBanco;
        end;
    end;
end;
end;
Usuario.Destroy;
end else begin
    MessageDlg('Favor preencher todos os campos.', mtInformation, [mbOK], 0);
end;
AtualizaGrid;
end;

```

```

procedure TFrmUsuarios.BtnCancelarClick(Sender: TObject);
begin
    LimparCampos;
    StatusDisable;
end;

```

```

procedure TFrmUsuarios.FormCreate(Sender: TObject);
begin
    DMFireBird.QryUsuario.Active := True;
end;

```

```

procedure TFrmUsuarios.DbUsuariosDbClick(Sender: TObject);
begin
    EdLogin.Text := DMFireBird.QryUsuario.FieldName('Login').Value;
    EdNome.Text := DMFireBird.QryUsuario.FieldName('Nome').Value;
    EdSetor.Text := DMFireBird.QryUsuario.FieldName('Setor').Value;
    if DMFireBird.QryUsuario.FieldName('Perfil').Value = 'A' then
        RgTipoPerfil.ItemIndex := 0
    else
        if DMFireBird.QryUsuario.FieldName('Perfil').Value = 'C' then
            RgTipoPerfil.ItemIndex := 1
        else
            if DMFireBird.QryUsuario.FieldName('Perfil').Value = 'D' then
                RgTipoPerfil.ItemIndex := 2
            else

```

```
    RgTipoPerfil.ItemIndex := 3;

    if DMFireBird.QryUsuario.FieldByName('Status').Value = 'I' then begin
        CbStatus.Checked := True;
    end else begin
        CbStatus.Checked := False;
    end;
end;

procedure TFrmUsuarios.LimparCampos;
begin
    FrmUsuarios.EdLogin.Text := "";
    FrmUsuarios.EdNome.Text := "";
    FrmUsuarios.EdSetor.Text := "";
    FrmUsuarios.RgTipoPerfil.ItemIndex := 3;
end;

function TFrmUsuarios.VerificaCampos : Boolean;
begin
    Result := True;
    if EdLogin.Text = "" then
        Result := False;
    if EdNome.Text = "" then
        Result := False;
    if EdSetor.Text = "" then
        Result := False;
end;

procedure TFrmUsuarios.AtualizaGrid;
begin
    DMFireBird.QryUsuario.Close;
    DMFireBird.QryUsuario.Open;
end;

end.
```

```
unit UVincCampos;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, StdCtrls, DBCtrls, Buttons, ExtCtrls, Grids, DBGrids, DBTables, DB, UClasse;
```

```
type
```

```
TFrmVincularCampos = class(TForm)
```

```
Panel1: TPanel;
```

```
Label1: TLabel;
```

```
EdtDocumento: TEdit;
```

```
Panel2: TPanel;
```

```
Panel3: TPanel;
```

```
BtnSair: TSpeedButton;
```

```
DbCampGerais: TDBGrid;
```

```
DbCampVinc: TDBGrid;
```

```
BtnDir: TBitBtn;
```

```
BtnDirAll: TBitBtn;
```

```
BtnEsq: TBitBtn;
```

```
BtnEsqAll: TBitBtn;
```

```
BtnObrig: TBitBtn;
```

```
BtnNObrig: TBitBtn;
```

```
BtnRepet: TBitBtn;
```

```
BtnNRepet: TBitBtn;
```

```
procedure BtnSairClick(Sender: TObject);
```

```
procedure FormClose(Sender: TObject; var Action: TCloseAction);
```

```
procedure FormCreate(Sender: TObject);
```

```
procedure BtnDirClick(Sender: TObject);
```

```
procedure BtnDirAllClick(Sender: TObject);
```

```
procedure BtnEsqClick(Sender: TObject);
```

```
procedure BtnObrigClick(Sender: TObject);
```

```
procedure BtnNObrigClick(Sender: TObject);
```

```
procedure BtnRepetClick(Sender: TObject);
```

```
procedure BtnNRepetClick(Sender: TObject);
```

```
procedure BtnEsqAllClick(Sender: TObject);
```

```
private
```

```
procedure Atualiza_Campos_Gerais;
```

```
procedure Atualiza_Campos_Vinculados;
```

```
public
```

```
{ Public declarations }
```

```
end;
```

```
var
```

```
FrmVincularCampos: TFrmVincularCampos;
```

```
Documento : TDocumento;
```

implementation

uses UDMFireBird, UTipoDoc;

{ \$R *.dfm }

```
procedure TFrmVincularCampos.BtnSairClick(Sender: TObject);
begin
  Close;
end;
```

```
procedure TFrmVincularCampos.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  FrmTipoDoc.Enabled := True;
  FrmVincularCampos := nil;
  Action := caFree;
end;
```

```
procedure TFrmVincularCampos.Atualiza_Campos_Gerais;
begin
  DMFireBird.QryVincCamposG.Close;
  DMFireBird.QryVincCamposG.SQL.Text := 'SELECT TAG, CAMPO, TAMANHO, TIPO
FROM CAMPO'+
      ' WHERE NOT EXISTS (SELECT * FROM
CAMPO_DOCUMENTO'+
      ' WHERE TAG_CAMPO = CAMPO.TAG AND
CAMPO_DOCUMENTO.COD_DOC=:Codigo)'+
      ' ORDER BY CAMPO.TAG';
  DMFireBird.QryVincCamposG.ParamByName('Codigo').Value :=
FrmTipoDoc.EdtCodigo.Text;
  DMFireBird.QryVincCamposG.Open;
end;
```

```
procedure TFrmVincularCampos.FormCreate(Sender: TObject);
begin
  EdtDocumento.Text := FrmTipoDoc.EdtDocumento.Text;
  Atualiza_Campos_Gerais;

  Atualiza_Campos_Vinculados;
  DbCampVinc.SelectedIndex := -1;
end;
```

```
procedure TFrmVincularCampos.Atualiza_Campos_Vinculados;
begin
  DMFireBird.QryCampVinc.Close;
  DMFireBird.QryCampVinc.SQL.Text := 'SELECT TAG_CAMPO, CAMPO,'+
```

```

        ' OBRIGATORIO, REPETITIVO'+
        ' FROM CAMPO_DOCUMENTO INNER JOIN'+
        ' CAMPO ON TAG = TAG_CAMPO'+
        ' WHERE COD_DOC = :Codigo'+
        ' ORDER BY CAMPO_DOCUMENTO.TAG_CAMPO';
    DMFireBird.QryCampVinc.ParamByName('Codigo').Value :=
    FrmTipoDoc.EdtCodigo.Text;
    DMFireBird.QryCampVinc.Open;
    DbCampVinc.SelectedIndex := -1;
end;

```

```

procedure TFrmVincularCampos.BtnDirClick(Sender: TObject);
var
    Cont : Integer;
begin
    Documento := TDocumento.Create;
    Documento.NewInstance;
    Documento.FtipoDoc.Fcod := FrmTipoDoc.EdtCodigo.Text;
    with DbCampGerais do begin
        for Cont := 0 to Pred(SelectedRows.Count) do begin
            Datasource.DataSet.Bookmark := SelectedRows[Cont];
            Documento.Fcampo.Ftag := DbCampGerais.Fields[Tag].Value;
            if Documento.VincCampos = False then begin
                DMFireBird.MensagemErroBanco;
            end;
        end;
    end;
    Atualiza_Campos_Gerais;
    Atualiza_Campos_Vinculados;
    Documento.Destroy;
end;

```

```

procedure TFrmVincularCampos.BtnDirAllClick(Sender: TObject);
begin
    Documento := TDocumento.Create;
    Documento.NewInstance;
    Documento.FtipoDoc.Fcod := FrmTipoDoc.EdtCodigo.Text;
    if Documento.VincCamposT then begin
        Atualiza_Campos_Gerais;
        Atualiza_Campos_Vinculados;
    end else begin
        DMFireBird.MensagemErroBanco;
    end;
    Documento.Destroy;
end;

```

```

procedure TFrmVincularCampos.BtnEsqClick(Sender: TObject);
var

```

```

    Cont : Integer;
begin
    Documento := TDocumento.Create;
    Documento.NewInstance;
    Documento.FtipoDoc.Fcod := FrmTipoDoc.EdtCodigo.Text;
    with DbCampVinc do begin
        for Cont := 0 to Pred(SelectedRows.Count) do begin
            Datasource.DataSet.Bookmark := SelectedRows[Cont];
            Documento.Fcampo.Ftag := DbCampVinc.Fields[Tag].Value;
            if Documento.DesVincCampos = False then begin
                DMFireBird.MensagemErroBanco;
            end;
        end;
    end;
    Atualiza_Campos_Gerais;
    Atualiza_Campos_Vinculados;
    Documento.Destroy;
end;

```

```

procedure TFrmVincularCampos.BtnObrigClick(Sender: TObject);
var
    Cont : Integer;
begin
    if not DMFireBird.QryCampVinc.Eof then begin
        Documento := TDocumento.Create;
        Documento.NewInstance;
        Documento.FtipoDoc.Fcod := FrmTipoDoc.EdtCodigo.Text;
        Documento.Fid := 0;
        with DbCampVinc do begin
            for Cont := 0 to Pred(SelectedRows.Count) do begin
                DataSource.Datasheet.Bookmark:= SelectedRows[Cont];
                Documento.Fcampo.Ftag := DbCampVinc.Fields[Tag].Value;
                if Documento.CampoObrig = False then begin
                    DMFireBird.MensagemErroBanco;
                end;
            end;
        end;
        Atualiza_Campos_Gerais;
        Atualiza_Campos_Vinculados;
        Documento.Destroy;
    end;
end;

```

```

procedure TFrmVincularCampos.BtnNObrigClick(Sender: TObject);
var
    Cont : Integer;
begin
    if not DMFireBird.QryCampVinc.Eof then begin

```

```

Documento := TDocumento.Create;
Documento.NewInstance;
Documento.FtipoDoc.Fcod := FrmTipoDoc.EdtCodigo.Text;
Documento.Fid := 1;
with DbCampVinc do begin
  for Cont := 0 to Pred(SelectedRows.Count) do begin
    DataSource.Dataset.Bookmark:= SelectedRows[Cont];
    Documento.Fcampo.Ftag := DbCampVinc.Fields[Tag].Value;
    if Documento.CampoObrig = False then begin
      DMFireBird.MensagemErroBanco;
    end;
  end;
end;
Atualiza_Campos_Gerais;
Atualiza_Campos_Vinculados;
Documento.Destroy;
end;
end;

procedure TFrmVincularCampos.BtnRepetClick(Sender: TObject);
var
  Cont : Integer;
begin
  if not DMFireBird.QryCampVinc.Eof then begin
    Documento := TDocumento.Create;
    Documento.NewInstance;
    Documento.FtipoDoc.Fcod := FrmTipoDoc.EdtCodigo.Text;
    Documento.Fid := 0;
    with DbCampVinc do begin
      for Cont := 0 to Pred(SelectedRows.Count) do begin
        DataSource.Dataset.Bookmark:= SelectedRows[Cont];
        Documento.Fcampo.Ftag := DbCampVinc.Fields[Tag].Value;
        if Documento.CampoRepet = False then begin
          DMFireBird.MensagemErroBanco;
        end;
      end;
    end;
  end;
  Atualiza_Campos_Gerais;
  Atualiza_Campos_Vinculados;
  Documento.Destroy;
end;
end;

procedure TFrmVincularCampos.BtnNRepetClick(Sender: TObject);
var
  Cont : Integer;
begin
  if not DMFireBird.QryCampVinc.Eof then begin

```

```

Documento := TDocumento.Create;
Documento.NewInstance;
Documento.FtipoDoc.Fcod := FrmTipoDoc.EdtCodigo.Text;
Documento.Fid := 1;
with DbCampVinc do begin
  for Cont := 0 to Pred(SelectedRows.Count) do begin
    DataSource.Dataset.Bookmark:= SelectedRows[Cont];
    Documento.Fcampo.Ftag := DbCampVinc.Fields[Tag].Value;
    if Documento.CampoRepet = False then begin
      DMFireBird.MensagemErroBanco;
    end;
  end;
end;
Atualiza_Campos_Gerais;
Atualiza_Campos_Vinculados;
Documento.Destroy;
end;
end;

procedure TFrmVincularCampos.BtnEsqAllClick(Sender: TObject);
begin
  Documento := TDocumento.Create;
  Documento.NewInstance;
  Documento.FtipoDoc.Fcod := FrmTipoDoc.EdtCodigo.Text;
  if Documento.DesVincCamposT then begin
    Atualiza_Campos_Gerais;
    Atualiza_Campos_Vinculados;
  end else begin
    DMFireBird.MensagemErroBanco;
  end;
  Documento.Destroy;
end;

end.

```

Bibliotecas Externas Utilizadas

{DELPHI IMPLEMENTATION OF TWAIN INTERFACE}
 {december 2003®, initially created by Gustavo Daud}

{This is my newest contribution for Delphi community, a powerfull}
 {implementation of latest Twain features. As you know, twain is }
 {the most common library to acquire images from most acquisition}
 {devices such as Scanners and Web-Cameras.}

{Twain library is a bit different from other libraries, because}
 {most of the hard work can be done by a a single method. Also it}
 {automatically changes in the application message loop, which is}
 {not a simple task, at least in delphi VCL.}

{It is not 100% sure to to Twain not to be installed in Windows,}
 {as it ships with Windows and later and with most of the }
 {acquisition device drivers (automatically with their installation)}
 {This library dynamically calls the library, avoiding the application}
 {hand when it is not present.}

{Also, as in most of my other components, I included a trigger}
 {to allow the component to work without the heavy delphi VCL}
 {for small final executables. To enable, edit DelphiTwain.inc}

{20/01/2004 - Some updates and bug fixes by Nemeth Peter}

{\$INCLUDE DelphiTwain.inc}

```

unit DelphiTwain;

interface

{Used units}
uses
  Twain, Windows {$IFDEF DONTUSEVCL}, Classes, SysUtils, Graphics{$ENDIF},
  DelphiTwainUtils;

const
  {Name of the Twain library for 32 bits enviroment}
  TWAINLIBRARY = 'TWAIN_32.DLL';
  VIRTUALWIN_CLASSNAME = 'DELPHITWAIN_VIRTUALWINDOW';

const
  {Error codes}
  ERROR_BASE          = 300;
  ERROR_INT16: TW_INT16 = HIGH(TW_INT16);

type
  {From twain}
  TW_STR255 = Twain.TW_STR255;

  {Forward declaration}
  TDelphiTwain = class;

  {Component kinds}
  {$IFDEF DONTUSEVCL} TTwainComponent = TObject;
  {$ELSE} TTwainComponent = TComponent; {$ENDIF}

  {File formats}
  TTwainFormat = (tfTIFF, tfPict, tfBMP, tfXBM, tfJPEG, tfFPX,
    tfTIFFMulti, tfPNG, tfSPIFF, tfEXIF, tfUnknown);
    {Twain units}
  TTwainUnit = (tuInches, tuCentimeters, tuPicas, tuPoints, tuTwips,
    tuPixels, tuUnknown);
  TTwainUnitSet = set of TTwainUnit;
  {Twain pixel flavor}
  TTwainPixelFormat = (tpfChocolate, tpfVanilla, tpfUnknown);
  TTwainPixelFormatSet = set of TTwainPixelFormat;
  {Twain pixel type}
  TTwainPixelFormatType = (tbdBw, tbdGray, tbdRgb, tbdPalette, tbdCmy, tbdCmyk,
    tbdYuv, tbdYuvk, tbdCieXYZ, tbdUnknown);
  TTwainPixelFormatTypeSet = set of TTwainPixelFormatType;
  {Twain bit depth}
  TTwainBitDepth = array of TW_UINT16;
  {Twain resolutions}

```

TTwainResolution = array of Extended;

{Events}

TOnTwainError = procedure(Sender: TObject; const Index: Integer; ErrorCode, Additional: Integer) of object;

TOnTwainAcquire = procedure(Sender: TObject; const Index: Integer; Image: {\$IFDEF DONTUSEVCL}TBitmap{\$ELSE}HBitmap{\$ENDIF}; var Cancel: Boolean) of object;

TOnAcquireProgress = procedure(Sender: TObject; const Index: Integer; const Image: HBitmap; const Current, Total: Integer) of object;

TOnSourceNotify = procedure(Sender: TObject; const Index: Integer) of object;

TOnSourceFileTransfer = procedure(Sender: TObject; const Index: Integer; Filename: TW_STR255; Format: TTwainFormat; var Cancel: Boolean) of object;

{Available twain languages}

TTwainLanguage = ({-1}tlUserLocale, tlDanish, tlDutch, tlInternationalEnglish, tlFrenchCanadian, tlFinnish, tlFrench, tlGerman, tlIcelandic, tlItalian, tlNorwegian, tlPortuguese, tlSpanish, tlSwedish, tlUsEnglish, tlAfrikaans, tlAlbania, tlArabic, tlArabicAlgeria, tlArabicBahrain, {18} tlArabicEgypt, tlArabicIraq, tlArabJordan, tlArabicKuwait, tlArabicLebanon, tlArabicLibya, tlArabicMorocco, tlArabicOman, tlArabicQatar, tlArabicSaudiArabia, tlArabicSyria, tlArabicTunisia, tlArabicUae, tlArabicYemen, tlBasque, tlByelorussian, tlBulgarian, {35} tlCatalan, tlChinese, tlChineseHongkong, tlChinesePeoplesRepublic, tlChineseSingapore, tlChineseSimplified, tlChineseTwain, {42} tlChineseTraditional, tlCroatia, tlCzech, tlDutchBelgian, {46} tlEnglishAustralian, tlEnglishCanadian, tlEnglishIreland, tlEnglishNewZealand, tlEnglishSouthAfrica, tlEnglishUk, {52} tlEstonian, tlFaeroese, tlFarsi, tlFrenchBelgian, tlFrenchLuxembourg, {57} tlFrenchSwiss, tlGermanAustrian, tlGermanLuxembourg, tlGermanLiechtenstein, tlGermanSwiss, tlGreek, tlHebrew, tlHungarian, tlIndonesian, {66} tlItalianSwiss, tlJapanese, tlKorean, tlKoreanJohab, tlLatvian, {71} tlLithuanian, tlNorewegianBokmal, tlNorwegianNynorsk, tlPolish, {75} tlPortugueseBrazil, tlRomanian, tlRussian, tlSerbianLatin, tlSlovak, tlSlovenian, tlSpanishMexican, tlSpanishModern, tlThai, tlTurkish, tlUkrainian, tlAssamese, tlBengali, tlBihari, tlBodo, tlDogri, tlGujarati {92}, tlHarayanvi, tlHindi, tlKannada, tlKashmiri, tlMalayalam, tlMarathi, tlMarwari, tlMeghalayan, tlMizo, tlNaga {102}, tlOrissi, tlPunjabi, tlPushtu, tlSerbianCyrillic, tlSikkimi, tlSwedishFinland, tlTamil, tlTelugu, tlTripuri, tlUrdu, tlVietnamese);

{Twain supported groups}

TTwainGroups = set of (tgControl, tgImage, tgAudio);

{Transfer mode for twain}

TTwainTransferMode = (ttmFile, ttmNative, ttmMemory);

{rect for LAYOUT; npeter 2004.01.12.}

TTwainRect =

```

record
  Left: double;
  Top: double;
  Right: double;
  Bottom: double;
end;

{Object to handle TW_IDENTITY}
TTwainIdentity = class{$IFDEF DONTUSEVCL}(TPersistent){$ENDIF}
private
  {Structure which should be filled}
  Structure: TW_IDENTITY;
  {Owner}
  fOwner: {$IFDEF DONTUSEVCL}TComponent{$ELSE}TObject{$ENDIF};
  {Returns/sets application language property}
  function GetLanguage(): TTwainLanguage;
  procedure SetLanguage(const Value: TTwainLanguage);
  {Returns/sets text values}
  function GetString(const Index: integer): String;
  procedure SetString(const Index: Integer; const Value: String);
  {Returns/sets available groups}
  function GetGroups(): TTwainGroups;
  procedure SetGroups(const Value: TTwainGroups);
protected
  {$IFDEF DONTUSEVCL}function GetOwner(): TPersistent; override;{$ENDIF}
public
  {Object being created}
  {$IFDEF DONTUSEVCL} constructor Create(AOwner: TComponent);
  {$ELSE} constructor Create(AOwner: TObject); {$ENDIF}
  {Copy properties from another TTwainIdentity}
  {$IFDEF DONTUSEVCL} procedure Assign(Source: TObject); {$ELSE}
  procedure Assign(Source: TPersistent); override; {$ENDIF}
published
  {Application major version}
  property MajorVersion: TW_UINT16 read Structure.Version.MajorNum
    write Structure.Version.MajorNum;
  {Application minor version}
  property MinorVersion: TW_UINT16 read Structure.Version.MinorNum
    write Structure.Version.MinorNum;
  {Language}
  property Language: TTwainLanguage read GetLanguage write SetLanguage;
  {Country code}
  property CountryCode: word read Structure.Version.Country write
    Structure.Version.Country;
  {Supported groups}
  property Groups: TTwainGroups read GetGroups write SetGroups;
  {Text values}
  property VersionInfo: String index 0 read GetString write

```

```

    SetString;
property Manufacturer: String index 1 read GetString write
    SetString;
property ProductFamily: String index 2 read GetString write
    SetString;
property ProductName: String index 3 read GetString write
    SetString;
end;

```

```

{Return set for capability retrieving/setting}
TCapabilityRet = (crSuccess, crUnsupported, crBadOperation, crDependencyError,
    crLowMemory, crInvalidState, crInvalidContainer);
{Kinds of capability retrieving}
TRetrieveCap = (rcGet, rcGetCurrent, rcGetDefault, rcReset);
{Capability list type}
TGetCapabilityList = array of string;
TSetCapabilityList = array of pointer;

```

```

{Source object}
TTwainSource = class(TTwainIdentity)
private
    {Holds the item index}
    fIndex: Integer;
    {Transfer mode for the images}
    fTransferMode: TTwainTransferMode;
    {Stores if user interface should be shown}
    fShowUI: Boolean;
    {Stores if the source window is modal}
    fModal: Boolean;
    {Stores if the source is enabled}
    fEnabled: Boolean;
    {Stores if the source is loaded}
    fLoaded: Boolean;
    {Stores the owner}
    fOwner: TDelphiTwain;
    {Used with property SourceManagerLoaded to test if the source manager}
    {is loaded or not.}
    function GetSourceManagerLoaded(): Boolean;
    {Returns a pointer to the application}
    function GetApplInfo(): pTW_IDENTITY;
    {Sets if the source is loaded}
    procedure SetLoaded(const Value: Boolean);
    {Sets if the source is enabled}
    procedure SetEnabled(const Value: Boolean);
    {Returns a pointer to the source pTW_IDENTITY}
    function GetStructure: pTW_IDENTITY;
    {Returns a resolution}
    function GetResolution(Capability: TW_UINT16; var Return: Extended;

```

```

    var Values: TTWainResolution; Mode: TRetrieveCap): TCapabilityRet;
protected
    {Reads a native image}
    procedure ReadNative(Handle: TW_UINT32; var Cancel: Boolean);
    {Reads the file image}
    procedure ReadFile(Name: TW_STR255; Format: TW_UINT16; var Cancel: Boolean);
    {Call event for memory image}
    procedure ReadMemory(Image: HBitmap; var Cancel: Boolean);
protected
    {Prepare image memory transference}
    function PrepareMemXfer(var BitmapHandle: HBitmap;
        var PixelType: TW_INT16): TW_UINT16;
    {Transfer image memory}
    function TransferImageMemory(var ImageHandle: HBitmap;
        PixelType: TW_INT16): TW_UINT16;
    {Returns a pointer to the TW_IDENTITY for the application}
    property AppInfo: pTW_IDENTITY read GetAppInfo;
    {Method to transfer the images}
    procedure TransferImages();
    {Message received in the event loop}
    function ProcessMessage(const Msg: TMsg): Boolean;
    {Returns if the source manager is loaded}
    property SourceManagerLoaded: Boolean read GetSourceManagerLoaded;
    {Source configuration methods}
    {*****}
protected
    {Gets an item and returns it in a string}
    procedure GetItem(var Return: String; ItemType: TW_UINT16; Data: Pointer);
    {Converts from a result to a TCapabilityRec}
    function ResultToCapabilityRec(const Value: TW_UINT16): TCapabilityRet;
    {Sets a capability}
    function SetCapabilityRec(const Capability, ConType: TW_UINT16;
        Data: HGLOBAL): TCapabilityRet;
public
    {Returns a capability strucutre}
    function GetCapabilityRec(const Capability: TW_UINT16;
        var Handle: HGLOBAL; Mode: TRetrieveCap;
        var Container: TW_UINT16): TCapabilityRet;
    {*****}
    {Returns an one value capability}
    function GetOneValue(Capability: TW_UINT16;
        var ItemType: TW_UINT16; var Value: string;
        Mode: TRetrieveCap{$IFDEF DEFAULTPARAM}=rcGet{$ENDIF};
        MemHandle: HGLOBAL{$IFDEF DEFAULTPARAM}=0{$ENDIF}): TCapabilityRet;
    {Returns an range capability}
    function GetRangeValue(Capability: TW_UINT16; var ItemType: TW_UINT16;
        var Min, Max, Step, Default, Current: String;
        MemHandle: HGLOBAL{$IFDEF DEFAULTPARAM}=0{$ENDIF}): TCapabilityRet;

```

```

{Returns an enumeration capability}
function GetEnumerationValue(Capability: TW_UINT16;
  var ItemType: TW_UINT16; var List: TGetCapabilityList; var Current,
  Default: Integer; Mode: TRetrieveCap{$IFDEF DEFAULTPARAM}=rcGet{$ENDIF};
  MemHandle: HGLOBAL{$IFDEF DEFAULTPARAM}=0{$ENDIF}): TCapabilityRet;
{Returns an array capability}
function GetArrayValue(Capability: TW_UINT16; var ItemType: TW_UINT16;
  var List: TGetCapabilityList; MemHandle: HGLOBAL
  {$IFDEF DEFAULTPARAM}=0{$ENDIF}): TCapabilityRet;
{*****}
{Sets an one value capability}
function SetOneValue(Capability: TW_UINT16; ItemType: TW_UINT16;
  Value: Pointer): TCapabilityRet;
{Sets a range capability}
function SetRangeValue(Capability, ItemType: TW_UINT16; Min, Max, Step,
  Current: TW_UINT32): TCapabilityRet;
{Sets an enumeration capability}
function SetEnumerationValue(Capability, ItemType: TW_UINT16;
  CurrentIndex: TW_UINT32; List: TSetCapabilityList): TCapabilityRet;
{Sets an array capability}
function SetArrayValue(Capability, ItemType: TW_UINT16;
  List: TSetCapabilityList): TCapabilityRet;
public
  {Setup file transfer}
  function SetupFileTransfer(Filename: String; Format: TTwainFormat): Boolean;
protected
  {Used with property PendingXfers}
  function GetPendingXfers(): TW_INT16;
public
  {Set source transfer mode}
  function ChangeTransferMode(NewMode: TTwainTransferMode): TCapabilityRet;
  {Returns return status information}
  function GetReturnStatus(): TW_UINT16;
  {Capability setting}
  {Set the number of images that the application wants to receive}
  function SetCapXferCount(Value: SmallInt): TCapabilityRet;
  {Returns the number of images that the source will return}
  function GetCapXferCount(var Return: SmallInt;
    Mode: TRetrieveCap{$IFDEF DEFAULTPARAM}=rcGet{$ENDIF}): TCapabilityRet;
  {Retrieve the unit measure for all quantities}
  function GetlCapUnits(var Return: TTwainUnit;
    var Supported: TTwainUnitSet; Mode: TRetrieveCap
    {$IFDEF DEFAULTPARAM}=rcGet{$ENDIF}): TCapabilityRet;
  {Set the unit measure}
  function SetlCapUnits(Value: TTwainUnit): TCapabilityRet;
  {npeter 2004.01.12 begin}
  function SetImagelayoutFrame(const fLeft,fTop,fRight,
    fBottom: double): TCapabilityRet;

```

```

function SetIndicators(Value: boolean): TCapabilityRet;
{npeter 2004.01.12 end}
{Retrieve the pixel flavor values}
function GetIPixelFlavor(var Return: TTwainPixelFlavor;
    var Supported: TTwainPixelFlavorSet; Mode: TRetrieveCap
    {$IFDEF DEFAULTPARAM}=rcGet{$ENDIF}): TCapabilityRet;
{Set the pixel flavor values}
function SetIPixelFlavor(Value: TTwainPixelFlavor): TCapabilityRet;
{Returns bitdepth values}
function GetIBitDepth(var Return: Word;
    var Supported: TTwainBitDepth; Mode: TRetrieveCap
    {$IFDEF DEFAULTPARAM}=rcGet{$ENDIF}): TCapabilityRet;
{Set current bitdepth value}
function SetIBitDepth(Value: Word): TCapabilityRet;
{Returns pixel type values}
function GetIPixelType(var Return: TTwainPixelType;
    var Supported: TTwainPixelTypeSet; Mode: TRetrieveCap
    {$IFDEF DEFAULTPARAM}=rcGet{$ENDIF}): TCapabilityRet;
{Set the pixel type value}
function SetIPixelType(Value: TTwainPixelType): TCapabilityRet;
{Returns X and Y resolutions}
function GetIXResolution(var Return: Extended; var Values: TTwainResolution;
    Mode: TRetrieveCap {$IFDEF DEFAULTPARAM}=rcGet{$ENDIF}): TCapabilityRet;
function GetIYResolution(var Return: Extended; var Values: TTwainResolution;
    Mode: TRetrieveCap {$IFDEF DEFAULTPARAM}=rcGet{$ENDIF}): TCapabilityRet;
{Sets X and X resolutions}
function SetIXResolution(Value: Extended): TCapabilityRet;
function SetIYResolution(Value: Extended): TCapabilityRet;
{Returns physical width and height}
function GetIPhysicalWidth(var Return: Extended; Mode: TRetrieveCap
    {$IFDEF DEFAULTPARAM}=rcGet{$ENDIF}): TCapabilityRet;
function GetIPhysicalHeight(var Return: Extended; Mode: TRetrieveCap
    {$IFDEF DEFAULTPARAM}=rcGet{$ENDIF}): TCapabilityRet;
{Returns if user interface is controllable}
function GetUIControllable(var Return: Boolean): TCapabilityRet;
{Returns feeder is loaded or not}
function GetFeederLoaded(var Return: Boolean): TCapabilityRet;
{Returns/sets if feeder is enabled}
function GetFeederEnabled(var Return: Boolean): TCapabilityRet;
function SetFeederEnabled(Value: WordBool): TCapabilityRet;
{Returns/sets if auto feed is enabled}
function GetAutofeed(var Return: Boolean): TCapabilityRet;
function SetAutoFeed(Value: WordBool): TCapabilityRet;
{Returns number of pending transfer}
property PendingXfers: TW_INT16 read GetPendingXfers;
public
{Enables the source}
function EnableSource(ShowUI, Modal: Boolean): Boolean;

```

```

{Disables the source}
function DisableSource: Boolean;
{Loads the source}
function LoadSource(): Boolean;
{Unloads the source}
function UnloadSource(): Boolean;
{Returns a pointer to the source identity}
property SourceIdentity: pTW_IDENTITY read GetStructure;
{Returns/sets if the source is enabled}
property Enabled: Boolean read fEnabled write SetEnabled;
{Returns/sets if this source is loaded}
property Loaded: Boolean read fLoaded write SetLoaded;
{Object being created/destroyed}
constructor Create(AOwner: TDelphiTwain);
destructor Destroy; override;
{Returns owner}
property Owner: TDelphiTwain read fOwner;
{Source window is modal}
property Modal: Boolean read fModal write fModal;
{Sets if user interface should be shown}
property ShowUI: Boolean read fShowUI write fShowUI;
{Transfer mode for transferring images from the source to}
{the component and finally to the application}
property TransferMode: TTwainTransferMode read fTransferMode
    write fTransferMode;
{Returns the item index}
property Index: Integer read fIndex;
{Convert properties from write/read to read only}
{(read description on TTwainIdentity source)}
property MajorVersion: TW_UINT16 read Structure.Version.MajorNum;
property MinorVersion: TW_UINT16 read Structure.Version.MinorNum;
property Language: TTwainLanguage read GetLanguage;
property CountryCode: word read Structure.Version.Country;
property Groups: TTwainGroups read GetGroups;
property VersionInfo: String index 0 read GetString;
property Manufacturer: String index 1 read GetString;
property ProductFamily: String index 2 read GetString;
property ProductName: String index 3 read GetString;
end;

{Component part}
TDelphiTwain = class(TTwainComponent)
private
    {Should contain the number of Twain sources loaded}
    fSourcesLoaded: Integer;
    {Contains if the select source dialog is being displayed}
    SelectDialogDisplayed: Boolean;
private

```

```

{Event pointer holders}
fOnSourceDisable: TOnSourceNotify;
fOnAcquireCancel: TOnSourceNotify;
fOnTwainAcquire: TOnTwainAcquire;
fOnSourceSetupFileXfer: TOnSourceNotify;
fOnSourceFileTransfer: TOnSourceFileTransfer;
fOnAcquireError: TOnTwainError;
fOnAcquireProgress: TOnAcquireProgress;
private
{Temp variable to allow SourceCount to be displayed in delphi}
{property editor}
fDummySourceCount: integer;
{Contains list of source devices}
DeviceList: TPointerList;
{Contains a pointer to the structure with the application}
{information}
AppInfo: pTW_IDENTITY;
{Holds the object to allow the user to set the application information}
fInfo: TTwainIdentity;
{Holds the handle for the virtual window which will receive}
{twain message notifications}
VirtualWindow: THandle;
{Will hold Twain library handle}
fHandle: HInst;
{Holds if the component has enumerated the devices}
fHasEnumerated: Boolean;
{Holds twain dll procedure handle}
fTwainProc: TDSMEntryProc;
{Holds the transfer mode to be used}
fTransferMode: TTwainTransferMode;
{Contains if the library is loaded}
fLibraryLoaded: Boolean;
{Contains if the source manager was loaded}
fSourceManagerLoaded: Boolean;
{Procedure to load and unload twain library and update property}
procedure SetLibraryLoaded(const Value: Boolean);
{Procedure to load or unloaded the twain source manager}
procedure SetSourceManagerLoaded(const Value: Boolean);
{Updates the application information object}
procedure SetInfo(const Value: TTwainIdentity);
{Returns the number of sources}
function GetSourceCount(): Integer;
{Returns a source from the list}
function GetSource(Index: Integer): TTwainSource;
{Finds a matching source index}
function FindSource(Value: pTW_IDENTITY): Integer;
protected
{Returns the default source}

```

```

function GetDefaultSource: Integer;
{Creates the virtual window}
procedure CreateVirtualWindow();
{Clears the list of sources}
procedure ClearDeviceList();
public
{Allows Twain to display a dialog to let the user choose any source}
{and returns the source index in the list}
function SelectSource(): Integer;
{Returns the number of loaded sources}
property SourcesLoaded: Integer read fSourcesLoaded;
{Enumerate the available devices after Source Manager is loaded}
function EnumerateDevices(): Boolean;
{Object being created}
{$IFDEF DONTUSEVCL}
    constructor Create(AOwner: TComponent);override;
{$ELSE}
    constructor Create;
{$ENDIF}
{Object being destroyed}
destructor Destroy(); override;
{Loads twain library and returns if it loaded successfully}
function LoadLibrary(): Boolean;
{Unloads twain and returns if it unloaded successfully}
function UnloadLibrary(): Boolean;
{Loads twain source manager}
function LoadSourceManager(): Boolean;
{Unloads the source manager}
function UnloadSourceManager(forced: boolean): Boolean;
{Returns the application TW_IDENTITY}
property AppIdentity: pTW_IDENTITY read AppInfo;
{Returns Twain library handle}
property Handle: HInst read fHandle;
{Returns a pointer to Twain only procedure}
property TwainProc: TDSMEntryProc read fTwainProc;
{Holds if the component has enumerated the devices}
property HasEnumerated: Boolean read fHasEnumerated;
{Returns a source}
property Source[Index: Integer]: TTwainSource read GetSource;
published
{Events}
{Source being disabled}
property OnSourceDisable: TOnSourceNotify read fOnSourceDisable
    write fOnSourceDisable;
{Acquire cancelled}
property OnAcquireCancel: TOnSourceNotify read fOnAcquireCancel
    write fOnAcquireCancel;
{Image acquired}

```

```

property OnTwainAcquire: TOnTwainAcquire read fOnTwainAcquire
  write fOnTwainAcquire;
{User should set information to prepare for the file transfer}
property OnSourceSetupFileXfer: TOnSourceNotify read fOnSourceSetupFileXfer
  write fOnSourceSetupFileXfer;
{File transfered}
property OnSourceFileTransfer: TOnSourceFileTransfer read
  fOnSourceFileTransfer write fOnSourceFileTransfer;
{Acquire error}
property OnAcquireError: TOnTwainError read fOnAcquireError
  write fOnAcquireError;
{Acquire progress, for memory transfers}
property OnAcquireProgress: TOnAcquireProgress read fOnAcquireProgress
  write fOnAcquireProgress;
published
{Default transfer mode to be used with sources}
property TransferMode: TTwainTransferMode read fTransferMode
  write fTransferMode;
{Returns the number of sources, after Library and Source Manager}
{has being loaded}
property SourceCount: Integer read GetSourceCount write fDummySourceCount;
{User should fill the application information}
property Info: TTwainIdentity read fInfo write SetInfo;
{Loads or unload Twain library}
property LibraryLoaded: Boolean read fLibraryLoaded write SetLibraryLoaded;
{Loads or unloads the source manager}
property SourceManagerLoaded: Boolean read fSourceManagerLoaded write
  SetSourceManagerLoaded;
end;

{Puts a string inside a TW_STR255}
function StrToStr255(Value: String): TW_STR255;
{This method returns if Twain is installed in the current machine}
function IsTwainInstalled(): Boolean;
{Called by Delphi to register the component}
procedure Register();
{Returns the size of a twain type}
function TWTypeSize(TypeName: TW_UINT16): Integer;

implementation

{Units used bellow}
uses
  Messages;

{Called by Delphi to register the component}
procedure Register();
begin

```

```

{$IFDEF DONTUSEVCL}
  RegisterComponents('NP', [TDelphiTwain]);
{$ENDIF}
end;

```

{Returns the size of a twain type}

```

function TWTypeSize(TypeName: TW_UINT16): Integer;
begin

```

```

  {Test the type to return the size}

```

```

  case TypeName of

```

```

    TWTY_INT8 : Result := sizeof(TW_INT8);

```

```

    TWTY_UINT8 : Result := sizeof(TW_UINT8);

```

```

    TWTY_INT16 : Result := sizeof(TW_INT16);

```

```

    TWTY_UINT16: Result := sizeof(TW_UINT16);

```

```

    TWTY_INT32 : Result := sizeof(TW_INT32);

```

```

    TWTY_UINT32: Result := sizeof(TW_UINT32);

```

```

    TWTY_FIX32 : Result := sizeof(TW_FIX32);

```

```

    TWTY_FRAME : Result := sizeof(TW_FRAME);

```

```

    TWTY_STR32 : Result := sizeof(TW_STR32);

```

```

    TWTY_STR64 : Result := sizeof(TW_STR64);

```

```

    TWTY_STR128: Result := sizeof(TW_STR128);

```

```

    TWTY_STR255: Result := sizeof(TW_STR255);

```

```

    //npeter: the following types were not implemented

```

```

    //especially the bool caused problems

```

```

    TWTY_BOOL:   Result := sizeof(TW_BOOL);

```

```

    TWTY_UNI512: Result := sizeof(TW_UNI512);

```

```

    TWTY_STR1024: Result := sizeof(TW_STR1024);

```

```

    else      Result := 0;

```

```

  end {case}

```

```

end;

```

{Puts a string inside a TW_STR255}

```

function StrToStr255(Value: String): TW_STR255;

```

```

begin

```

```

  {Clean result}

```

```

  Fillchar(Result, sizeof(TW_STR255), #0);

```

```

  {If value fits inside the TW_STR255, copy memory}

```

```

  if Length(Value) <= sizeof(TW_STR255) then

```

```

    CopyMemory(@Result[0], @Value[1], Length(Value))

```

```

  else CopyMemory(@Result[0], @Value[1], sizeof(TW_STR255));

```

```

end;

```

{Returns full Twain directory (usually in Windows directory)}

```

function GetTwainDirectory(): String;

```

```

var

```

```

  i: TDirectoryKind;

```

```

  Dir: String;

```

```

begin

```

```

{Searches in all the directories}
FOR i := LOW(TDirectoryKind) TO HIGH(TDirectoryKind) DO
begin

    {Directory to search}
    Dir := GetCustomDirectory(i);
    {Tests if the file exists in this directory}
    if FileExists(Dir + TWAINLIBRARY) then
    begin
        {In case it exists, returns this directory and exit}
        {the for loop}
        Result := Dir;
        Break;
    end {if FileExists}

end {FOR i}
end;

{This method returns if Twain is installed in the current machine}
function IsTwainInstalled(): Boolean;
begin
    {If GetTwainDirectory function returns an empty string, it means}
    {that Twain was not found}
    Result := (GetTwainDirectory() <> "");
end;

{ TTwainIdentity object implementation }

{Object being created}
{$IFDEF DONTUSEVCL} constructor TTwainIdentity.Create(AOwner: TComponent);
{$ELSE} constructor TTwainIdentity.Create(AOwner: TObject); {$ENDIF}
begin
    {Allows ancestor to work}
    inherited Create;

    {Set initial properties}
    FillChar(Structure, sizeof(Structure), #0);
    Language := tlUserLocale;
    CountryCode := 1;
    MajorVersion := 1;
    VersionInfo := 'Application name';
    Structure.ProtocolMajor := TWON_PROTOCOLMAJOR;
    Structure.ProtocolMinor := TWON_PROTOCOLMINOR;
    Groups := [tgImage, tgControl];
    Manufacturer := 'Application manufacturer';
    ProductFamily := 'App product family';
    ProductName := 'App product name';

```

```

    fOwner := AOwner; {Copy owner pointer}
end;

{$IFDEF DONTUSEVCL}
function TTwainIdentity.GetOwner(): TPersistent;
begin
    Result := fOwner;
end;
{$ENDIF}

{Sets a text value}
procedure TTwainIdentity.SetString(const Index: Integer;
    const Value: String);
var
    PropStr: PChar;
begin
    {Select and copy pointer}
    case Index of
        0: PropStr := @Structure.Version.Info[0];
        1: PropStr := @Structure.Manufacturer[0];
        2: PropStr := @Structure.ProductFamily[0];
    else PropStr := @Structure.ProductName[0];
    end {case};

    {Set value}
    Fillchar(PropStr^, sizeof(TW_STR32), #0);
    if Length(Value) > sizeof(TW_STR32) then
        CopyMemory(PropStr, @Value[1], sizeof(TW_STR32))
    else
        CopyMemory(PropStr, @Value[1], Length(Value));
    end;

{Returns a text value}
function TTwainIdentity.GetString(const Index: Integer): String;
begin
    {Test for the required property}
    case Index of
        0: Result := Structure.Version.Info;
        1: Result := Structure.Manufacturer;
        2: Result := Structure.ProductFamily;
    else Result := Structure.ProductName;
    end {case}
end;

{Returns application language property}
function TTwainIdentity.GetLanguage(): TTwainLanguage;
begin
    Result := TTwainLanguage(Structure.Version.Language + 1);
end;

```

```

end;

{Sets application language property}
procedure TTwainIdentity.SetLanguage(const Value: TTwainLanguage);
begin
  Structure.Version.Language := Word(Value) - 1;
end;

{Copy properties from another TTwainIdentity}
{$IFDEF DONTUSEVCL} procedure TTwainIdentity.Assign(Source: TObject);
{$ELSE} procedure TTwainIdentity.Assign(Source: TPersistent); {$ENDIF}
begin
  {The source should also be a TTwainIdentity}
  if Source is TTwainIdentity then
    {Copy properties}
    Structure := TTwainIdentity(Source).Structure
  else
    {$IFDEF DONTUSEVCL}inherited; {$ENDIF}
  end;

{Returns available groups}
function TTwainIdentity.GetGroups(): TTwainGroups;
begin
  {Convert from Structure.SupportedGroups to TTwainGroups}
  Include(Result, tgControl);
  if DG_IMAGE AND Structure.SupportedGroups <> 0 then
    Include(Result, tgImage);
  if DG_AUDIO AND Structure.SupportedGroups <> 0 then
    Include(Result, tgAudio);
end;

{Sets available groups}
procedure TTwainIdentity.SetGroups(const Value: TTwainGroups);
begin
  {Convert from TTwainGroups to Structure.SupportedGroups}
  Structure.SupportedGroups := DG_CONTROL;
  if tgImage in Value then
    Structure.SupportedGroups := Structure.SupportedGroups or DG_IMAGE;
  if tgAudio in Value then
    Structure.SupportedGroups := Structure.SupportedGroups or DG_AUDIO;
end;

{ TDelphiTwain component implementation }

{Loads twain library and returns if it loaded sucessfully}
function TDelphiTwain.LoadLibrary(): Boolean;
var
  TwainDirectory: String;

```

```

begin
  {The library must not be already loaded}
  if (not LibraryLoaded) then
    begin
      Result := FALSE; {Initially returns FALSE}
      {Searches for Twain directory}
      TwainDirectory := GetTwainDirectory();
      {Continue only if twain is installed in an known directory}
      if TwainDirectory <> "" then
        begin

          fHandle := Windows.LoadLibrary(PChar(TwainDirectory + TWAINLIBRARY));
          {If the library was sucessfully loaded}
          if (fHandle <> INVALID_HANDLE_VALUE) then
            begin

              {Obtains method handle}
              @fTwainProc := GetProcAddress(fHandle, MAKEINTRESOURCE(1));
              {Returns TRUE/FALSE if the method was obtained}
              Result := (@fTwainProc <> nil);

              {If the method was not obtained, also free the library}
              if not Result then
                begin
                  {Free the handle and clears the variable}
                  Windows.FreeLibrary(fHandle);
                  fHandle := 0;
                end {if not Result}
            end
          else
            {If it was not loaded, clears handle value}
            fHandle := 0;

          end {if TwainDirectory <> ""};

        end
      else
        {If it was already loaded, returns true, since that is}
        {what was supposed to happen}
        Result := TRUE;

        {In case the method was sucessful, updates property}
        if Result then fLibraryLoaded := TRUE;
      end;

      {Unloads twain and returns if it unloaded sucessfully}
    end
  function TDelphiTwain.UnloadLibrary(): Boolean;

```

```

begin
  {The library must not be already unloaded}
  if (LibraryLoaded) then
    begin
      {Unloads the source manager}
      SourceManagerLoaded := FALSE;
      {Just call windows method to unload}
      Result := Windows.FreeLibrary(Handle);
      {If it was sucessfull, also clears handle value}
      if Result then fHandle := 0;
      {Updates property}
      fLibraryLoaded := not Result;
    end
  else
    {If it was already unloaded, returns true, since that is}
    {what was supposed to happen}
    Result := TRUE;

    {In case the method was sucessful, updates property}
    if Result then fLibraryLoaded := FALSE;
  end;

  {Enumerate the avaiable devices after Source Manager is loaded}
  function TDelphiTwain.EnumerateDevices(): Boolean;
  var
    NewSource: TTwainSource;
    CallRes : TW_UINT16;
  begin
    {Booth library and source manager must be loaded}
    if (LibraryLoaded and SourceManagerLoaded) then
      begin
        {Clears the preview list of sources}
        ClearDeviceList();

        {Allocate new identity and tries to enumerate}
        NewSource := TTwainSource.Create(Self);
        CallRes := TwainProc(AppInfo, nil, DG_CONTROL, DAT_IDENTITY,
          MSG_GETFIRST, @NewSource.Structure);
        if CallRes = TWRC_SUCCESS then
          repeat

            {Add this item to the list}
            DeviceList.Add(NewSource);
            {Allocate memory for the next}
            NewSource := TTwainSource.Create(Self);
            NewSource.TransferMode := Self.TransferMode;
            NewSource.fIndex := DeviceList.Count;

```

```

    {Try to get the next item}
    until TwainProc(AppInfo, nil, DG_CONTROL, DAT_IDENTITY,
        MSG_GETNEXT, @NewSource.Structure) <> TWRC_SUCCESS;

    {Set that the component has enumerated the devices}
    {if everything went correctly}
    Result := TRUE;
    fHasEnumerated := Result;

    {Dispose un-needed source object}
    NewSource.Free;

end
else Result := FALSE; {If library and source manager aren't loaded}
end;

{Procedure to load and unload twain library and update property}
procedure TDelphiTwain.SetLibraryLoaded(const Value: Boolean);
begin
    {The value must be changing to activate}
    if (Value <> fLibraryLoaded) then
    begin
        {Depending on the parameter load/unload the library and updates}
        {property whenever it loaded or unloaded successfully}
        if Value then LoadLibrary()
        else {if not Value then} UnloadLibrary();

    end {if (Value <> fLibraryLoaded)}
end;

{Loads twain source manager}
function TDelphiTwain.LoadSourceManager(): Boolean;
begin
    {The library must be loaded}
    if LibraryLoaded and not SourceManagerLoaded then
    begin
        {Loads source manager}
        Result := (fTwainProc(AppInfo, nil, DG_CONTROL, DAT_PARENT,
            MSG_OPENDSM, @VirtualWindow) = TWRC_SUCCESS)
    end
    else
    begin
        {The library is not loaded, thus the source manager could}
        {not be loaded}
        Result := FALSE or SourceManagerLoaded;

    end
    {In case the method was successful, updates property}
    if Result then fSourceManagerLoaded := TRUE;
end;

{UnLoads twain source manager}

```

```

function TDelphiTwain.UnloadSourceManager(forced: boolean): Boolean;
begin
  {The library must be loaded}
  if LibraryLoaded and SourceManagerLoaded then
    begin
      {Clears the list of sources}
      ClearDeviceList();
      {Unload source manager}
      if not forced then
        Result := (TwainProc(AppInfo, nil, DG_CONTROL, DAT_PARENT, MSG_CLOSEDM,
@VirtualWindow) = TWRC_SUCCESS)
        else result:=true;
      end
    else
      {The library is not loaded, meaning that the Source Manager isn't either}
      Result := TRUE;

      {In case the method was sucessful, updates property}
      if Result then fSourceManagerLoaded := FALSE;
    end;

{Procedure to load or unloaded the twain source manager}
procedure TDelphiTwain.SetSourceManagerLoaded(const Value: Boolean);
begin
  {The library must be loaded to have access to the method}
  if LibraryLoaded and (Value <> fSourceManagerLoaded) then
    begin
      {Load/unload the source manager}
      if Value      then LoadSourceManager()
      else {if not Value then} UnloadSourceManager(false);
    end {if LibraryLoaded}
  end;

{Clears the list of sources}
procedure TDelphiTwain.ClearDeviceList();
var
  i: Integer;
begin
  {Deallocate pTW_IDENTITY}
  FOR i := 0 TO DeviceList.Count - 1 DO
    TTwainSource(DeviceList.Item[i]).Free;
  {Clears the list}
  DeviceList.Clear;
  {Set trigger to tell that it has not enumerated again yet}
  fHasEnumerated := FALSE;

end;

```

```

{Finds a matching source index}
function TDelphiTwain.FindSource(Value: pTW_IDENTITY): Integer;
var
  i      : Integer;
begin
  Result := -1; {Default result}

  {Search for this source in the list}
  for i := 0 TO SourceCount - 1 DO
    if CompareMem(@Source[i].Structure, pChar(Value), SizeOf(TW_IDENTITY)) then
      begin
        {Return index and exit}
        Result := i;
        break;
      end; {if CompareMem, for i}
  end;

  {Allows Twain to display a dialog to let the user choose any source}
  {and returns the source index in the list}
  function TDelphiTwain.SelectSource: Integer;
  var
    Identity: TW_IDENTITY;
  begin
    Result := -1; {Default result}
    {Booth library and source manager must be loaded}
    if (LibraryLoaded and SourceManagerLoaded and not SelectDialogDisplayed) then
      begin
        {Don't allow this dialog to be displayed twice}
        SelectDialogDisplayed := TRUE;

        {Call twain to display the dialog}
        if TwainProc(AppInfo, nil, DG_CONTROL, DAT_IDENTITY, MSG_USERSELECT,
          @Identity) = TWRC_SUCCESS then
          Result := FindSource(@Identity);

        {Ended using}
        SelectDialogDisplayed := FALSE
      end {(LibraryLoaded and SourceManagerLoaded)}
  end;

  {Returns the number of sources}
  function TDelphiTwain.GetSourceCount(): Integer;
  begin
    {Library and source manager must be loaded}
    if (LibraryLoaded and SourceManagerLoaded) then
      begin
        {Enumerate devices, if needed}
        if not HasEnumerated then EnumerateDevices();

```

```

    {Returns}
    Result := DeviceList.Count;
end
{In case library and source manager aren't loaded, returns 0}
else Result := 0
end;

{Returns the default source}
function TDelphiTwain.GetDefaultSource: Integer;
var
    Identity: TW_IDENTITY;
begin
    {Call twain to display the dialog}
    if SourceManagerLoaded and (TwainProc(AppInfo, nil, DG_CONTROL, DAT_IDENTITY,
        MSG_GETDEFAULT, @Identity) = TWRC_SUCCESS) then
        Result := FindSource(@Identity)
    else Result := 0 {Returns}
end;

{Returns a source from the list}
function TDelphiTwain.GetSource(Index: Integer): TTwainSource;
begin
    {Booth library and source manager must be loaded}
    if (LibraryLoaded and SourceManagerLoaded) then
        begin

            {If index is in range, returns}
            {(Call to SourceCount property enumerates the devices, if needed)}
            if Index in [0..SourceCount - 1] then
                Result := DeviceList.Item[Index]
            else if (Index = -1) and (SourceCount > 0) then
                Result := DeviceList.Item[GetDefaultSource]
            {Unknown object, returns nil}
            else Result := nil;

        end
    {In case either the library or the source manager aren't}
    {loaded, it returns nil}
    else Result := nil
end;

{Object being created}
constructor TDelphiTwain.Create{$IFDEF DONTUSEVCL}(AOwner:
TComponent){$ENDIF};
begin
    {Let the ancestor class also handle the call}
    inherited;

```

```

{Create source list}
DeviceList := TPointerList.Create;
{Clear variables}
fSourcesLoaded := 0;
fHandle := 0;
@fTwainProc := nil;
SelectDialogDisplayed := FALSE;
fSourceManagerLoaded := FALSE;
fHasEnumerated := FALSE;
fTransferMode := ttmMemory;
{Creates the virtual window which will intercept messages}
{from Twain}
CreateVirtualWindow();
{Creates the object to allow the user to set the application}
{information to inform twain source manager and sources}
fInfo := TTwainIdentity.Create(Self);
AppInfo := @fInfo.Structure;
end;

```

```

{Object being destroyed}
destructor TDelphiTwain.Destroy;
begin
  {Full unload the library}
  LibraryLoaded := FALSE;
  {Free the virtual window handle}
  DestroyWindow(VirtualWindow);
  {Free the object}
  fInfo.Free;
  {Clears and free source list}
  ClearDeviceList();
  DeviceList.Free();
  {Let ancestor class handle}
  inherited Destroy;
end;

```

```

{Creates the virtual window}
procedure TDelphiTwain.CreateVirtualWindow;
begin
  {Creates the window and passes a pointer to the class object}
  VirtualWindow := CreateWindow(VIRTUALWIN_CLASSNAME, 'Delphi Twain virtual ' +
    'window', 0, 10, 10, 100, 100, 0, 0, hInstance, Self);
end;

```

```

{Updates the application information object}
procedure TDelphiTwain.SetInfo(const Value: TTwainIdentity);
begin
  {Assign one object to another}
  fInfo.Assign(Value);
end;

```

end;

{ TTwainSource object implementation }

{Used with property SourceManagerLoaded to test if the source manager
{is loaded or not.}}

```
function TTwainSource.GetSourceManagerLoaded: Boolean;
begin
  {Obtain information from owner TDelphiTwain}
  Result := Owner.SourceManagerLoaded;
end;
```

{Sets if the source is loaded}

```
procedure TTwainSource.SetLoaded(const Value: Boolean);
begin
  {Value should be changing}
  if (Value <> fLoaded) then
    begin
      {Loads or unloads the source}
      if Value      then LoadSource()
      else {if not Value then} UnloadSource();
    end {if (Value <> fLoaded)}
end;
```

{Sets if the source is enabled}

```
procedure TTwainSource.SetEnabled(const Value: Boolean);
begin
  {Source must be already enabled and value changing}
  if (Loaded) and (Value <> fEnabled) then
    begin
      {Enables/disables}
      if Value      then EnableSource(ShowUI, Modal)
      else {if not Value then} DisableSource();
    end {if (Loaded) and (Value <> fEnabled)}
end;
```

{Enables the source}

```
function TTwainSource.EnableSource(ShowUI, Modal: Boolean): Boolean;
var
  twUserInterface: TW_USERINTERFACE;
begin
  {Source must be loaded and the value changing}
  if (Loaded) and (not Enabled) then
    begin
      {Builds UserInterface structure}
      twUserInterface.ShowUI := ShowUI;
      twUserInterface.ModalUI := Modal;
      twUserInterface.hParent := owner.VirtualWindow;
```

```

//npeter may be it is better to send messages to VirtualWindow
//I am not sure, but it seems more stable with a HP TWAIN driver
//it was: := GetActiveWindow;
fEnabled := TRUE;
{Call method}
Result := (Owner.TwainProc(AppInfo, @Structure, DG_CONTROL,
    DAT_USERINTERFACE, MSG_ENABLEDS, @twUserInterface) in
    [TWRC_SUCCESS, TWRC_CHECKSTATUS]);
end
else {If it's either not loaded or already enabled}
    {If it is not loaded}
    Result := FALSE or Enabled;

    {Updates property}
    if (Result = TRUE) then fEnabled := TRUE;
end;

{Disables the source}
function TTwainSource.DisableSource(): Boolean;
var
    twUserInterface: TW_USERINTERFACE;
begin
    {Source must be loaded and the value changing}
    if (Loaded) and (Enabled) then
        begin
            {Call method}
            Result := (Owner.TwainProc(AppInfo, @Structure, DG_CONTROL,
                DAT_USERINTERFACE, MSG_DISABLED, @twUserInterface) =
                TWRC_SUCCESS);
            {Call notification event if being used}
            if (Result) and (Assigned(Owner.OnSourceDisable)) then
                Owner.OnSourceDisable(Owner, Index);

        end
    else {If it's either not loaded or already disabled}
        {If it is not loaded}
        Result := TRUE;

        {Updates property}
        if (Result = TRUE) then fEnabled := FALSE;
    end;

    {Loads the source}
    function TTwainSource.LoadSource: Boolean;
    begin
        {Only loads if it is not already loaded}
        if Not Loaded then

```

```

begin
  Result := (Owner.TwainProc(AppInfo, nil, DG_CONTROL, DAT_IDENTITY,
    MSG_OPENDS, @Structure) = TWRC_SUCCESS);
  {Increase the loaded sources count variable}
  if Result then inc(Owner.fSourcesLoaded);
end
else
  {If it was already loaded, returns true}
  Result := TRUE;

  {In case the method was successful, updates property}
  if Result then
    fLoaded := TRUE;

end;

{Unloads the source}
function TTwainSource.UnloadSource: Boolean;
begin
  {Only unloads if it is loaded}
  if Loaded then
    begin
      {If the source was enabled, disable it}
      DisableSource();
      {Call method to load}
      Result := (Owner.TwainProc(AppInfo, nil, DG_CONTROL, DAT_IDENTITY,
        MSG_CLOSED, @Structure) = TWRC_SUCCESS);
      {Decrease the loaded sources count variable}
      if Result then dec(Owner.fSourcesLoaded);
    end
  else
    {If it was already unloaded, returns true}
    Result := TRUE;

    {In case the method was successful, updates property}
    fLoaded := FALSE;

end;

{Object being destroyed}
destructor TTwainSource.Destroy;
begin
  {If loaded, unloads source}
  UnloadSource();
  {Let ancestor class process}
  inherited Destroy;
end;

{Returns a pointer to the application}

```

```

function TTWainSource.GetAppInfo: pTW_IDENTITY;
begin
    Result := Owner.AppInfo;
end;

{Returns a pointer to the source identity}
function TTWainSource.GetStructure: pTW_IDENTITY;
begin
    Result := @Structure;
end;

{Object being created}
constructor TTWainSource.Create(AOwner: TDelphiTwain);
begin
    {Allows ancestor class to process}
    inherited Create(AOwner);

    {Initial values}
    fTransferMode := ttmNative;
    fLoaded := FALSE;
    fShowUI := TRUE;
    fEnabled := FALSE;
    fModal := TRUE;
    {Stores owner}
    fOwner := AOwner;
end;

{Set source transfer mode}
function TTWainSource.ChangeTransferMode(
    NewMode: TTWainTransferMode): TCapabilityRet;
const
    TransferModeToTwain: Array[TTWainTransferMode] of TW_UINT16 =
        (TWSX_FILE, TWSX_NATIVE, TWSX_MEMORY);
var
    Value: TW_UINT16;
begin
    {Set transfer mode method}
    Value := TransferModeToTwain[NewMode];
    Result := SetOneValue(ICAP_XFERMECH, TWTY_UINT16, @Value);
    TransferMode := NewMode;
end;

{Message received in the event loop}
function TTWainSource.ProcessMessage(const Msg: TMsg): Boolean;
var
    twEvent: TW_EVENT;
begin
    {Make twEvent structure}

```

```

twEvent.TWMessage := MSG_NULL;
twEvent.pEvent := TW_MEMREF(@Msg);
{Call Twain procedure to handle message}
Result := (Owner.TwainProc(AppInfo, @Structure, DG_CONTROL, DAT_EVENT,
    MSG_PROCESSEVENT, @twEvent) = TWRC_DSEVENT);

```

```

{If it is a message from the source, process}
if Result then
    case twEvent.TWMessage of
        {No message from the source}
        MSG_NULL: exit;
        {Requested to close the source}
        MSG_CLOSEDREQ:
        begin
            {Call notification event}
            if (Assigned(Owner.OnAcquireCancel)) then
                Owner.OnAcquireCancel(Owner, Index);
            {Disable the source}
            DisableSource();
        end;
        {Ready to transfer the images}
        MSG_XFERREADY:
        {Call method to transfer}
        TransferImages();

        MSG_CLOSEDOK:
        result:=true;

        MSG_DEVICEEVENT:
        result:=true;

```

```

    end {case twEvent.TWMessage}
end;

```

```

{Returns return status information}
function TTWainSource.GetReturnStatus: TW_UINT16;
var
    StatusInfo: TW_STATUS;
begin
    {The source must be loaded in order to get the status}
    if Loaded then
        begin
            {Call method to get the information}
            Owner.TwainProc(AppInfo, @Structure, DG_CONTROL, DAT_STATUS, MSG_GET,
                @StatusInfo);
            Result := StatusInfo.ConditionCode;
        end else Result := 0 {In case it was called while the source was not loaded}
    end;
end;

```

```

{Converts from a result to a TCapabilityRec}
function TTWainSource.ResultToCapabilityRec(
  const Value: TW_UINT16): TCapabilityRet;
begin

  {Test result code to return}
  case Value of
    {Successull, copy handle and return a success value}
    TWRC_SUCCESS: Result := crSuccess;
    {Error, get more on the error, and return result}
    {case} else
      case GetReturnStatus() of
        TWCC_CAPUNSUPPORTED: Result := crUnsupported;
        TWCC_CAPBADOPERATION: Result := crBadOperation;
        TWCC_CAPSEQERROR: Result := crDependencyError;
        TWCC_LOWMEMORY: Result := crLowMemory;
        TWCC_SEQERROR: Result := crInvalidState;
        else Result := crBadOperation;
      end {case GetReturnStatus of}
    end {case};

end;

{Sets a capability}
function TTWainSource.SetCapabilityRec(const Capability,
  ConType: TW_UINT16; Data: HGlobal): TCapabilityRet;
var
  CapabilityInfo: TW_CAPABILITY;
begin
  {Source must be loaded to set}
  if Loaded then
    begin

      {Fill structure}
      CapabilityInfo.Cap := Capability;
      CapabilityInfo.ConType := ConType;
      CapabilityInfo.hContainer := Data;

      {Call method and store return}
      Result := ResultToCapabilityRec(Owner.TwainProc(AppInfo, @Structure,
        DG_CONTROL, DAT_CAPABILITY, MSG_SET, @CapabilityInfo));

    end
    else Result := crInvalidState {In case the source is not loaded}
  end;

{Returns a capability strucutre}

```

```

function TTwainSource.GetCapabilityRec( const Capability: TW_UINT16;
  var Handle: HGLOBAL; Mode: TRetrieveCap;
  var Container: TW_UINT16): TCapabilityRet;
const
  ModeToTwain: Array[TRetrieveCap] of TW_UINT16 = (MSG_GET,
MSG_GETCURRENT,
  MSG_GETDEFAULT, MSG_RESET);
var
  CapabilityInfo: TW_CAPABILITY;
begin
  {Source must be loaded}
  if Loaded then
  begin

    {Fill structure}
    CapabilityInfo.Cap := Capability;
    CapabilityInfo.ConType := TWON_DONTCARE16;
    CapabilityInfo.hContainer := 0;

    {Call method and store return}
    Result := ResultToCapabilityRec(Owner.TwainProc(AppInfo, @Structure,
      DG_CONTROL, DAT_CAPABILITY, ModeToTwain[Mode], @CapabilityInfo));

    if Result = crSuccess then
    begin
      Handle := CapabilityInfo.hContainer;
      Container := CapabilityInfo.ConType;
    end
  end {if not Loaded}
  else Result := crInvalidState {In case the source is not loaded}
end;

{Gets an item and returns it in a string}
procedure TTwainSource.GetItem(var Return: String; ItemType: TW_UINT16;
  Data: Pointer);
begin
  {Test the item type}
  case ItemType of
    TWTY_INT8 :      Return := IntToStr(pTW_INT8(Data)^);
    TWTY_UINT8 :     Return := IntToStr(pTW_UINT8(Data)^);
    TWTY_INT16,
    44 {TWTY_HANDLE} : Return := IntToStr(pTW_INT16(Data)^);
    TWTY_UINT16,
    TWTY_BOOL :      Return := IntToStr(pTW_UINT16(Data)^);
    TWTY_INT32 :     Return := IntToStr(pTW_INT32(Data)^);
    TWTY_UINT32,
    43 {TWTY_MEMREF} : Return := IntToStr(pTW_UINT32(Data)^);
  {Floating integer type}

```

```

TWTY_FIX32:
  with pTW_FIX32(Data)^ do
    //npeter bugfix:
    //it is better to use the actual decimal separator
    //and not a wired in value!
    //If not, you may get error on strtod
    //original: Return := IntToStr(Whole) + ',' + IntToStr(Frac);
    Return := IntToStr(Whole) + decimalseparator + IntToStr(Frac);
  {String types, which are all ended by a null char (#0)}
  TWTY_STR32,
  TWTY_STR64,
  TWTY_STR128,
  TWTY_STR255 :      Return := PChar(Data);

  end {case ItemType}
end;

{Returns an array capability}
function TTWainSource.GetArrayValue(Capability: TW_UINT16;
  var ItemType: TW_UINT16; var List: TGetCapabilityList;
  MemHandle: HGLOBAL): TCapabilityRet;
var
  ArrayV : pTW_ARRAY;
  ItemSize : Integer;
  Data : PChar;
  CurlItem : Integer;
  Value : String;
  Container: TW_UINT16;
begin
  {Call method to get the memory to the return}
  if MemHandle = 0 then
    Result := GetCapabilityRec(Capability, MemHandle, rcGet, Container)
  else
    begin
      Result := crSuccess;
      Container := TWON_ARRAY;
    end;

    if (Result = crSuccess) and (Container <> TWON_ARRAY) then
      begin
        Result := crInvalidContainer;
        GlobalFree(MemHandle);
        Exit;
      end;

      {If result was sucessfull and memory was allocated}
      if (Result = crSuccess) then
        begin

```

```

{Obtain structure pointer}
ArrayV := GlobalLock(MemHandle);

{Fill return properties}
ItemType := ArrayV^.ItemType;

{Prepare to list items}
ItemSize := TWTypeSize(ItemType);
Data := @ArrayV^.ItemList[0];
SetLength(List, ArrayV^.NumItems);

{Copy items}
for CurlItem := 0 TO ArrayV^.NumItems - 1 do
begin
  {Obtain this item}
  GetItem(Value, ItemType, Data);
  List[CurlItem] := Value;
  {Move memory to the next}
  inc(Data, ItemSize);
end;

{Unlock memory and unallocate}
GlobalUnlock(MemHandle);
GlobalFree(MemHandle);
end {if (Result = crSuccess)}
end;

{Returns an enumeration capability}
function TTWainSource.GetEnumerationValue(Capability: TW_UINT16;
  var ItemType: TW_UINT16; var List: TGetCapabilityList;
  var Current, Default: Integer; Mode: TRetrieveCap;
  MemHandle: HGLOBAL): TCapabilityRet;
var
  EnumV : pTW_ENUMERATION;
  ItemSize : Integer;
  Data : PChar;
  CurlItem : Integer;
  Value : String;
  Container: TW_UINT16;
begin
  {Call method to get the memory to the return}
  if MemHandle = 0 then
    Result := GetCapabilityRec(Capability, MemHandle, Mode, Container)
  else
    begin
      Result := crSuccess;
      Container := TWON_ENUMERATION;
    end
  end
end;

```

end;

if (Result = crSuccess) and (Container <> TWON_ENUMERATION) then
begin

 Result := crInvalidContainer;

 GlobalFree(MemHandle);

 Exit;

end;

{If result was sucessfull and memory was allocated}

if (Result = crSuccess) then

begin

 {Obtain structure pointer}

 EnumV := GlobalLock(MemHandle);

 {Fill return properties}

 Current := EnumV^.CurrentIndex;

 Default := EnumV^.DefaultIndex;

 ItemType := EnumV^.ItemType;

 {Prepare to list items}

 ItemSize := TWTypeSize(ItemType);

 Data := @EnumV^.ItemList[0];

 SetLength(List, EnumV^.NumItems);

 {Copy items}

 for Curltem := 0 TO EnumV^.NumItems - 1 do

 begin

 {Obtain this item}

 GetItem(Value, ItemType, Data);

 List[Curltem] := Value;

 {Move memory to the next}

 inc(Data, ItemSize);

 end;

 {Unlock memory and unallocate}

 GlobalUnlock(MemHandle);

 GlobalFree(MemHandle);

end {if (Result = crSuccess)}

end;

{Returns a range capability}

function TTwainSource.GetRangeValue(Capability: TW_UINT16;

 var ItemType: TW_UINT16; var Min, Max, Step, Default,

 Current: String; MemHandle: HGLOBAL): TCapabilityRet;

var

 RangeV : pTW_RANGE;

```

    Container: TW_UINT16;
begin
    {Call method to get the memory to the return}
    if MemHandle = 0 then
        Result := GetCapabilityRec(Capability, MemHandle, rcGet, Container)
    else
        begin
            Result := crSuccess;
            Container := TWON_RANGE;
        end;

    if (Result = crSuccess) and (Container <> TWON_RANGE) then
        begin
            Result := crInvalidContainer;
            GlobalFree(MemHandle);
            Exit;
        end;

    {If result was sucessfull and memory was allocated}
    if (Result = crSuccess) then
        begin
            {Obtain structure pointer}
            RangeV := GlobalLock(MemHandle);
            {Fill return}
            ItemType := RangeV^.ItemType;
            GetItem(Min, ItemType, @RangeV^.MinValue);
            GetItem(Max, ItemType, @RangeV^.MaxValue);
            GetItem(Step, ItemType, @RangeV^.StepSize);
            GetItem(Default, ItemType, @RangeV^.DefaultValue);
            GetItem(Current, ItemType, @RangeV^.CurrentValue);

            {Unlock memory and unallocate}
            GlobalUnlock(MemHandle);
            GlobalFree(MemHandle);
        end {if (Result = crSuccess)}
    end;

    {Returns an one value capability}
    function TTwainSource.GetOneValue(Capability: TW_UINT16;
    var ItemType: TW_UINT16; var Value: String;
    Mode: TRetrieveCap; MemHandle: HGLOBAL): TCapabilityRet;
var
    OneV    : pTW_ONEVALUE;
    Container: TW_UINT16;
begin
    {Call method to get the memory to the return}
    if MemHandle = 0 then
        Result := GetCapabilityRec(Capability, MemHandle, Mode, Container)

```

```

else
begin
    Result := crSuccess;
    Container := TWON_ONEVALUE;
end;

if (Result = crSuccess) and (Container <> TWON_ONEVALUE) then
begin
    Result := crInvalidContainer;
    GlobalFree(MemHandle);
    Exit;
end;

{If result was sucessfull and memory was allocated}
if (Result = crSuccess) then
begin
    {Obtain structure pointer}
    OneV := GlobalLock(MemHandle);
    {Fill return}
    ItemType := OneV^.ItemType;
    GetItem(Value, OneV^.ItemType, @OneV^.Item);

    {Unlock memory and unallocate}
    GlobalUnlock(MemHandle);
    GlobalFree(MemHandle);
end {if (Result = crSuccess)}
end;

{Sets an one value capability}
function TTwainSource.SetOneValue(Capability: TW_UINT16;
    ItemType: TW_UINT16; Value: Pointer): TCapabilityRet;
var
    Data: HGLOBAL;
    OneV: pTW_ONEVALUE;
    ItemSize, ItemSize2: Integer;
begin
    {Allocate enough memory for the TW_ONEVALUE and obtain pointer}
    ItemSize := TWTypeSize(ItemType);
    //npeter: TW_ONEVALUE minimal size !!!
    //I think to meet the specifications the
    //Item's size must be at least sizeof(TW_UINT32)!
    //when I did it, some mistic errors on some drivers went gone
    if ItemSize < TWTypeSize(TWTY_UINT32) then ItemSize2 := TWTypeSize(TWTY_UINT32)
else ItemSize2 := ItemSize;
    Data := GlobalAlloc(GHND, sizeof(OneV^.ItemType) + ItemSize2);
    OneV := GlobalLock(Data);

    {Fill value}

```

```

OneV^.ItemType := ItemType;
CopyMemory(@OneV^.Item, Value, ItemSize);
GlobalUnlock(Data);

```

```

{Call method to set}
Result := SetCapabilityRec(Capability, TWON_ONEVALUE, Data);

```

```

{Unload memory}
GlobalFree(Data);
end;

```

```

{Sets a range capability}
function TTwainSource.SetRangeValue(Capability: TW_UINT16;
  ItemType: TW_UINT16; Min, Max, Step, Current: TW_UINT32): TCapabilityRet;
var
  Data: HGLOBAL;
  RangeV: pTW_RANGE;
begin
  {Allocate enough memory for the TW_RANGE and obtain pointer}
  Data := GlobalAlloc(GHND, sizeof(TW_RANGE));
  RangeV := GlobalLock(Data);

```

```

  {Fill value}
  RangeV^.ItemType := ItemType;
  RangeV^.MinValue := Min;
  RangeV^.MaxValue := Max;
  RangeV^.StepSize := Step;
  RangeV^.CurrentValue := Current;
  GlobalUnlock(Data);

```

```

  {Call method to set}
  Result := SetCapabilityRec(Capability, TWON_RANGE, Data);

```

```

  {Unload memory}
  GlobalFree(Data);
end;

```

```

{Sets an array capability}
function TTwainSource.SetArrayValue(Capability: TW_UINT16;
  ItemType: TW_UINT16; List: TSetCapabilityList): TCapabilityRet;
var
  Data: HGLOBAL;
  EnumV: pTW_ENUMERATION;
  i, ItemSize: Integer;
  DataPt: PChar;
begin
  {Allocate enough memory for the TW_ARRAY and obtain pointer}
  ItemSize := TWTypeSize(ItemType);

```

```
Data := GlobalAlloc(GHND, sizeof(TW_ARRAY) + ItemSize * Length(List));
EnumV := GlobalLock(Data);
```

```
{Fill values}
EnumV^.ItemType := ItemType;
EnumV^.NumItems := Length(List);
```

```
{Copy item values}
DataPt := @EnumV^.ItemList[0];
for i := Low(List) TO High(List) do
begin
  {Copy item}
  CopyMemory(DataPt, List[i], ItemSize);
  {Move to next item}
  inc(DataPt, ItemSize);
end;
GlobalUnlock(Data);
```

```
{Call method to set}
Result := SetCapabilityRec(Capability, TWON_ARRAY, Data);
```

```
{Unload memory}
GlobalFree(Data);
end;
```

```
{Sets an enumeration capability}
function TTwainSource.SetEnumerationValue(Capability: TW_UINT16;
  ItemType: TW_UINT16; CurrentIndex: TW_UINT32;
  List: TSetCapabilityList): TCapabilityRet;
var
  Data: HGLOBAL;
  EnumV: pTW_ENUMERATION;
  i, ItemSize: Integer;
  DataPt: PChar;
begin
  {Allocate enough memory for the TW_ENUMERATION and obtain pointer}
  ItemSize := TWTypeSize(ItemType);
  Data := GlobalAlloc(GHND, sizeof(TW_ENUMERATION) + ItemSize * Length(List));
  EnumV := GlobalLock(Data);
```

```
{Fill values}
EnumV^.ItemType := ItemType;
EnumV^.NumItems := Length(List);
EnumV^.CurrentIndex := CurrentIndex;
```

```
{Copy item values}
DataPt := @EnumV^.ItemList[0];
for i := Low(List) TO High(List) do
```

```

begin
  {Copy item}
  CopyMemory(DataPt, List[i], ItemSize);
  {Move to next item}
  inc(DataPt, ItemSize);
end;
GlobalUnlock(Data);

{Call method to set}
Result := SetCapabilityRec(Capability, TWON_ENUMERATION, Data);

{Unload memory}
GlobalFree(Data);
end;

{Transfer image memory}
function TTwainSource.TransferImageMemory(var ImageHandle: HBitmap;
  PixelType: TW_INT16): TW_UINT16;
var
  {Memory buffer information from the source}
  Setup : TW_SETUPMEMXFER;
  {Memory information from the image}
  Xfer : TW_IMAGEMEMXFER;
  {Image processing variables}
  ImageInfo : Windows.TBitmap;
  Ptr : pChar;
  LineLength,
  CurLine: Cardinal;
  LinePtr,
  AllocPtr : pointer;
  DataSize,
  Readed,
  Index : Cardinal;
  ItemPtr : pRGBTriple;
  Temp : Byte;
begin
  {Obtain information on the transference buffers}
  Result := Owner.TwainProc(AppInfo, @Structure, DG_CONTROL,
DAT_SETUPMEMXFER,
  MSG_GET, @Setup);

  {Get information on the bitmap}
  GetObject(ImageHandle, sizeof(Windows.TBitmap), @ImageInfo);
  LineLength := (((ImageInfo.bmWidth * ImageInfo.bmBitsPixel + 31) div 32) * 4);
  {Get pointer for the last line}
  CurLine := ImageInfo.bmHeight - 1;
  Cardinal(LinePtr) := Cardinal(ImageInfo.bmBits) + LineLength * CurLine;

```

```

Ptr := LinePtr;
DataSize := 0;

{Prepare buffer record to transfer}
Fillchar(Xfer, SizeOf(TW_IMAGEMEMXFER), $FF);
Xfer.Memory.Flags := TWMF_APPOWNS or TWMF_POINTER;
Xfer.Memory.Length := Setup.Preferred;
GetMem(AllocPtr, Setup.Preferred);
Xfer.Memory.TheMem := AllocPtr;

{Transfer data until done or cancelled}
if Result = TWRC_SUCCESS then
  repeat
    {Retrieve another piece of memory to the pointer}
    Xfer.BytesWritten := 0;
    Result := Owner.TwainProc(AppInfo, @Structure, DG_IMAGE,
      DAT_IMAGEMEMXFER, MSG_GET, @Xfer);
    {Test the result}
    {Piece successfully transfer, move to next}
    if (Result = TWRC_SUCCESS) or (Result = TWRC_XFERDONE) then
      begin
        {While we have data}
        while Xfer.BytesWritten > 0 do
          begin
            {In case the total bytes received now have more than we}
            {need to complete the line}
            if Xfer.BytesWritten + DataSize > LineLength then
              begin
                Readed := LineLength - DataSize;
                CopyMemory(Ptr, Xfer.Memory.TheMem, LineLength - DataSize);
              end
            else
              {Otherwise, continue completing the line}
              begin
                Readed := Xfer.BytesWritten;
                CopyMemory(Ptr, Xfer.Memory.TheMem, Readed);
              end;
            {Adjust}
            inc(DataSize, Readed); inc(Ptr, Readed);
            dec(Xfer.BytesWritten, Readed);
            Cardinal(Xfer.Memory.TheMem) :=
              Cardinal(Xfer.Memory.TheMem) + Readed;

            {Reached end of line}
            if DataSize >= LineLength then
              begin
                {Fix RGB to BGR}

```

```

if PixelType = TWPT_RGB then
begin
  ItemPtr := LinePtr;
  FOR Index := 1 TO ImageInfo.bmWidth DO
  begin
    Temp := ItemPtr^.rgbtRed;
    ItemPtr^.rgbtRed := ItemPtr^.rgbtBlue;
    ItemPtr^.rgbtBlue := Temp;
    inc(ItemPtr);
  end {FOR Index};
end {if PixelType = TWPT_RGB};

{Adjust pointers}
Cardinal(LinePtr) := Cardinal(LinePtr) - LineLength;
Ptr := LinePtr; dec(CurLine); DataSize := 0;

{Call event}
if Assigned(Owner.OnAcquireProgress) then
  Owner.OnAcquireProgress(Self, Self.Index, ImageHandle,
    Cardinal(ImageInfo.bmHeight) - CurLine - 1,
    ImageInfo.bmHeight - 1);

end {if DataSize >= LineLength}

end {while Xfer.BytesWritten > 0};

{Set again pointer to write to}
Xfer.Memory.TheMem := AllocPtr;
end {TWRC_SUCCESS};

until Result <> TWRC_SUCCESS;

{Free allocated memory}
FreeMem(AllocPtr, Setup.Preferred);

{Some error occurred, free memory and returns}
if Result <> TWRC_XFERDONE then
  DeleteObject(ImageHandle);
end;

{Prepare image memory transference}
function TTwainSource.PrepareMemXfer(var BitmapHandle: HBitmap;
  var PixelType: TW_INT16): TW_UINT16;
const
  PixelColor: Array[TTwainPixelFlavor] of Array[0..1] of Byte =
    ((0, $FF), ($FF, 00), (0, $FF));
var

```

```

Handle: HGlobal;
Info: TW_IMAGEINFO;
Setup: TW_SETUPMEMXFER;
structsize, index, Size, Blocks: Integer;
XRes, YRes: Extended;
Pal : TW_PALETTE8;
vUnit : TTWainUnit;
vUnits: TTWainUnitSet;
Dib : pBitmapInfo;
PixelFlavor: TTWainPixelFlavor;
PixelFlavors: TTWainPixelFlavorSet;
DC: HDC;
Data : Pointer;
begin
  {First of all, get information on the image being acquired}
  Result := Owner.TwainProc(AppInfo, @Structure, DG_IMAGE, DAT_IMAGEINFO,
    MSG_GET, @Info);
  if Result <> TWRC_SUCCESS then exit;

  {Calculate image size}
  with Info do
    size := (((ImageWidth * BitsPerPixel + 31) div 32)*4) * info.ImageLength;

  {Obtain image buffer transference sizes}
  Owner.TwainProc(AppInfo, @Structure, DG_CONTROL, DAT_SETUPMEMXFER,
    MSG_GET, @Setup);
  blocks := (size div Integer(setup.Preferred));
  size := (blocks + 1) * Integer(setup.Preferred);

  {Prepare new bitmap}
  structsize := size + sizeof(BITMAPINFOHEADER) + 256 * sizeof(RGBQUAD);

  Handle := GlobalAlloc(GHND, StructSize);
  Dib := GlobalLock(Handle);
  Fillchar(Dib^, structsize, #0);
  {Fill image information}
  Dib^.bmiHeader.biSize := sizeof(BITMAPINFOHEADER);
  Dib^.bmiHeader.biWidth := info.ImageWidth;
  Dib^.bmiHeader.biHeight := info.ImageLength;
  {Only 1 plane supported}
  Dib^.bmiHeader.biPlanes := 1;
  Dib^.bmiHeader.biBitCount := info.BitsPerPixel;
  {No compression}
  Dib^.bmiHeader.biCompression := BI_RGB;
  Dib^.bmiHeader.biSizeImage := Size;

  {Adjust units}
  XRes := Fix32ToFloat(Info.XResolution);

```

```

YRes := Fix32ToFloat(Info.YResolution);
GetICapUnits(vUnit, vUnits);
case vUnit of
  tuInches: begin
    Dib^.bmiHeader.biXPelsPerMeter := Trunc((XRes*2.54)*100);
    Dib^.bmiHeader.biYPelsPerMeter := Trunc((YRes*2.54)*100);
  end;
  tuCentimeters: begin
    Dib^.bmiHeader.biXPelsPerMeter := Trunc(XRes*100);
    Dib^.bmiHeader.biYPelsPerMeter := Trunc(YRes*100);
  end
else begin
  Dib^.bmiHeader.biXPelsPerMeter := 0;
  Dib^.bmiHeader.biYPelsPerMeter := 0;
end
end {case vUnits of};

{Now it should setup the palette to be used by the image}
{by either building a defined palette or retrieving the}
{image's one}
case (Info.PixelType) of
  TWPT_BW:
    begin
      {Only two colors are used}
      Dib^.bmiHeader.biClrUsed := 2;
      Dib^.bmiHeader.biClrImportant := 0;
      {Try obtaining the pixel flavor}
      if GetIPixelFlavor(PixelFlavor, PixelFlavors) <> crSuccess then
        PixelFlavor := tpfChocolate;
      {Set palette colors}
      for Index := 0 to 1 do
        begin
          Dib^.bmiColors[Index].rgbRed := PixelColor[PixelFlavor][Index];
          Dib^.bmiColors[Index].rgbGreen := PixelColor[PixelFlavor][Index];
          Dib^.bmiColors[Index].rgbBlue := PixelColor[PixelFlavor][Index];
          Dib^.bmiColors[Index].rgbReserved := 0;
        end;
      end;

    end;
  TWPT_GRAY:
    begin
      {Creates a 256 shades of gray palette}
      Dib^.bmiHeader.biClrUsed := 256;
      for index := 0 to 255 do
        begin
          Dib^.bmiColors[index].rgbRed := index;
          Dib^.bmiColors[index].rgbGreen := index;
          Dib^.bmiColors[index].rgbBlue := index;
        end;
      end;
    end;
end;

```

```

    Dib^.bmiColors[index].rgbReserved := 0;
  end {for i}
end;
TWPT_RGB: Dib^.bmiHeader.biClrUsed := 0;
else
begin
  {Try obtaining the palette}
  if Owner.TwainProc(AppInfo, @Structure, DG_CONTROL, DAT_PALETTE8,
    MSG_GET, @Pal) <> TWRC_SUCCESS then
  begin
    {If the source did not provide a palette, uses shades of gray here}
    Dib^.bmiHeader.biClrUsed := 256;
    for index := 0 to 255 do
    begin
      Dib^.bmiColors[index].rgbRed := index;
      Dib^.bmiColors[index].rgbGreen := index;
      Dib^.bmiColors[index].rgbBlue := index;
      Dib^.bmiColors[index].rgbReserved := 0;
    end {for i}
  end
  else
  begin
    {Uses source palette here}
    Dib^.bmiHeader.biClrUsed := Pal.NumColors;
    for Index := 0 TO Pal.NumColors - 1 do
    begin
      Dib^.bmiColors[index].rgbRed := pal.Colors[index].Channel1;
      Dib^.bmiColors[index].rgbGreen := pal.Colors[index].Channel2;
      Dib^.bmiColors[index].rgbBlue := pal.Colors[index].Channel3;
      Dib^.bmiColors[index].rgbReserved := 0;
    end {for Index}
  end {if Owner.TwainProc(AppInfo...)}

  end {case else};
end {case Info.PixelType};

{Creates the bitmap}
DC := GetDC(Owner.VirtualWindow);
Cardinal(Data) := Cardinal(Dib) + Dib^.bmiHeader.biSize +
  (Dib^.bmiHeader.biClrUsed * sizeof(RGBQUAD));
BitmapHandle := CreateDIBSection(DC, Dib^, DIB_RGB_COLORS, Data, 0, 0);
ReleaseDC(Owner.VirtualWindow, DC);
PixelType := Info.PixelType;

{Unlock and free data}
GlobalUnlock(Handle);
GlobalFree(Handle);
end;
```

```

{Method to transfer the images}
procedure TTWainSource.TransferImages();
var
  {To test if the image transfer is done}
  Cancel, Done : Boolean;
  {Return code from Twain method}
  rc : TW_UINT16;
  {Handle to the native Device independent Image (DIB)}
  hNative: TW_UINT32;
  {Pending transfers structure}
  PendingXfers: TW_PENDINGXFERS;
  {File transfer info}
  Info: TW_SETUPFILEXFER;
  {Image handle and pointer}
  ImageHandle: HBitmap;
  PixelType : TW_INT16;
begin
  {Set the transfer mode}
  //npeter:
  //on a HP driver I got error events
  //when it was set above state 5;
  //commented out
  // ChangeTransferMode(TransferMode);

  Cancel := FALSE; {Testing if it was cancelled}
  Done := FALSE; {Initialize done variable}

  {Obtain all the images from the source}
  repeat
    {Transfer depending on the transfer mode}
    case TransferMode of
      {Native transfer, the source creates the image thru a device}
      {dependent image}
      ttmNative:
      begin
        {Call method to obtain the image}
        hNative := 0;
        rc := Owner.TwainProc(AppInfo, @Structure, DG_IMAGE,
          DAT_IMAGENATIVEXFER, MSG_GET, @hNative);
      end {case ttmNative};
      {File transferring, the source should create a file with}
      {the acquired image}
      ttmFile:
      begin
        {Event to allow user to set the file transfer information}
        if Assigned(Owner.OnSourceSetupFileXfer) then
          Owner.OnSourceSetupFileXfer(Owner, Index);
      end;
    end;
  until Done;
end;

```

```

Owner.TwainProc(AppInfo, @Structure, DG_CONTROL, DAT_SETUPFILEXFER,
MSG_GET, @Info);
{Call method to make source acquire and create file}
rc := Owner.TwainProc(AppInfo, @Structure, DG_IMAGE,
DAT_IMAGEFILEXFER, MSG_GET, nil);
end {case ttmFile};
{Memory buffer transfers}
ttmMemory:
begin
{Prepare for memory transference}
rc := PrepareMemXfer(ImageHandle, PixelType);
{If the image was sucessfully prepared to be transfered, it's}
{now time to transfer it}
if rc = TWRC_SUCCESS then rc := TransferImageMemory(ImageHandle,
PixelType);
end
{Unknown transfer mode ?}
else Rc := 0;
end;

{Twain call to transfer image return}
case rc of
{Transfer sucessfully done}
TWRC_XFERDONE:
case TransferMode of
{Native transfer sucessfull}
ttmNative: ReadNative(hNative, Cancel);
{File transfer sucessfull}
ttmFile: ReadFile(Info.FileName, Info.Format, Cancel);
{Memory transfer sucessfull}
ttmMemory: ReadMemory(ImageHandle, Cancel);
end {case TransferMode, TWRC_XFERDONE};
{User cancelled the transfers}
TWRC_CANCEL:
begin
{Acknowledge end of transfer}
Done := TRUE;
{Call event, if avaiable}
if Assigned(Owner.OnAcquireCancel) then
Owner.OnAcquireCancel(Owner, Index)
end
else {Unknown return or error}
if Assigned(Owner.OnAcquireError) then
Owner.OnAcquireError(Owner, Index, Rc, GetReturnStatus())
end;

{Check if there are pending transfers}
if not Done then

```

```

    Done := (Owner.TwainProc(AppInfo, @Structure, DG_CONTROL,
        DAT_PENDINGXFERS, MSG_ENDXFER, @PendingXfers) <> TWRC_SUCCESS)
or
    (PendingXfers.Count = 0);

    {If user has cancelled}
    if not Done and Cancel then
        Done := (Owner.TwainProc(AppInfo, @Structure, DG_CONTROL,
            DAT_PENDINGXFERS, MSG_RESET, @PendingXfers) = TWRC_SUCCESS);

until Done;

{Disable source}
Enabled := False;
end;

{Returns the number of colors in the DIB}
function DibNumColors (pv: Pointer): Word;
var
    Bits: Integer;
    lpbi: PBITMAPINFOHEADER absolute pv;
    lpbc: PBITMAPCOREHEADER absolute pv;
begin
    //With the BITMAPINFO format headers, the size of the palette
    //is in biClrUsed, whereas in the BITMAPCORE - style headers, it
    //is dependent on the bits per pixel ( = 2 raised to the power of
    //bits/pixel).
    if (lpbi^.biSize <> sizeof(BITMAPCOREHEADER)) then
        begin
            if (lpbi^.biClrUsed <> 0) then
                begin
                    result := lpbi^.biClrUsed;
                    exit;
                end;
            Bits := lpbi^.biBitCount;
        end
    else
        Bits := lpbc^.bcBitCount;

    {Test bits to return}
    case (Bits) of
        1: Result := 2;
        4: Result := 16;
        8: Result := 256;
        else Result := 0;
    end {case};

end;

```

```

{Converts from TWain TW_UINT16 to TTWainFormat}
function TwainToTTWainFormat(Value: TW_UINT16): TTWainFormat;
begin
  Case Value of
    TWFF_TIFF      : Result := tfTIFF;
    TWFF_PICT      : Result := tfPict;
    TWFF_BMP       : Result := tfBMP;
    TWFF_XBM       : Result := tfXBM;
    TWFF_JFIF      : Result := tfJPEG;
    TWFF_FPX       : Result := tfFPX;
    TWFF_TIFFMULTI : Result := tfTIFFMulti;
    TWFF_PNG       : Result := tfPNG;
    TWFF_SPIFF     : Result := tfSPIFF;
    TWFF_EXIF      : Result := tfEXIF;
    else           : Result := tfUnknown;
  end {case Value of}
end;

{Reads the file image}
procedure TTWainSource.ReadFile(Name: TW_STR255; Format: TW_UINT16;
  var Cancel: Boolean);
begin
  {Call event, if set}
  if Assigned(Owner.OnSourceFileTransfer) then
    Owner.OnSourceFileTransfer(Self, Index, Name, TwainToTTWainFormat(Format),
      Cancel);
end;

{Call event for memory image}
procedure TTWainSource.ReadMemory(Image: HBitmap; var Cancel: Boolean);
{$IFDEF DONTUSEVCL} var BitmapObj: TBitmap;{$ENDIF}
begin
  if Assigned(Owner.OnTwainAcquire) then
    {$IFDEF DONTUSEVCL}
    Owner.OnTwainAcquire(Owner, Index, Image, Cancel); {$ELSE}
    begin
      BitmapObj := TBitmap.Create;
      BitmapObj.Handle := Image;
      Owner.OnTwainAcquire(Owner, Index, BitmapObj, Cancel);
      BitmapObj.Free;
    end; {$ENDIF}
  end;
end;

{Reads a native image}
procedure TTWainSource.ReadNative(Handle: TW_UINT32; var Cancel: Boolean);
var

```

```

DibInfo: ^TBITMAPINFO;
ColorTableSize: Integer;
lpBits: PChar;
DC: HDC;
BitmapHandle: HBitmap;
{$IFDEF DONTUSEVCL}BitmapObj: TBitmap;{$ENDIF}
begin

  {Get image information pointer and size}
  DibInfo := GlobalLock(Handle);
  ColorTableSize := (DibNumColors(DibInfo) * SizeOf(RGBQUAD));

  {Get data memory position}
  lpBits := PChar(DibInfo);
  Inc(lpBits, DibInfo.bmiHeader.biSize);
  Inc(lpBits, ColorTableSize);

  {Creates the bitmap}
  DC := GetDC(Owner.VirtualWindow);
  BitmapHandle := CreateDIBitmap(DC, DibInfo.bmiHeader, CBM_INIT,
    lpBits, DibInfo^, DIB_RGB_COLORS);
  ReleaseDC(Owner.VirtualWindow, DC);

  if Assigned(Owner.OnTwainAcquire) then
  {$IFDEF DONTUSEVCL}
  Owner.OnTwainAcquire(Owner, Index, BitmapHandle, Cancel); {$ELSE}
  begin
    BitmapObj := TBitmap.Create;
    BitmapObj.Handle := BitmapHandle;
    Owner.OnTwainAcquire(Owner, Index, BitmapObj, Cancel);
    BitmapObj.Free;
  end; {$ENDIF}

  {Free bitmap}
  GlobalUnlock(Handle);
  GlobalFree(Handle);
end;

{Setup file transfer}
function TTwainSource.SetupFileTransfer(Filename: String;
  Format: TTwainFormat): Boolean;
const
  FormatToTwain: Array[TTwainFormat] of TW_UINT16 = (TWFF_TIFF,
    TWFF_PICT, TWFF_BMP, TWFF_XBM, TWFF_JFIF, TWFF_FPX, TWFF_TIFFMULTI,
    TWFF_PNG, TWFF_SPIFF, TWFF_EXIF, 0);
var
  FileTransferInfo: TW_SETUPFILEXFER;
begin

```

```

{Source must be loaded to set things}
if (Loaded) then
begin
  {Prepare structure}
  FileTransferInfo.FileName := StrToStr255(FileName);
  FileTransferInfo.Format := FormatToTwain[Format];

  {Call method}
  Result := (Owner.TwainProc(AppInfo, @Structure, DG_CONTROL,
    DAT_SETUPFILEXFER, MSG_SET, @FileTransferInfo) = TWRC_SUCCESS);
end
else Result := FALSE; {Could not set file transfer with source unloaded}
end;

{Set the number of images that the application wants to receive}
function TTwainSource.SetCapXferCount(Value: SmallInt): TCapabilityRet;
begin
  {Call method to set the value}
  Result := SetOneValue(CAP_XFERCOUNT, TWTY_UINT16, @Value);
end;

{Returns the number of images that the source will return}
function TTwainSource.GetCapXferCount(var Return: SmallInt;
  Mode: TRetrieveCap): TCapabilityRet;
var
  {Will hold the capability information}
  ItemType: TW_UINT16;
  Value : String;
begin
  {Call method to return information}
  Result := GetOneValue(CAP_XFERCOUNT, ItemType, Value, Mode);
  {Item type must be of TW_UINT16}
  if (Result = crSuccess) and (ItemType <> TWTY_INT16) then
    Result := crUnsupported;
  {If everything gone ok, fill result}
  if Result = crSuccess then Return := StrToIntDef(Value, -1);
end;

{Set the unit measure}
function TTwainSource.SetICapUnits(Value: TTwainUnit): TCapabilityRet;
//npeter
//the TTwainUnit is byte!!!
//so we have to convert it to TW_UINT16
//before this fix I was not able to set this capability
//on a HP driver
const Transfer: Array[TTwainUnit] of TW_UINT16 =
  (TWUN_INCHES, TWUN_CENTIMETERS, TWUN_PICAS, TWUN_POINTS,
  TWUN_TWIPS, TWUN_PIXELS, TWUN_INCHES);

```

```

var
  iValue: TW_UINT16;
begin
  iValue:=Transfer[Value];
  Result := SetOneValue(ICAP_UNITS, TWTY_UINT16, @iValue);
end;

{Convert from Twain to TTWainPixelFormat}
function TwainToTTWainPixelFormat(Value: TW_UINT16): TTWainPixelFormat;
begin
  {Test the value to make the conversion}
  case Value of
    TWPF_CHOCOLATE: Result := tpfChocolate;
    TWPF_VANILLA : Result := tpfVanilla;
  else Result := tpfUnknown;
  end {case Value}
end;

{Convert from Twain to TTWainUnit}
function TwainToTTWainUnit(Value: TW_UINT16): TTWainUnit;
begin
  {Test the value to make the conversion}
  case Value of
    TWUN_INCHES : Result := tuInches;
    TWUN_CENTIMETERS: Result := tuCentimeters;
    TWUN_PICAS : Result := tuPicas;
    TWUN_POINTS : Result := tuPoints;
    TWUN_TWIPS : Result := tuTwips;
    TWUN_PIXELS : Result := tuPixels;
  else Result := tuUnknown;
  end {case Value}
end;

{Retrieve the unit measure for all quantities}
function TTWainSource.GetICapUnits(var Return: TTWainUnit;
  var Supported: TTWainUnitSet; Mode: TRetrieveCap): TCapabilityRet;
var
  ItemType: TW_UINT16;
  List : TGetCapabilityList;
  Current, i,
  Default : Integer;
begin
  {Call method to get result}
  Result := GetEnumerationValue(ICAP_UNITS, ItemType, List, Current, Default,
    Mode);
  if ItemType <> TWTY_UINT16 then Result := crUnsupported;

  {If it was sucessfull, return values}

```

```

if Result = crSuccess then
begin
  {Make list}
  for i := Low(List) to High(List) do
    Include(Supported, TwainToTTwainUnit(StrToIntDef(List[i], -1)));
  {Return values depending on the mode}
  if Mode = rcGetDefault then
    Return := TwainToTTwainUnit(StrToIntDef(List[Default], -1))
  else
    Return := TwainToTTwainUnit(StrToIntDef(List[Current], -1));
  end {if Result = crSuccess}

end;

{Retrieve the pixel flavor values}
function TTwainSource.GetIPixelFlavor(var Return: TTwainPixelFlavor;
  var Supported: TTwainPixelFlavorSet; Mode: TRetrieveCap): TCapabilityRet;
var
  ItemType: TW_UINT16;
  List : TGetCapabilityList;
  Current, i,
  Default : Integer;
begin
  {Call method to get result}
  Result := GetEnumerationValue(ICAP_PIXELFLAVOR, ItemType, List, Current,
    Default, Mode);
  if ItemType <> TWTY_UINT16 then Result := crUnsupported;

  {If it was sucessfull, return values}
  if Result = crSuccess then
  begin
    {Make list}
    for i := Low(List) to High(List) do
      Include(Supported, TwainToTTwainPixelFlavor(StrToIntDef(List[i], -1)));
    {Return values depending on the mode}
    if Mode = rcGetDefault then
      Return := TwainToTTwainPixelFlavor(StrToIntDef(List[Default], -1))
    else
      Return := TwainToTTwainPixelFlavor(StrToIntDef(List[Current], -1));
    end {if Result = crSuccess}
  end;

function TTwainSource.SetIPixelFlavor(Value: TTwainPixelFlavor): TCapabilityRet;
//npeter
//the TTwainPixelFlavor is byte!!!
//so we have to convert it to TW_UINT16
//before this fix I was not able to set this capability
//on a HP driver

```

```

const Transfer: array [TTwainPixelFlavor] of TW_UINT16 =
(TWPF_CHOCOLATE,TWPF_VANILLA,TWPF_CHOCOLATE);
var iValue: TW_UINT16;
begin
  iValue:=Transfer[value];
  Result := SetOneValue(ICAP_PIXELFLAVOR, TWTY_UINT16, @iValue);
end;

{Convert from Twain to TTwainPixelFormat}
function TwainToTTwainPixelFormat(Value: TW_UINT16): TTwainPixelFormat;
begin
  {Test the value to make the conversion}
  case Value of
    TWPT_BW      : Result := tbdBw;
    TWPT_GRAY    : Result := tbdGray;
    TWPT_RGB     : Result := tbdRgb;
    TWPT_PALETTE : Result := tbdPalette;
    TWPT_CMY     : Result := tbdCmy;
    TWPT_CMYK    : Result := tbdCmyk;
    TWPT_YUV     : Result := tbdYuv;
    TWPT_YUVK    : Result := tbdYuvk;
    TWPT_CIEXYZ  : Result := tbdCieXYZ;
  else Result := tbdUnknown;
  end {case Value}
end;

{Returns pixel type values}
function TTwainSource.GetPixelFormat(var Return: TTwainPixelFormat;
  var Supported: TTwainPixelFormatSet; Mode: TRetrieveCap): TCapabilityRet;
var
  ItemType: TW_UINT16;
  List : TGetCapabilityList;
  Current, i,
  Default : Integer;
begin
  {Call method to get result}
  Result := GetEnumerationValue(ICAP_PIXELTYPE, ItemType, List, Current,
    Default, Mode);
  if ItemType <> TWTY_UINT16 then Result := crUnsupported;

  {If it was sucessfull, return values}
  if Result = crSuccess then
  begin
    {Make list}
    for i := Low(List) to High(List) do
      Include(Supported, TwainToTTwainPixelFormat(StrToIntDef(List[i], -1)));
    {Return values depending on the mode}
    if Mode = rcGetDefault then

```

```

    Return := TwainToTTwainPixelFormat(StrToIntDef(List[Default], -1))
  else
    Return := TwainToTTwainPixelFormat(StrToIntDef(List[Current], -1));
  end {if Result = crSuccess}
end;

{Set the pixel type value}
function TTWainSource.SetPixelFormat(Value: TTWainPixelFormat): TCapabilityRet;
//npeter
//the TTWainPixelFormat is byte!!!
//so we have to convert it to TW_UINT16
//before this fix occasionally I was not able to set this capability
//on a HP driver
var ivalue: smallint;
begin
  ivalue:=ord(Value);
  Result := SetOneValue(ICAP_PIXELTYPE, TWTY_UINT16, @ivalue);
end;

{Returns bitdepth values}
function TTWainSource.GetBitDepth(var Return: Word;
  var Supported: TTWainBitDepth; Mode: TRetrieveCap): TCapabilityRet;
var
  ItemType: TW_UINT16;
  List : TGetCapabilityList;
  Current, i,
  Default : Integer;
begin
  {Call GetOneValue to obtain this property}
  Result := GetEnumerationValue(ICAP_BITDEPTH, ItemType, List, Current,
    Default, Mode);
  if ItemType <> TWTY_UINT16 then Result := crUnsupported;

  {In case everything went ok, fill parameters}
  if Result = crSuccess then
    begin
      {Build bit depth list}
      SetLength(Supported, Length(List));
      FOR i := LOW(List) TO HIGH(List) DO
        Supported[i] := StrToIntDef(List[i], -1);
      {Return values depending on the mode}
      if Mode = rcGetDefault then Return := StrToIntDef(List[Default], -1)
      else Return := StrToIntDef(List[Current], -1);
      end {if Result = crSuccess}
    end;

  {Set current bitdepth value}
  function TTWainSource.SetBitDepth(Value: Word): TCapabilityRet;

```

```

begin
  Result := SetOneValue(ICAP_BITDEPTH, TWTY_UINT16, @Value);
end;

{Returns physical width}
function TTWainSource.GetPhysicalWidth(var Return: Extended;
  Mode: TRetrieveCap): TCapabilityRet;
var
  Handle: HGlobal;
  OneV : pTW_ONEVALUE;
  Container: TW_UINT16;
begin
  {Obtain handle to data from this capability}
  Result := GetCapabilityRec(ICAP_PHYSICALWIDTH, Handle, Mode, Container);
  if Result = crSuccess then
    begin
      {Obtain data}
      OneV := GlobalLock(Handle);
      if OneV^.ItemType <> TWTY_FIX32 then Result := crUnsupported
      else Return := Fix32ToFloat(pTW_FIX32(@OneV^.Item)^);
      {Free data}
      GlobalUnlock(Handle);
      GlobalFree(Handle);
    end;
  end;
end;

{Returns physical height}
function TTWainSource.GetPhysicalHeight(var Return: Extended;
  Mode: TRetrieveCap): TCapabilityRet;
var
  Handle: HGlobal;
  OneV : pTW_ONEVALUE;
  Container: TW_UINT16;
begin
  {Obtain handle to data from this capability}
  Result := GetCapabilityRec(ICAP_PHYSICALHEIGHT, Handle, Mode, Container);
  if Result = crSuccess then
    begin
      {Obtain data}
      OneV := GlobalLock(Handle);
      if OneV^.ItemType <> TWTY_FIX32 then Result := crUnsupported
      else Return := Fix32ToFloat(pTW_FIX32(@OneV^.Item)^);
      {Free data}
      GlobalUnlock(Handle);
      GlobalFree(Handle);
    end;
  end;
end;

```

```

{Returns a resolution}
function TTWainSource.GetResolution(Capability: TW_UINT16; var Return: Extended;
  var Values: TTWainResolution; Mode: TRetrieveCap): TCapabilityRet;
var
  Handle: HGlobal;
  EnumV: pTW_ENUMERATION;
  Container: TW_UINT16;
  Item: pTW_FIX32;
  i : Integer;
begin
  {Obtain handle to data from this capability}
  Result := GetCapabilityRec(Capability, Handle, Mode, Container);
  if Result = crSuccess then
  begin
    {Obtain data}
    //npeter
    //the "if" is just for sure!
    if (Container<>TWON_ENUMERATION) and (Container<>TWON_ARRAY) then
    begin
      result:=crUnsupported;
      exit;
    end;

    EnumV := GlobalLock(Handle);
    if EnumV^.ItemType <> TWTY_FIX32 then Result := crUnsupported
    else begin
      {Set array size and pointer to the first item}
      Item := @EnumV^.ItemList[0];
      SetLength(Values, EnumV^.NumItems);
      {Fill array}
      FOR i := 1 TO EnumV^.NumItems DO
      begin
        {Fill array with the item}
        Values[i - 1] := Fix32ToFloat(Item^);
        {Move to next item}
        inc(Item);
      end {FOR i};

      {Fill return}

      //npeter
      //DefaultIndex and CurrentIndex valid for enum only!
      //I got nice AV with an old Mustek scanner which uses TWON_ARRAY
      //i return 0 in this case (may be not the best solution, but not AV at least :-)
      if (Container<>TWON_ARRAY) then
      begin
        if Mode = rcGetDefault then Return := Values[EnumV^.DefaultIndex]
        else Return := Values[EnumV^.CurrentIndex];
      end;
    end;
  end;
end;

```

```

        end
        else return:=0;
    end;
    {Free data}
    GlobalUnlock(Handle);
    GlobalFree(Handle);
end;
end;

{Sets X resolution}
function TTWainSource.SetIXResolution(Value: Extended): TCapabilityRet;
var
    Fix32: TW_FIX32;
begin
    Fix32 := FloatToFix32(Value);
    Result := SetOneValue(ICAP_XRESOLUTION, TWTY_FIX32, @Fix32);
end;

{Sets Y resolution}
function TTWainSource.SetIYResolution(Value: Extended): TCapabilityRet;
var
    Fix32: TW_FIX32;
begin
    Fix32 := FloatToFix32(Value);
    Result := SetOneValue(ICAP_YRESOLUTION, TWTY_FIX32, @Fix32);
end;

{Returns X resolution}
function TTWainSource.GetIXResolution(var Return: Extended;
    var Values: TTWainResolution; Mode: TRetrieveCap): TCapabilityRet;
begin
    Result := GetResolution(ICAP_XRESOLUTION, Return, Values, Mode);
end;

{Returns Y resolution}
function TTWainSource.GetIYResolution(var Return: Extended;
    var Values: TTWainResolution; Mode: TRetrieveCap): TCapabilityRet;
begin
    Result := GetResolution(ICAP_YRESOLUTION, Return, Values, Mode);
end;

{Returns if user interface is controllable}
function TTWainSource.GetUIControllable(var Return: Boolean): TCapabilityRet;
var
    ItemType: TW_UINT16;
    Value : String;
begin
    {Try to obtain value and make sure it is of type TW_BOOL}

```

```

Result := GetOneValue(CAP_UICONTROLLABLE, ItemType, Value, rcGet);
if (Result = crSuccess) and (ItemType <> TWTY_BOOL) then
    Result := crUnsupported;
{Return value, by checked the return value from GetOneValue}
if Result = crSuccess then Return := (Value = '1');
end;

```

```

{Returns if feeder is loaded}
function TTWainSource.GetFeederLoaded(var Return: Boolean): TCapabilityRet;
var
    ItemType: TW_UINT16;
    Value : String;
begin
    {Try to obtain value and make sure it is of type TW_BOOL}
    Result := GetOneValue(CAP_FEEDERLOADED, ItemType, Value, rcGet);
    if (Result = crSuccess) and (ItemType <> TWTY_BOOL) then
        Result := crUnsupported;
    {Return value, by checked the return value from GetOneValue}
    if Result = crSuccess then Return := (Value = '1');
end;

```

```

{Returns if feeder is enabled}
function TTWainSource.GetFeederEnabled(var Return: Boolean): TCapabilityRet;
var
    ItemType: TW_UINT16;
    Value : String;
begin
    {Try to obtain value and make sure it is of type TW_BOOL}
    Result := GetOneValue(CAP_FEEDERENABLED, ItemType, Value, rcGet);
    if (Result = crSuccess) and (ItemType <> TWTY_BOOL) then
        Result := crUnsupported;
    {Return value, by checked the return value from GetOneValue}
    if Result = crSuccess then Return := (Value = '1');
end;

```

```

{Set if feeder is enabled}
function TTWainSource.SetFeederEnabled(Value: WordBool): TCapabilityRet;
begin
    {Call SetOneValue to set value}
    Result := SetOneValue(CAP_FEEDERENABLED, TWTY_BOOL, @Value);
end;

```

```

{Returns if autofeed is enabled}
function TTWainSource.GetAutofeed(var Return: Boolean): TCapabilityRet;
var
    ItemType: TW_UINT16;
    Value : String;

```

```

begin
  {Try to obtain value and make sure it is of type TW_BOOL}
  Result := GetOneValue(CAP_AUTOFEED, ItemType, Value, rcGet);
  if (Result = crSuccess) and (ItemType <> TWTY_BOOL) then
    Result := crUnsupported;
  {Return value, by checked the return value from GetOneValue}
  if Result = crSuccess then Return := (Value = '1');
end;

{Set if autofeed is enabled}
function TTWainSource.SetAutoFeed(Value: WordBool): TCapabilityRet;
begin
  {Call SetOneValue to set value}
  Result := SetOneValue(CAP_AUTOFEED, TWTY_BOOL, @Value);
end;

{Used with property PendingXfers}
function TTWainSource.GetPendingXfers: TW_INT16;
var
  PendingXfers: TW_PENDINGXFERS;
begin
  if Loaded and Enabled then
    begin
      {Call method to retrieve}
      if Owner.TwainProc(AppInfo, @Structure, DG_CONTROL, DAT_PENDINGXFERS,
        MSG_GET, @PendingXfers) = TWRC_SUCCESS then
        Result := PendingXfers.Count
      else Result := ERROR_INT16; {Some error occurred while calling message}
    end
  else Result := ERROR_INT16; {Source not loaded/enabled}
end;

{Returns a TMsg structure}
function MakeMsg(const Handle: THandle; uMsg: UINT; wParam: WPARAM;
  lParam: LPARAM): TMsg;
begin
  {Fill structure with the parameters}
  Result.hwnd := Handle;
  Result.message := uMsg;
  Result.wParam := wParam;
  Result.lParam := lParam;
  GetCursorPos(Result.pt);
end;

{Virtual window procedure handler}
function VirtualWinProc(Handle: THandle; uMsg: UINT; wParam: WPARAM;
  lParam: LPARAM): LResult; stdcall;

```

```

{Returns the TDelphiTwain object}
function Obj: TDelphiTwain;
begin
  Longint(Result) := GetWindowLong(Handle, GWL_USERDATA);
end {function};

var
  Twain: TDelphiTwain;
  i   : Integer;
  Msg  : TMsg;
begin
  {Tests for the message}
  case uMsg of
    {Creation of the window}
    WM_CREATE:
      {Stores the TDelphiTwain object handle}
      with pCreateStruct(IParam)^ do
        SetWindowLong(Handle, GWL_USERDATA, Longint(lpCreateParams));
    {case} else
      begin
        {Try to obtain the current object pointer}
        Twain := Obj;

        if Assigned(Twain) then
          {If there are sources loaded, we need to verify}
          {this message}
          if (Twain.SourcesLoaded > 0) then
            begin
              {Convert parameters to a TMsg}
              Msg := MakeMsg(Handle, uMsg, wParam, lParam);
              {Tell about this message}
              FOR i := 0 TO Twain.SourceCount - 1 DO
                if ((Twain.Source[i].Loaded) and (Twain.Source[i].Enabled)) then
                  if Twain.Source[i].ProcessMessage(Msg) then
                    begin
                      {Case this was a message from the source, there is}
                      {no need for the default procedure to process}
                      Result := 0;
                      Exit;
                    end;
                  end;
                end;
              end;

            end {if (Twain.SourcesLoaded > 0)}

          end {case Else}
        end {case uMsg of};

```

```

{Calls method to handle}
Result := DefWindowProc(Handle, uMsg, wParam, lParam);
end;

//npeter: 2004.01.12
//sets the acquired area
function TTWainSource.SetImagelayoutFrame(const fLeft, fTop, fRight,
  fBottom: double): TCapabilityRet;
var ImageLayout: TW_IMAGELAYOUT;
begin
  if not Loaded then
    begin
      Result := crInvalidState; {In case the source is not loaded}
      exit;
    end;

  fillchar(ImageLayout,sizeof(TW_IMAGELAYOUT),0);
  with ImageLayout.Frame do
    begin
      Left:=FloatToFIX32(fLeft);
      Top:=FloatToFIX32(fTop);
      Right:=FloatToFIX32(fRight);
      Bottom:=FloatToFIX32(fBottom);
    end;
  {Call method and store return}
  Result := ResultToCapabilityRec(Owner.TwainProc(AppInfo, @Structure,
    DG_IMAGE, DAT_IMAGELAYOUT, MSG_SET, @ImageLayout));
end;

//npeter: 2004.01.12
//enable/disable progress indicators
function TTWainSource.SetIndicators(Value: boolean): TCapabilityRet;
begin
  {Call SetOneValue to set value}
  Result := SetOneValue(CAP_INDICATORS, TWTY_BOOL, @Value);
end;

{Information for the virtual window class}
var
  VirtualWinClass: TWNDClass;

initialization
  {Registers the virtual window class}
  VirtualWinClass.hInstance := hInstance;
  VirtualWinClass.style := 0;

```

```

VirtualWinClass.lpfWndProc := @VirtualWinProc;
VirtualWinClass.cbClsExtra := 0;
VirtualWinClass.cbWndExtra := 0;
VirtualWinClass.hIcon := 0;
VirtualWinClass.hCursor := 0;
VirtualWinClass.hbrBackground := COLOR_WINDOW + 1;
VirtualWinClass.lpszMenuName := '';
VirtualWinClass.lpszClassName := VIRTUALWIN_CLASSNAME;
Windows.RegisterClass(VirtualWinClass);
finalization
  {Unregisters the virtual window class}
  Windows.UnregisterClass(VIRTUALWIN_CLASSNAME, hInstance);
end.

```

```

{GENERAL METHODS USED BY TWAIN DELPHI}
{december 2001®, made by Gustavo Daud}

```

```

{This unit contains general methods used by Delphi}
{Twain component. Some of the methods bellow aren't}
{directly related to Twain, but are pieces needed}
{to implement the component.}

```

```
unit DelphiTwainUtils;
```

```
{$INCLUDE DELPHITWAIN.INC}
```

```
interface
```

```
uses
```

```
    Twain;
```

```
type
```

```
{Kinds of directories to be obtained with GetCustomDirectory}
```

```
TDirectoryKind = (dkWindows, dkSystem, dkCurrent, dkApplication, dkTemp);
```

```
{Class to store a list of pointers}
```

```
TPointerList = class
```

```
private
```

```
    {Stores pointer to the allocated data}
```

```
    Data: Pointer;
```

```
    {Contains number of additional items allocated every time}
```

```
    {it needs more data to store}
```

```
    fAdditionalBlock: Integer;
```

```
    {Contains the number of items in the list}
```

```
    fCount: Integer;
```

```
    {Contains number of allocated items}
```

```
    fAllocated: Integer;
```

```
    {Allocate/deallocate memory to have enough memory}
```

```
    {to hold the new number of items}
```

```
    procedure SetAllocated(const Value: Integer);
```

```
    {Sets the AdditionalBlock property}
```

```
    procedure SetAdditionalBlock(const Value: Integer);
```

```
    {Set the number of items in the list}
```

```
    procedure SetCount(const Value: Integer);
```

```
    function GetItem(Index: Integer): Pointer;
```

```
    procedure PutItem(Index: Integer; const Value: Pointer);
```

```
public
```

```
    {Add a new item}
```

```
    procedure Add(Value: Pointer);
```

```
    {Clear all the items in the list}
```

```
    procedure Clear;
```

```
    {Object being created or destroyed}
```

```
    constructor Create;
```

```
    destructor Destroy; override;
```

```
    {Returns/sets an item value}
```

```
    property Item[Index: Integer]: Pointer read GetItem write PutItem; default;
```

```
    {Returns the number of items}
```

```
    property Count: Integer read fCount write SetCount;
```

```
    {Number of allocated items}
```

```

property Allocated: Integer read fAllocated write SetAllocated;
{Additional items to alloc when it needs more memory}
property AdditionalBlock: Integer read fAdditionalBlock write
    SetAdditionalBlock;
end;

```

```

{Returns custom Microsoft Windows® directories}
function GetCustomDirectory(const DirectoryKind: TDirectoryKind): String;
{Returns the last error string from Microsoft Windows®}
function GetLastErrorText(): String;
{Returns if the directory exists}
function DirectoryExists(const Directory: String): Boolean;
{Returns if the file exists}
function FileExists(const FilePath: String): Boolean;
{Extracts the file directory part}
function ExtractDirectory(const FilePath: String): String;
{Convert from integer to string}
{$IFDEF DONTUSEVCL}function IntToStr(Value: Integer): String;{$ENDIF}
{$IFDEF DONTUSEVCL}function StrToIntDef(Value: String;
    Default: Integer): Integer;{$ENDIF}
{$IFDEF DONTUSEVCL}function CompareMem(P1, P2: pChar;
    Size: Integer): Boolean;{$ENDIF}
{Convert from twain Fix32 to extended}
function Fix32ToFloat(Value: TW_FIX32): Extended;
{Convert from extended to Fix32}
function FloatToFix32 (floater: extended): TW_FIX32;

```

implementation

{Units used bellow}

uses

Windows;

```

{$IFDEF DONTUSEVCL}
function CompareMem(P1, P2: pChar; Size: Integer): Boolean;
var
    i: Integer;
begin
    {Default result}
    Result := TRUE;
    {Search each byte}
    FOR i := 1 TO Size DO
    begin
        {Compare booth bytes}
        if P1^ <> P2^ then
        begin
            Result := FALSE;
            Exit;
        end;
    end;
end;

```

```

    end; {if P1^ <> P2^}
    {Move to next byte}
    Inc(P1); Inc(P2);
  end {FOR i}
end {function};
{$ENDIF}

```

```

{$IFDEF DONTUSEVCL}
function IntToStr(Value: Integer): String;
begin
  Str(Value, Result);
end;
{$ENDIF}

```

```

{$IFDEF DONTUSEVCL}
function StrToIntDef(Value: String; Default: Integer): Integer;
var Code: Integer;
begin
  {Try converting from string to integer}
  Val(Value, Result, Code);
  {If any error occurred, returns default value}
  if Code <> 0 then Result := Default;
end;
{$ENDIF}

```

```

{Convert from extended to Fix32}
function FloatToFix32 (floater: extended): TW_FIX32;
var
  fracpart : extended;
begin
  //Obtain numerical part by truncating the float number
  Result.Whole := trunc(floater);
  //Obtain fracional part by subtracting float number by
  //numerical part. Also we make sure the number is not
  //negative by multipling by -1 if it is negative
  fracpart := floater - result.Whole;
  if fracpart < 0 then fracpart := fracpart * -1;
  //Multiply by 10 until there is no fracional part any longer
  while FracPart - trunc(FracPart) <> 0 do fracpart := fracpart * 10;
  //Return fracional part
  Result.Frac := trunc(fracpart);
end;

```

```

{Convert from twain Fix32 to extended}
function Fix32ToFloat(Value: TW_FIX32): Extended;
begin
  Result := Value.Whole + (Value.Frac / 65536.0);
end;

```

end;

{Returns the last position for any of the characters in the parameter}

function LastPosition(const Text, characters: String): Integer;

var

x, y: Integer; {For loop variables}

begin

Result := Length(Text); {Initial result}

{Search each character in the text}

FOR x := 1 TO Length(Text) DO

begin

{Test for each character}

FOR y := 1 TO Length(characters) DO

if Text[x] = characters[y] then

Result := x;

end {for x}

end;

{Extracts the file directory}

function ExtractDirectory(const FilePath: String): String;

begin

{Searches for the last \ or : characters}

{ex: c:\windows\system32\yfile.ext or c:autoexec.bat}

Result := Copy(FilePath, 1, LastPosition(FilePath, '\:'));

end;

{Returns if the file exists}

function FileExists(const FilePath: String): Boolean;

var

FindData : TWin32FindData;

FindHandle: THandle;

begin

{Searches for the file}

FindHandle := FindFirstFile(PChar(FilePath), FindData);

Result := (FindHandle <> INVALID_HANDLE_VALUE);

{In case it found, closes the FindFirstFile handle}

if Result then FindClose(FindHandle);

end;

{Returns if the directory exists}

function DirectoryExists(const Directory: String): Boolean;

var

Attr: DWORD;

begin

{Calls GetFileAttributes to verify}

Attr := GetFileAttributes(PChar(Directory));

Result := (Attr <> \$FFFFFFFF) and (Attr and FILE_ATTRIBUTE_DIRECTORY <> 0);

end;

{Makes an language identifier using the two ids}

function MAKELANGID(p, s: WORD): DWORD;

begin

Result := (s shl 10) or p;

end;

{Returns the last error string from Microsoft Windows®}

function GetLastErrorText(): String;

var

Buffer: Array[Byte] of Char;

Len : DWORD;

begin

{Calls format message to translate from the error code ID to}

{a text understandable error}

Len := Windows.FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM or

FORMAT_MESSAGE_ARGUMENT_ARRAY, nil, GetLastError(),

MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), Buffer, sizeof(Buffer), nil);

{Remove this chars from the ending of the result}

while (Len > 0) and (Buffer[Len - 1] in [#0..#32, '.']) do Dec(Len);

{Fills result}

SetString(Result, Buffer, Len);

end;

{Includes a trailing backslash in the end of the directory; if necessary}

procedure IncludeTrailingBackslash(var Directory: String);

begin

{If there isn't already a backslash, add one}

if Directory[Length(Directory)] <> '\' then

Directory := Directory + '\'

end;

{Returns custom Microsoft Windows® directories}

function GetCustomDirectory(const DirectoryKind: TDirectoryKind): String;

const

{Default maximum size for directories}

DEF_DIRLEN = MAX_PATH;

{Calls appropriate method and returns necessary size}

function CallDirectoryMethod(Buffer: Pointer; Size: UINT): UINT;

begin

{Test the directory needed by the parameter}

case DirectoryKind of

{Windows directory}

dkWindows: Result := Windows.GetWindowsDirectory(Buffer, Size);

{System directory}

dkSystem : Result := Windows.GetSystemDirectory(Buffer, Size);

```

    {Current directory}
    dkCurrent: Result := Windows.GetCurrentDirectory(Size, Buffer);
    {Application directory}
    dkApplication: Result := Windows.GetModuleFileName(0, Buffer, Size);
    {Temp directory}
    dkTemp : Result := Windows.GetTempPath(Size, Buffer);
    {Unknown directory}
    else Result := 0;
  end {case}
end;

```

```

var
  DirectoryLen: UINT;
begin
  {Set length of the resulting buffer to MAX_PATH to try to hold}
  {windows directory}
  SetLength(Result, DEF_DIRLEN + 1);
  {Tries to obtain the windows directory and stores the size}
  DirectoryLen := CallDirectoryMethod(@Result[1], DEF_DIRLEN);

```

```

  {In case it was not enough to hold windows directory, enlarge}
  if DirectoryLen > DEF_DIRLEN then
    begin
      {Try again, now with the right size}
      SetLength(Result, DirectoryLen + 1);
      CallDirectoryMethod(@Result[1], DirectoryLen);
    end
  else {Otherwise, adjust the result to excluded unused data}
    SetLength(Result, DirectoryLen);

```

```

  {In case the user searched for the application directory}
  {extracts just the directory part}
  if DirectoryKind = dkApplication then
    Result := ExtractDirectory(Result);
  {Add a trailing backslash to end of the directory name}
  IncludeTrailingBackslash(Result);
end;

```

```

{ TPointerList object implementation }

```

```

{Add a new item}
procedure TPointerList.Add(Value: Pointer);
begin
  {Increase number of items and update new item}
  Count := Count + 1;
  Item[Count - 1] := Value;
end;

```

```

{Clear all the items in the list}
procedure TPointerList.Clear;
begin
  {Set number of items to 0 and initialize again allocated items}
  Count := 0;
  Allocated := AdditionalBlock;
end;

{TPointerList being created}
constructor TPointerList.Create;
begin
  {Let ancestor receive the call}
  inherited Create;

  {Allocate a number of items}
  fAdditionalBlock := 10;
  fAllocated := fAdditionalBlock;
  GetMem(Data, (fAllocated * sizeof(Pointer)));
end;

{TPointerList being destroyed}
destructor TPointerList.Destroy;
begin
  {Deallocate data}
  FreeMem(Data, (fAllocated * sizeof(Pointer)));

  {Let ancestor receive and finish}
  inherited Destroy;
end;

{Returns an item from the list}
function TPointerList.GetItem(Index: Integer): Pointer;
begin
  {Check item bounds and return item}
  if Index in [0..Count - 1] then
    Longint(Result) := pLongint(Longint(Data) + (Index * sizeof(Pointer)))^
  else Result := nil; {Otherwise returns nil}
end;

{Sets an item from the list}
procedure TPointerList.PutItem(Index: Integer; const Value: Pointer);
begin
  {Check item bounds and sets item}
  if Index in [0..Count - 1] then
    pLongint(Longint(Data) + (Index * sizeof(Pointer)))^ := Longint(Value);
end;

{Sets the AdditionalBlock property}

```

```

procedure TPointerList.SetAdditionalBlock(const Value: Integer);
begin
  {Value must be a positive number greater than 0}
  if (Value > 0) then
    fAdditionalBlock := Value;
end;

```

```

{Allocate/deallocate memory to have enough memory to hold}
{the new number of items}
procedure TPointerList.SetAllocated(const Value: Integer);
begin
  {Must be always greater than 0 the number of allocated items}
  {And it also should not be smaller than count}
  if (Value > 0) and (Value <= Count) then
    begin
      {Just realloc memory and update property variable}
      ReallocMem(Data, (Value * sizeof(Pointer)));
      fAllocated := Value;
    end {if (Value <> 0)}
end;

```

```

{Set the number of items in the list}
procedure TPointerList.SetCount(const Value: Integer);
begin
  {Value must be 0 or greater}
  if (Value >= 0) then
    begin
      {If there is no more memory to hold data, allocate some more}
      while (Value > fAllocated) do
        Allocated := Allocated + fAdditionalBlock;
      {Update property}
      fCount := Value;
    end {if (Value >= 0)}
end;

```

```

end.

```

```

{*****}
{
{ Borland Delphi Runtime Library
{ Twain interface unit
{
{ Portions created by TWAIN Working Group,
{ see Copyright statement from original file below
{
{ The original file is: twain.h, released March 15, 2000.
{ The original Pascal code is: twain.pas, released 20. Dez 1999.
{ The initial developer of the Pascal code is: Uli Tessel (UT)
{ (UliTessel@swol.de) with help of Matthias Thoma (MT)
{ (ma.thoma@gmx.de)
{ Translation cleaned up and updated to twain 1.9 by:
{ Martin Olsson (MO), mnemo@home.se
{
{ Obtained through:
{ Joint Endeavour of Delphi Innovators (Project JEDI)
{
{ You may retrieve the latest version of this file at the Project
{ JEDI home page, located at http://delphi-jedi.org
{
{ The contents of this file are used with permission, subject to
{ the Mozilla Public License Version 1.1 (the "License"); you may
{ not use this file except in compliance with the License. You may
{ obtain a copy of the License at
{ http://www.mozilla.org/MPL/MPL-1.1.html
{
{ Software distributed under the License is distributed on an
{ "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or
{ implied. See the License for the specific language governing
{ rights and limitations under the License.
{
{*****}

{ =====
=====

```

Copyright (C) 1991, 1992 TWAIN Working Group: Aldus, Caere, Eastman-Kodak, Hewlett-Packard and Logitech Corporations. All rights reserved.

Copyright (C) 1997 TWAIN Working Group: Bell+Howell, Canon, DocuMagix, Fujitsu, Genoa Technology, Hewlett-Packard, Kofax Imaging Products, and Ricoh Corporation. All rights reserved.

Copyright © 1998 TWAIN Working Group: Adobe Systems Incorporated, Canon Information Systems, Eastman Kodak Company, Fujitsu Computer Products of America, Genoa Technology,

Hewlett-Packard Company, Intel Corporation, Kofax Image Products,
JFL Peripheral Solutions Inc., Ricoh Corporation, and Xerox Corporation.
All rights reserved.

Copyright © 2000 TWAIN Working Group: Adobe Systems Incorporated,
Canon Information Systems, Digimarc Corporation, Eastman Kodak Company,
Fujitsu Computer Products of America, Hewlett-Packard Company,
JFL Peripheral Solutions Inc., Ricoh Corporation, and Xerox Corporation.
All rights reserved.

TWAIN.h - This is the definitive include file for applications and
data sources written to the TWAIN specification.
It defines constants, data structures, messages etc.
for the public interface to TWAIN.

Revision History:

version 1.0, March 6, 1992. TWAIN 1.0.

version 1.1, January 1993. Tech Notes 1.1

version 1.5, June 1993. Specification Update 1.5

Change DC to TW

Change filename from DC.H to TWAIN.H

version 1.5, July 1993. Remove spaces from country identifiers

version 1.7, July 1997 Added Capabilities and data structure for
document imaging and digital cameras.

KHL.

version 1.7, July 1997 Inserted Borland compatible structure packing
directives provided by Mentor. JMH

version 1.7, Aug 1997 Expanded file tabs to spaces.

NOTE: future authors should be sure to have
their editors set to automatically expand tabs
to spaces (original tab setting was 4 spaces).

version 1.7, Sept 1997 Added job control values

Added return codes

version 1.7, Sept 1997 changed definition of pRGBRESPONSE to
pTW_RGBRESPONSE

version 1.7 Aug 1998 Added missing TWEI_BARCODEROTATION values
TWBCOR_ types JMH

version 1.8 August 1998 Added new types and definitions required
for 1.8 Specification JMH

version 1.8 January 1999 Changed search mode from SRCH_ to TWBD_ as
in 1.8 Specification, added TWBT_MAXICODE JMH

version 1.8 January 1999 Removed undocumented duplicate AUTO<cap> JMH

version 1.8 March 1999 Removed undocumented 1.8 caps:

CAP_FILESYSTEM

CAP_PAPERBINDING

CAP_PASSTHRU

CAP_POWERDOWNTIME

ICAP_AUTODISCARDBLANKPAGES

* CAP_PAGEMULTIPLEACQUIRE - is CAP_REACQUIREALLOWED,
requires spec change. JMH

Added Mac structure packing modifications JMH

version 1.9 March 2000 Added new types and definitions required
for 1.9 Specification MLM

version 1.9 March 2000 Added ICAP_JPEGQUALITY, TWJQ_ values,
updated TWON_PROTOCOLMINOR for Release v1.9 MN

=====
==== }

{
Revision History for translation:

Version 1.8.0: 29.08.99 - UT
Initial translation, based on twain.h, version 1.8

Version 1.8.1: 12.09.99 - UT
SizeOf for all structures checked and corrected.
(Alignment is 2 Bytes for the C Code and 'packed record' uses
1 Byte alignment. Only types using TW_xINT8 are affected)

Version 1.8.2: 19.12.99 UT
Added MPL and the other JEDI Statements
Added EXTERNALSYMS to support C++ Builder
Created the .PAR file (no Unicode things are used by the TWAIN API?)
A bit better formatting of the source

Version 1.8.3: 20.12.99
MT: Added Delphi-Aliases to the structures (like TTWFrame = TW_FRAME)
UT: Added missing Externalyms for some constants

Version 1.9.0: 01.12.00
MO: Updated translation to conform with twain.h 1.9
MO: Cleaned up style to to fit JEDI standards

}

```

unit Twain;

interface

{$HPPEMIT '#include <twain.h>' }

uses
  Windows;

{*****
 * TWAIN Version
***** }

const
  TWON_PROTOCOLMINOR = 9;  { Changed for Version 1.9 }
  {$EXTERNALSYM TWON_PROTOCOLMINOR}
  TWON_PROTOCOLMAJOR = 1;
  {$EXTERNALSYM TWON_PROTOCOLMAJOR}

{*****
 * Platform Dependent Definitions and Typedefs
***** }

type
  TW_HANDLE = THandle;
  {$EXTERNALSYM TW_HANDLE}
  TTWHandle = TW_HANDLE;
  TW_MEMREF = Pointer;
  {$EXTERNALSYM TW_MEMREF}
  TTWMemRef = TW_MEMREF;

{*****
 * Type Definitions
***** }

{ String types. These include room for the strings and a NULL char, *
 * or, on the Mac, a length byte followed by the string. *
 * TW_STR255 must hold less than 256 chars so length fits in first byte. }

type
  TW_STR32 = array[0..33] of Char; // char TW_STR32[34]
  {$EXTERNALSYM TW_STR32}
  pTW_STR32 = ^TW_STR32;
  {$EXTERNALSYM pTW_STR32}
  TTWStr32 = TW_STR32;
  PTWStr32 = pTW_STR32;

  TW_STR64 = array[0..65] of Char; // char TW_STR64[66]

```

```

{$EXTERNALSYM TW_STR64}
pTW_STR64 = ^TW_STR64;
{$EXTERNALSYM pTW_STR64}
TTWStr64 = TW_STR64;
PTWStr64 = pTW_STR64;

```

```

TW_STR128 = array[0..129] of Char; // char TW_STR128[130]
{$EXTERNALSYM TW_STR128}
pTW_STR128 = ^TW_STR128;
{$EXTERNALSYM pTW_STR128}
TTWStr128 = TW_STR128;
PTWStr128 = pTW_STR128;

```

```

TW_STR255 = array[0..255] of Char; // char TW_STR255[256]
{$EXTERNALSYM TW_STR255}
pTW_STR255 = ^TW_STR255;
{$EXTERNALSYM pTW_STR255}
TTWStr255 = TW_STR255;
PTWStr255 = pTW_STR255;

```

```

TW_STR1024 = array[0..1025] of Char; // char TW_STR1024[1026]
{$EXTERNALSYM TW_STR1024}
pTW_STR1024 = ^TW_STR1024;
{$EXTERNALSYM pTW_STR1024}
TTWStr1024 = TW_STR1024; // added 1.9
PTWStr1024 = pTW_STR1024;

```

```

TW_UNI512 = array[0..511] of WideChar; // wchar_t TW_UNI512[512]
{$EXTERNALSYM TW_UNI512}
pTW_UNI512 = ^TW_UNI512;
{$EXTERNALSYM pTW_UNI512}
TTWUni512 = TW_UNI512; // added 1.9
PTWUni512 = pTW_UNI512;

```

```

{ Numeric types. }
TW_INT8 = ShortInt; // char TW_INT8
{$EXTERNALSYM TW_INT8}
pTW_INT8 = ^TW_INT8;
{$EXTERNALSYM pTW_INT8}
TTWInt8 = TW_INT8;
PTWInt8 = pTW_INT8;

```

```

TW_INT16 = SmallInt; // short TW_INT16
{$EXTERNALSYM TW_INT16}
pTW_INT16 = ^TW_INT16;
{$EXTERNALSYM pTW_INT16}
TTWInt16 = TW_INT16;
PTWInt16 = pTW_INT16;

```

```

TW_INT32 = LongInt;    // long TW_INT32
{$EXTERNALSYM TW_INT32}
pTW_INT32 = ^TW_INT32;
{$EXTERNALSYM pTW_INT32}
TTWInt32 = TW_INT32;
PTWInt32 = pTW_INT32;

```

```

TW_UINT8 = Byte;      // unsigned char TW_UINT8
{$EXTERNALSYM TW_UINT8}
pTW_UINT8 = ^TW_UINT8;
{$EXTERNALSYM pTW_UINT8}
TTWUInt8 = TW_UINT8;
PTWUInt8 = pTW_UINT8;

```

```

TW_UINT16 = Word;     // unsigned short TW_UINT16
{$EXTERNALSYM TW_UINT16}
pTW_UINT16 = ^TW_UINT16;
{$EXTERNALSYM pTW_UINT16}
TTWUInt16 = TW_UINT16;
PTWUInt16 = pTW_UINT16;

```

```

TW_UINT32 = ULONG;    // unsigned long TW_UINT32
{$EXTERNALSYM TW_UINT32}
pTW_UINT32 = ^TW_UINT32;
{$EXTERNALSYM pTW_UINT32}
TTWUInt32 = TW_UINT32;
PTWUInt32 = pTW_UINT32;

```

```

TW_BOOL = WordBool;   // unsigned short TW_BOOL
{$EXTERNALSYM TW_BOOL}
pTW_BOOL = ^TW_BOOL;
{$EXTERNALSYM pTW_BOOL}
TTWBool = TW_BOOL;
PTWBool = pTW_BOOL;

```

```

{ Fixed point structure type. }
type

```

```

  TW_FIX32 = packed record
    Whole: TW_INT16; { maintains the sign }
    Frac: TW_UINT16;
  end;
{$EXTERNALSYM TW_FIX32}
pTW_FIX32 = ^TW_FIX32;
{$EXTERNALSYM pTW_FIX32}

```

```

TTWFix32 = TW_FIX32;
PTWFix32 = pTW_FIX32;

```

```

{*****
* Structure Definitions
*****}

{ No DAT needed. }
type
  TW_CIEPOINT = packed record
    X: TW_FIX32;
    Y: TW_FIX32;
    Z: TW_FIX32;
  end;
{$EXTERNALSYM TW_CIEPOINT}
pTW_CIEPOINT = ^TW_CIEPOINT;
{$EXTERNALSYM pTW_CIEPOINT}

TTWCiePoint = TW_CIEPOINT;
PTWCiePoint = pTW_CIEPOINT;

{ No DAT needed. }
  TW_DECODEFUNCTION = packed record
    StartIn: TW_FIX32;
    BreakIn: TW_FIX32;
    EndIn: TW_FIX32;
    StartOut: TW_FIX32;
    BreakOut: TW_FIX32;
    EndOut: TW_FIX32;
    Gamma: TW_FIX32;
    SampleCount: TW_FIX32; { if =0 use the gamma }
  end;
{$EXTERNALSYM TW_DECODEFUNCTION}
pTW_DECODEFUNCTION = ^TW_DECODEFUNCTION;
{$EXTERNALSYM pTW_DECODEFUNCTION}

TTWDecodeFunction = TW_DECODEFUNCTION;
PTWDecodeFunction = pTW_DECODEFUNCTION;

{ No DAT needed. }
  TW_ELEMENT8 = packed record
    Index: TW_UINT8; { Value used to index into the color table. }
    Channel1: TW_UINT8; { First tri-stimulus value (e.g Red) }
    Channel2: TW_UINT8; { Second tri-stimulus value (e.g Green) }
    Channel3: TW_UINT8; { Third tri-stimulus value (e.g Blue) }
  end;
{$EXTERNALSYM TW_ELEMENT8}
pTW_ELEMENT8 = ^TW_ELEMENT8;

```

```
{ $EXTERNALSYM pTW_ELEMENT8 }
```

```
TTWElement8 = TW_ELEMENT8;  
PTWElement8 = pTW_ELEMENT8;
```

```
{ No DAT. Defines a frame rectangle in ICAP_UNITS coordinates. }
```

```
  TW_FRAME = packed record
```

```
    Left: TW_FIX32;
```

```
    Top: TW_FIX32;
```

```
    Right: TW_FIX32;
```

```
    Bottom: TW_FIX32;
```

```
  end;
```

```
{ $EXTERNALSYM TW_FRAME }
```

```
pTW_FRAME = ^TW_FRAME;
```

```
{ $EXTERNALSYM pTW_FRAME }
```

```
PTWFrame = pTW_FRAME;
```

```
TTWFrame = TW_FRAME;
```

```
{ No DAT needed. Used to manage memory buffers. }
```

```
  TW_MEMORY = packed record
```

```
    Flags: TW_UINT32; { Any combination of the TWMF_constants. }
```

```
    Length: TW_UINT32; { Number of bytes stored in buffer TheMem. }
```

```
    TheMem: TW_MEMREF; { Pointer or handle to the allocated memory buffer. }
```

```
  end;
```

```
{ $EXTERNALSYM TW_MEMORY }
```

```
pTW_MEMORY = ^TW_MEMORY;
```

```
{ $EXTERNALSYM pTW_MEMORY }
```

```
TTWMemory = TW_MEMORY;
```

```
PTWMemory = pTW_MEMORY;
```

```
{ No DAT needed. }
```

```
  TW_TRANSFORMSTAGE = packed record
```

```
    Decode: array[0..2] of TW_DECODEFUNCTION;
```

```
    Mix: array[0..2, 0..2] of TW_FIX32;
```

```
  end;
```

```
{ $EXTERNALSYM TW_TRANSFORMSTAGE }
```

```
pTW_TRANSFORMSTAGE = ^TW_TRANSFORMSTAGE;
```

```
{ $EXTERNALSYM pTW_TRANSFORMSTAGE }
```

```
TTWTransformStage = TW_TRANSFORMSTAGE;
```

```
PTWTransformStage = pTW_TRANSFORMSTAGE;
```

```
{ No DAT needed. Describes version of software currently running. }
```

```
  TW_VERSION = packed record
```

```

MajorNum: TW_UINT16; { Major revision number of the software. }
MinorNum: TW_UINT16; { Incremental revision number of the software. }
Language: TW_UINT16; { e.g. TWLG_SWISSFRENCH }
Country: TW_UINT16; { e.g. TWCY_SWITZERLAND }
Info: TW_STR32; { e.g. "1.0b3 Beta release" }
end;
{$EXTERNALSYM TW_VERSION}
pTW_VERSION = ^TW_VERSION;
{$EXTERNALSYM pTW_VERSION}

PTWVersion = pTW_VERSION;
TTWVersion = TW_VERSION;

{ TWON_ARRAY. Container for array of values (a simplified TW_ENUMERATION) }
TW_ARRAY = packed record
  ItemType: TW_UINT16;
  NumItems: TW_UINT32; { How many items in ItemList }
  ItemList: array[0..1] of TW_UINT8; { Array of ItemType values starts here }
  // UT: ..1 for alignment to 2 Byte Packing, so sizeof is correct
end;
{$EXTERNALSYM TW_ARRAY}
pTW_ARRAY = ^TW_ARRAY;
{$EXTERNALSYM pTW_ARRAY}

TTWArray = TW_ARRAY;
PTWArray = pTW_ARRAY;

{ TWON_ENUMERATION. Container for a collection of values. }
TW_ENUMERATION = packed record
  ItemType: TW_UINT16;
  NumItems: TW_UINT32; { How many items in ItemList }
  CurrentIndex: TW_UINT32; { Current value is in ItemList[CurrentIndex] }
  DefaultIndex: TW_UINT32; { Powerup value is in ItemList[DefaultIndex] }
  ItemList: array[0..1] of TW_UINT8; { Array of ItemType values starts here }
  // UT: ..1 for alignment to 2 Byte Packing, so sizeof is correct
end;
{$EXTERNALSYM TW_ENUMERATION}
pTW_ENUMERATION = ^TW_ENUMERATION;
{$EXTERNALSYM pTW_ENUMERATION}

TTWEnumeration = TW_ENUMERATION;
PTWEnumeration = pTW_ENUMERATION;

{ TWON_ONEVALUE. Container for one value. }
TW_ONEVALUE = packed record
  ItemType: TW_UINT16;
  Item: TW_UINT32;

```

```

end;
{$EXTERNALSYM TW_ONEVALUE}
pTW_ONEVALUE = ^TW_ONEVALUE;
{$EXTERNALSYM pTW_ONEVALUE}

```

```

TTWOneValue = TW_ONEVALUE;
PTWOneValue = pTW_ONEVALUE;

```

```

{ TWON_RANGE. Container for a range of values. }
TW_RANGE = packed record
  ItemType: TW_UINT16;
  MinValue: TW_UINT32; { Starting value in the range. }
  MaxValue: TW_UINT32; { Final value in the range. }
  StepSize: TW_UINT32; { Increment from MinValue to MaxValue. }
  DefaultValue: TW_UINT32; { Power-up value. }
  CurrentValue: TW_UINT32; { The value that is currently in effect. }
end;
{$EXTERNALSYM TW_RANGE}
pTW_RANGE = ^TW_RANGE;
{$EXTERNALSYM pTW_RANGE}

```

```

TTWRange = TW_RANGE;
PTWRange = pTW_RANGE;

```

```

{ DAT_CAPABILITY. Used by application to get/set capability from/in a data source. }
TW_CAPABILITY = packed record
  Cap: TW_UINT16; { id of capability to set or get, e.g. CAP_BRIGHTNESS }
  ConType: TW_UINT16; { TWON_ONEVALUE, _RANGE, _ENUMERATION or
_ARRAY }
  hContainer: TW_HANDLE; { Handle to container of type Dat }
end;
{$EXTERNALSYM TW_CAPABILITY}
pTW_CAPABILITY = ^TW_CAPABILITY;
{$EXTERNALSYM pTW_CAPABILITY}

```

```

TTWCapability = TW_CAPABILITY;
PTWCapability = pTW_CAPABILITY;

```

```

{ DAT_CIECOLOR. }
TW_CIECOLOR = packed record
  ColorSpace: TW_UINT16;
  LowEndian: TW_INT16;
  DeviceDependent: TW_INT16;
  VersionNumber: TW_INT32;
  StageABC: TW_TRANSFORMSTAGE;
  StageLMN: TW_TRANSFORMSTAGE;
  WhitePoint: TW_CIEPOINT;

```

```

    BlackPoint: TW_CIEPOINT;
    WhitePaper: TW_CIEPOINT;
    BlackInk: TW_CIEPOINT;
    Samples: array[0..0] of TW_FIX32;
end;
{$EXTERNALSYM TW_CIECOLOR}
pTW_CIECOLOR = ^TW_CIECOLOR;
{$EXTERNALSYM pTW_CIECOLOR}

TTWCieColor = TW_CIECOLOR;
PTWCieColor = pTW_CIECOLOR;

{ DAT_EVENT. For passing events down from the application to the DS. }
TW_EVENT = packed record
    pEvent: TW_MEMREF; { Windows pMSG or Mac pEvent. }
    TWMessage: TW_UINT16; { TW msg from data source, e.g. MSG_XFERREADY }
end;
{$EXTERNALSYM TW_EVENT}
pTW_EVENT = ^TW_EVENT;
{$EXTERNALSYM pTW_EVENT}

TTWEvent = TW_EVENT;
PTWEvent = pTW_EVENT;

{ DAT_GRAYRESPONSE }
TW_GRAYRESPONSE = packed record
    Response: array[0..0] of TW_ELEMENT8;
end;
{$EXTERNALSYM TW_GRAYRESPONSE}
pTW_GRAYRESPONSE = ^TW_GRAYRESPONSE;
{$EXTERNALSYM pTW_GRAYRESPONSE}

TTWGrayResponse = TW_GRAYRESPONSE;
PTWGrayResponse = pTW_GRAYRESPONSE;

{ DAT_IDENTITY. Identifies the program/library/code resource. }
TW_IDENTITY = packed record
    Id: TW_UINT32; { Unique number. In Windows, application hWnd }
    Version: TW_VERSION; { Identifies the piece of code }
    ProtocolMajor: TW_UINT16; { Application and DS must set to
TWON_PROTOCOLMAJOR }
    ProtocolMinor: TW_UINT16; { Application and DS must set to
TWON_PROTOCOLMINOR }
    SupportedGroups: TW_UINT32; { Bit field OR combination of DG_ constants }
    Manufacturer: TW_STR32; { Manufacturer name, e.g. "Hewlett-Packard" }
    ProductFamily: TW_STR32; { Product family name, e.g. "ScanJet" }
    ProductName: TW_STR32; { Product name, e.g. "ScanJet Plus" }
end;

```

```
{ $EXTERNALSYM TW_IDENTITY }
pTW_IDENTITY = ^TW_IDENTITY;
{ $EXTERNALSYM pTW_IDENTITY }
```

```
TTWIdentity = TW_IDENTITY;
PTWIdentity = pTW_IDENTITY;
```

```
{ DAT_IMAGEINFO. Application gets detailed image info from DS with this. }
TW_IMAGEINFO = packed record
  XResolution: TW_FIX32; { Resolution in the horizontal }
  YResolution: TW_FIX32; { Resolution in the vertical }
  ImageWidth: TW_INT32; { Columns in the image, -1 if unknown by DS }
  ImageLength: TW_INT32; { Rows in the image, -1 if unknown by DS }
  SamplesPerPixel: TW_INT16; { Number of samples per pixel, 3 for RGB }
  BitsPerSample: array[0..7] of TW_INT16; { Number of bits for each sample }
  BitsPerPixel: TW_INT16; { Number of bits for each padded pixel }
  Planar: TW_BOOL; { True if Planar, False if chunky }
  PixelType: TW_INT16; { How to interp data: ; photo interp (TWPT_) }
  Compression: TW_UINT16; { How the data is compressed (TWCP_xxxx) }
end;
{ $EXTERNALSYM TW_IMAGEINFO }
pTW_IMAGEINFO = ^TW_IMAGEINFO;
{ $EXTERNALSYM pTW_IMAGEINFO }
```

```
TTWImageInfo = TW_IMAGEINFO;
PTWImageInfo = pTW_IMAGEINFO;
```

```
{ DAT_IMAGELAYOUT. Provides image layout information in current units. }
TW_IMAGELAYOUT = packed record
  Frame: TW_FRAME; { Frame coords within larger document }
  DocumentNumber: TW_UINT32;
  PageNumber: TW_UINT32; { Reset when you go to next document }
  FrameNumber: TW_UINT32; { Reset when you go to next page }
end;
{ $EXTERNALSYM TW_IMAGELAYOUT }
pTW_IMAGELAYOUT = ^TW_IMAGELAYOUT;
{ $EXTERNALSYM pTW_IMAGELAYOUT }
```

```
TTWImageLayout = TW_IMAGELAYOUT;
PTWImageLayout = pTW_IMAGELAYOUT;
```

```
{ DAT_IMAGEMEMXFER. Used to pass image data (e.g. in strips) from DS to
application. }
TW_IMAGEMEMXFER = packed record
  Compression: TW_UINT16; { How the data is compressed }
  BytesPerRow: TW_UINT32; { Number of bytes in a row of data }
  Columns: TW_UINT32; { How many columns }
  Rows: TW_UINT32; { How many rows }
```

```

XOffset: TW_UINT32;    { How far from the side of the image }
YOffset: TW_UINT32;    { How far from the top of the image }
BytesWritten: TW_UINT32; { How many bytes written in Memory }
Memory: TW_MEMORY;     { Mem struct used to pass actual image data }
end;
{$EXTERNALSYM TW_IMAGEMEMXFER}
pTW_IMAGEMEMXFER = ^TW_IMAGEMEMXFER;
{$EXTERNALSYM pTW_IMAGEMEMXFER}

TTWImageMemXFER = TW_IMAGEMEMXFER;
PTWImageMemXFER = pTW_IMAGEMEMXFER;

{ Changed in 1.1: QuantTable, HuffmanDC, HuffmanAC TW_MEMREF -> TW_MEMORY }
{ DAT_JPEGCOMPRESSION. Based on JPEG Draft International Std, ver 10918-1. }
TW_JPEGCOMPRESSION = packed record
    ColorSpace: TW_UINT16; { One of the TWPT_xxxx values }
    SubSampling: TW_UINT32; { Two word "array" for subsampling values }
    NumComponents: TW_UINT16; { Number of color components in image }
    RestartFrequency: TW_UINT16; { Frequency of restart marker codes in MDU's }
    QuantMap: array[0..3] of TW_UINT16; { Mapping of components to QuantTables }
    QuantTable: array[0..3] of TW_MEMORY; { Quantization tables }
    HuffmanMap: array[0..3] of TW_UINT16; { Mapping of components to Huffman tables }
    HuffmanDC: array[0..1] of TW_MEMORY; { DC Huffman tables }
    HuffmanAC: array[0..1] of TW_MEMORY; { AC Huffman tables }
end;
{$EXTERNALSYM TW_JPEGCOMPRESSION}
pTW_JPEGCOMPRESSION = ^TW_JPEGCOMPRESSION;
{$EXTERNALSYM pTW_JPEGCOMPRESSION}

TTWJPEGCompression = TW_JPEGCOMPRESSION;
PTWJPEGCompression = pTW_JPEGCOMPRESSION;

{ DAT_PALETTE8. Color palette when TWPT_PALETTE pixels xfer'd in mem buf. }
TW_PALETTE8 = packed record
    NumColors: TW_UINT16; { Number of colors in the color table. }
    PaletteType: TW_UINT16; { TWPA_xxxx, specifies type of palette. }
    Colors: array[0..255] of TW_ELEMENT8; { Array of palette values starts here. }
end;
{$EXTERNALSYM TW_PALETTE8}
pTW_PALETTE8 = ^TW_PALETTE8;
{$EXTERNALSYM pTW_PALETTE8}

TTWPalette8 = TW_PALETTE8;
PTWPalette8 = pTW_PALETTE8;

{ DAT_PENDINGXFERS. Used with MSG_ENDXFER to indicate additional data. }
TW_PENDINGXFERS = packed record
    Count: TW_UINT16;

```

```

case boolean of
  False: (EOJ: TW_UINT32);
  True: (Reserved: TW_UINT32);
end;
{$EXTERNALSYM TW_PENDINGXFERS}
pTW_PENDINGXFERS = ^TW_PENDINGXFERS;
{$EXTERNALSYM pTW_PENDINGXFERS}

```

```

TTWPendingXFERS = TW_PENDINGXFERS;
PTWPendingXFERS = pTW_PENDINGXFERS;

```

```

{ DAT_RGBRESPONSE }
TW_RGBRESPONSE = packed record
  Response: array[0..0] of TW_ELEMENT8;
end;
{$EXTERNALSYM TW_RGBRESPONSE}
pTW_RGBRESPONSE = ^TW_RGBRESPONSE;
{$EXTERNALSYM pTW_RGBRESPONSE}

```

```

TTWRGBResponse = TW_RGBRESPONSE;
PTWRGBResponse = pTW_RGBRESPONSE;

```

```

{ DAT_SETUPFILEXFER. Sets up DS to application data transfer via a file. }
TW_SETUPFILEXFER = packed record
  FileName: TW_STR255;
  Format: TW_UINT16; { Any TWFF_constant }
  VRefNum: TW_INT16; { Used for Mac only }
end;
{$EXTERNALSYM TW_SETUPFILEXFER}
pTW_SETUPFILEXFER = ^TW_SETUPFILEXFER;
{$EXTERNALSYM pTW_SETUPFILEXFER}

```

```

TTWSetupFileXFER = TW_SETUPFILEXFER;
PTWSetupFileXFER = pTW_SETUPFILEXFER;

```

```

{ DAT_SETUPFILEXFER2. Sets up DS to application data transfer via a file. }
{ Added 1.9 }
TW_SETUPFILEXFER2 = packed record
  FileName: TW_MEMREF; { Pointer to file name text }
  FileNameType: TW_UINT16; { TWTY_STR1024 or TWTY_UNI512 }
  Format: TW_UINT16; { Any TWFF_constant }
  VRefNum: TW_INT16; { Used for Mac only }
  parID: TW_UINT32; { Used for Mac only }
end;
{$EXTERNALSYM pTW_SETUPFILEXFER2}
pTW_SETUPFILEXFER2 = ^TW_SETUPFILEXFER2;

```

```
{ $EXTERNALSYM pTW_SETUPFILEXFER2 }
```

```
TTWSetupFileXFER2 = TW_SETUPFILEXFER2;
PTWSetupFileXFER2 = pTW_SETUPFILEXFER2;
```

```
{ DAT_SETUPMEMXFER. Sets up DS to application data transfer via a memory buffer. }
```

```
  TW_SETUPMEMXFER = packed record
```

```
    MinBufSize: TW_UINT32;
```

```
    MaxBufSize: TW_UINT32;
```

```
    Preferred: TW_UINT32;
```

```
  end;
```

```
{ $EXTERNALSYM TW_SETUPMEMXFER }
```

```
pTW_SETUPMEMXFER = ^TW_SETUPMEMXFER;
```

```
{ $EXTERNALSYM pTW_SETUPMEMXFER }
```

```
TTWSetupMemXFER = TW_SETUPMEMXFER;
```

```
PTWSetupMemXFER = pTW_SETUPMEMXFER;
```

```
{ DAT_STATUS. Application gets detailed status info from a data source with this. }
```

```
  TW_STATUS = packed record
```

```
    ConditionCode: TW_UINT16; { Any TWCC_constant }
```

```
    Reserved: TW_UINT16; { Future expansion space }
```

```
  end;
```

```
{ $EXTERNALSYM TW_STATUS }
```

```
pTW_STATUS = ^TW_STATUS;
```

```
{ $EXTERNALSYM pTW_STATUS }
```

```
TTWStatus = TW_STATUS;
```

```
PTWStatus = pTW_STATUS;
```

```
{ DAT_USERINTERFACE. Coordinates UI between application and data source. }
```

```
  TW_USERINTERFACE = packed record
```

```
    ShowUI: TW_BOOL; { TRUE if DS should bring up its UI }
```

```
    ModalUI: TW_BOOL; { For Mac only - true if the DS's UI is modal }
```

```
    hParent: TW_HANDLE; { For windows only - Application window handle }
```

```
  end;
```

```
{ $EXTERNALSYM TW_USERINTERFACE }
```

```
pTW_USERINTERFACE = ^TW_USERINTERFACE;
```

```
{ $EXTERNALSYM pTW_USERINTERFACE }
```

```
TTWUserInterface = TW_USERINTERFACE;
```

```
PTWUserInterface = pTW_USERINTERFACE;
```

```
{ SDH - 03/21/95 - TWUNK }
```

```
{ DAT_TWUNKIDENTITY. Provides DS identity and 'other' information necessary }
```

```
{ across thunk link. }
```

```
  TW_TWUNKIDENTITY = packed record
```

```

    identity: TW_IDENTITY; { Identity of data source. }
    dsPath: TW_STR255;    { Full path and file name of data source. }
end;
{$EXTERNALSYM TW_TWUNKIDENTITY}
pTW_TWUNKIDENTITY = ^TW_TWUNKIDENTITY;
{$EXTERNALSYM pTW_TWUNKIDENTITY}

TTWTwunkIdentity = TW_TWUNKIDENTITY;
PTWTwunkIdentity = pTW_TWUNKIDENTITY;

{ SDH - 03/21/95 - TWUNK }
{ Provides DS_Entry parameters over thunk link. }
TW_TWUNKDSEENTRYPARAMS = packed record
    destFlag: TW_INT8;    { TRUE if dest is not NULL }
    alignment: TW_INT8; // UT: Packed to two byte alignment
    dest: TW_IDENTITY;    { Identity of data source (if used) }
    dataGroup: TW_INT32;  { DSM_Entry dataGroup parameter }
    dataArgType: TW_INT16; { DSM_Entry dataArgType parameter }
    message: TW_INT16;    { DSM_Entry message parameter }
    pDataSize: TW_INT32;  { Size of pData (0 if NULL) }
    //pData: TW_MEMREF;   { Based on implementation specifics, a }
                        { pData parameter makes no sense in this }
                        { structure, but data (if provided) will be }
                        { appended in the data block. }
end;
{$EXTERNALSYM TW_TWUNKDSEENTRYPARAMS}
pTW_TWUNKDSEENTRYPARAMS = ^TW_TWUNKDSEENTRYPARAMS;
{$EXTERNALSYM pTW_TWUNKDSEENTRYPARAMS}

TTWTwunkDSEEntryParams = TW_TWUNKDSEENTRYPARAMS;
PTWTwunkDSEEntryParams = pTW_TWUNKDSEENTRYPARAMS;

{ SDH - 03/21/95 - TWUNK }
{ Provides DS_Entry results over thunk link. }
TW_TWUNKDSEENTRYRETURN = packed record
    returnCode: TW_UINT16; { Thunker DsEntry return code. }
    conditionCode: TW_UINT16; { Thunker DsEntry condition code. }
    pDataSize: TW_INT32;    { Size of pData (0 if NULL) }
    //pData: TW_MEMREF;     { Based on implementation specifics, a }
                        { pData parameter makes no sense in this }
                        { structure, but data (if provided) will be }
                        { appended in the data block. }
end;
{$EXTERNALSYM TW_TWUNKDSEENTRYRETURN}
pTW_TWUNKDSEENTRYRETURN = ^TW_TWUNKDSEENTRYRETURN;
{$EXTERNALSYM pTW_TWUNKDSEENTRYRETURN}

TTWTwunkDSEEntryReturn = TW_TWUNKDSEENTRYRETURN;

```

```
PTWTwunkDSEntryReturn = pTW_TWUNKDSEntryReturn;
```

```
{ WJD - 950818 }
{ Added for 1.6 Specification }
{ TWAIN 1.6 CAP_SUPPORTEDCAPSEXT structure }
TW_CAPEXT = packed record
    Cap: TW_UINT16; { Which CAP/ICAP info is relevant to }
    Properties: TW_UINT16; { Messages this CAP/ICAP supports }
end;
{$EXTERNALSYM TW_CAPEXT}
pTW_CAPEXT = ^TW_CAPEXT;
{$EXTERNALSYM pTW_CAPEXT}
```

```
TTWCapExt = TW_CAPEXT;
PTWCapExt = pTW_CAPEXT;
```

```
{ ----- }
```

Version 1.7: Added Following data structure for Document Imaging
July 1997 Enhancement.

KHL TW_CUSTOMDSDATA -- For Saving and Restoring Source's
state.

TW_INFO -- Each attribute for extended image
information.

TW_EXTIMAGEINFO -- Extended image information structure.

```
----- }
```

```
TW_CUSTOMDSDATA = packed record
    InfoLength: TW_UINT32; { Length of Information in bytes. }
    hData: TW_HANDLE; { Place holder for data, DS Allocates }
end;
{$EXTERNALSYM TW_CUSTOMDSDATA}
pTW_CUSTOMDSDATA = ^TW_CUSTOMDSDATA;
{$EXTERNALSYM pTW_CUSTOMDSDATA}
```

```
TTWCustomDSData = TW_CUSTOMDSDATA;
PTWCustomDSData = pTW_CUSTOMDSDATA;
```

```
TW_INFO = packed record
    InfoID: TW_UINT16;
    ItemType: TW_UINT16;
    NumItems: TW_UINT16;
    CondCode: TW_UINT16;
    Item: TW_UINT32;
end;
{$EXTERNALSYM TW_INFO}
pTW_INFO = ^TW_INFO;
```

```
{ $EXTERNALSYM pTW_INFO }
```

```
TTWInfo = TW_INFO;  
PTWInfo = pTW_INFO;
```

```
TW_EXTIMAGEINFO = packed record  
    NumInfos: TW_UINT32;  
    Info: array[0..0] of TW_INFO;  
end;  
{ $EXTERNALSYM TW_EXTIMAGEINFO }  
pTW_EXTIMAGEINFO = ^TW_EXTIMAGEINFO;  
{ $EXTERNALSYM pTW_EXTIMAGEINFO }
```

```
TTWExtImageInfo = TW_EXTIMAGEINFO;  
PTWExtImageInfo = pTW_EXTIMAGEINFO;
```

```
{ Added 1.8 }
```

```
{ DAT_AUDIOINFO, information about audio data }
```

```
TW_AUDIOINFO = packed record  
    Name: TW_STR255;    { name of audio data }  
    Reserved: TW_UINT32; { reserved space }  
end;  
{ $EXTERNALSYM TW_AUDIOINFO }  
pTW_AUDIOINFO = ^TW_AUDIOINFO;  
{ $EXTERNALSYM pTW_AUDIOINFO }
```

```
TTWAudiolInfo = TW_AUDIOINFO;  
PTWAudiolInfo = pTW_AUDIOINFO;
```

```
{ DAT_DEVICEEVENT, information about events }
```

```
TW_DEVICEEVENT = packed record  
    Event: TW_UINT32;    { One of the TWDE_xxxx values. }  
    DeviceName: TW_STR255; { The name of the device that generated the event }  
    BatteryMinutes: TW_UINT32; { Battery Minutes Remaining }  
    BatteryPercentage: TW_INT16; { Battery Percentage Remaining }  
    PowerSupply: TW_INT32;    { Power Supply }  
    XResolution: TW_FIX32;    { Resolution }  
    YResolution: TW_FIX32;    { Resolution }  
    FlashUsed2: TW_UINT32;    { Flash Used2 }  
    AutomaticCapture: TW_UINT32; { Automatic Capture }  
    TimeBeforeFirstCapture: TW_UINT32; { Automatic Capture }  
    TimeBetweenCaptures: TW_UINT32; { Automatic Capture }  
end;  
{ $EXTERNALSYM TW_DEVICEEVENT }  
pTW_DEVICEEVENT = ^TW_DEVICEEVENT;  
{ $EXTERNALSYM pTW_DEVICEEVENT }
```

```
TTWDeviceEvent = TW_DEVICEEVENT;
PTWDeviceEvent = pTW_DEVICEEVENT;
```

```
{ DAT_FILESYSTEM, information about TWAIN file system }
TW_FILESYSTEM = packed record
  { DG_CONTROL / DAT_FILESYSTEM / MSG_xxxx fields }
  InputName: TW_STR255;    { The name of the input or source file }
  OutputName: TW_STR255;   { The result of an operation or the name of a destination
file }
  Context: TW_MEMREF;      { Source specific data used to remember state
information }
  { DG_CONTROL / DAT_FILESYSTEM / MSG_DELETE field }
  Recursive: Integer;{int} { recursively delete all sub-directories }
  { DG_CONTROL / DAT_FILESYSTEM / MSG_GETINFO fields }
  FileType: TW_INT32;      { One of the TWFT_xxxx values }
  Size: TW_UINT32;         { Size of current FileType }
  CreateTimeDate: TW_STR32; { creation date of the file }
  ModifiedTimeDate: TW_STR32; { last date the file was modified }
  FreeSpace: TW_UINT32;    { bytes of free space on the current device }
  NewImageSize: TW_INT32;  { estimate of the amount of space a new image would
take up }
  NumberOfFiles: TW_UINT32; { number of files, depends on FileType }
  NumberOfSnippets: TW_UINT32; { number of audio snippets }
  DeviceGroupMask: TW_UINT32; { used to group cameras (ex: front/rear bitonal,
front/rear grayscale...) }
  Reserved: array[0..507] of Char; { }
end;
{$EXTERNALSYM TW_FILESYSTEM}
pTW_FILESYSTEM = ^TW_FILESYSTEM;
{$EXTERNALSYM pTW_FILESYSTEM}
```

```
TTWFileSystem = TW_FILESYSTEM;
PTWFileSystem = pTW_FILESYSTEM;
```

```
{ DAT_PASSTHRU, device dependant data to pass through Data Source }
TW_PASSTHRU = packed record
  pCommand: TW_MEMREF;     { Pointer to Command buffer }
  CommandBytes: TW_UINT32; { Number of bytes in Command buffer }
  Direction: TW_INT32;     { One of the TWDR_xxxx values. Defines the direction of
data flow }
  pData: TW_MEMREF;        { Pointer to Data buffer }
  DataBytes: TW_UINT32;    { Number of bytes in Data buffer }
  DataBytesXfered: TW_UINT32; { Number of bytes successfully transferred }
end;
{$EXTERNALSYM TW_PASSTHRU}
pTW_PASSTHRU = ^TW_PASSTHRU;
{$EXTERNALSYM pTW_PASSTHRU}
```

```
TTWPassThru = TW_PASSTHRU;
PTWPassThru = pTW_PASSTHRU;
```

```
{ DAT_SETUPAUDIOFILEXFER, information required to setup an audio file transfer }
TW_SETUPAUDIOFILEXFER = packed record
  FileName: TW_STR255; { full path target file }
  Format: TW_UINT16; { one of TWAF_xxxx }
  VRefNum: TW_INT16;
end;
{$EXTERNALSYM TW_SETUPAUDIOFILEXFER}
pTW_SETUPAUDIOFILEXFER = ^TW_SETUPAUDIOFILEXFER;
{$EXTERNALSYM pTW_SETUPAUDIOFILEXFER}
```

```
TTWSetupAudioFileXFER = TW_SETUPAUDIOFILEXFER;
PTWSetupAudioFileXFER = pTW_SETUPAUDIOFILEXFER;
```

```
{*****
* Generic Constants *
***** }
```

```
const
```

```
TWON_ARRAY = 3; { indicates TW_ARRAY container }
{$EXTERNALSYM TWON_ARRAY}
TWON_ENUMERATION = 4; { indicates TW_ENUMERATION container }
{$EXTERNALSYM TWON_ENUMERATION}
TWON_ONEVALUE = 5; { indicates TW_ONEVALUE container }
{$EXTERNALSYM TWON_ONEVALUE}
TWON_RANGE = 6; { indicates TW_RANGE container }
{$EXTERNALSYM TWON_RANGE}
```

```
TWON_ICONID = 962; { res Id of icon used in USERSELECT lbox }
{$EXTERNALSYM TWON_ICONID}
TWON_DSMID = 461; { res Id of the DSM version num resource }
{$EXTERNALSYM TWON_DSMID}
TWON_DSMCODEID = 63; { res Id of the Mac SM Code resource }
{$EXTERNALSYM TWON_DSMCODEID}
```

```
TWON_DONTCARE8 = $ff;
{$EXTERNALSYM TWON_DONTCARE8}
TWON_DONTCARE16 = $ffff;
{$EXTERNALSYM TWON_DONTCARE16}
TWON_DONTCARE32 = DWORD($ffffffff);
{$EXTERNALSYM TWON_DONTCARE32}
```

```
{ Flags used in TW_MEMORY structure. }
```

```
TWMF_APPOWNS = $1;
{$EXTERNALSYM TWMF_APPOWNS}
TWMF_DSMOWNS = $2;
{$EXTERNALSYM TWMF_DSMOWNS}
```

```

TWMF_DSOWNS  = $4;
{$EXTERNALSYM TWMF_DSOWNS}
TWMF_POINTER  = $8;
{$EXTERNALSYM TWMF_POINTER}
TWMF_HANDLE   = $10;
{$EXTERNALSYM TWMF_HANDLE}

```

```

{ Palette types for TW_PALETTE8 }
TWPA_RGB      = 0;
{$EXTERNALSYM TWPA_RGB}
TWPA_GRAY     = 1;
{$EXTERNALSYM TWPA_GRAY}
TWPA_CMY      = 2;
{$EXTERNALSYM TWPA_CMY}

```

```

{ There are four containers used for capabilities negotiation:
* TWON_ONEVALUE, TWON_RANGE, TWON_ENUMERATION, TWON_ARRAY
* In each container structure ItemType can be TWTY_INT8, TWTY_INT16, etc.
* The kind of data stored in the container can be determined by doing
* DCItemSize[ItemType] where the following is defined in TWAIN glue code:
*   DCItemSize[]= sizeof(TW_INT8),
*                 sizeof(TW_INT16),
*                 etc.
*                 sizeof(TW_UINT32) : ;
*
}

```

```

TWTY_INT8     = $0000; { Means Item is a TW_INT8 }
{$EXTERNALSYM TWTY_INT8}
TWTY_INT16    = $0001; { Means Item is a TW_INT16 }
{$EXTERNALSYM TWTY_INT16}
TWTY_INT32    = $0002; { Means Item is a TW_INT32 }
{$EXTERNALSYM TWTY_INT32}

```

```

TWTY_UINT8    = $0003; { Means Item is a TW_UINT8 }
{$EXTERNALSYM TWTY_UINT8}
TWTY_UINT16   = $0004; { Means Item is a TW_UINT16 }
{$EXTERNALSYM TWTY_UINT16}
TWTY_UINT32   = $0005; { Means Item is a TW_UINT32 }
{$EXTERNALSYM TWTY_UINT32}

```

```

TWTY_BOOL     = $0006; { Means Item is a TW_BOOL }
{$EXTERNALSYM TWTY_BOOL}

```

```

TWTY_FIX32    = $0007; { Means Item is a TW_FIX32 }
{$EXTERNALSYM TWTY_FIX32}

```

```

TWTY_FRAME    = $0008; { Means Item is a TW_FRAME }

```

```
{ $EXTERNALSYM TWTY_FRAME }
```

```
TWTY_STR32 = $0009; { Means Item is a TW_STR32 }
```

```
{ $EXTERNALSYM TWTY_STR32 }
```

```
TWTY_STR64 = $000a; { Means Item is a TW_STR64 }
```

```
{ $EXTERNALSYM TWTY_STR64 }
```

```
TWTY_STR128 = $000b; { Means Item is a TW_STR128 }
```

```
{ $EXTERNALSYM TWTY_STR128 }
```

```
TWTY_STR255 = $000c; { Means Item is a TW_STR255 }
```

```
{ $EXTERNALSYM TWTY_STR255 }
```

```
TWTY_STR1024 = $000d; { Means Item is a TW_STR1024...added 1.9 }
```

```
{ $EXTERNALSYM TWTY_STR1024 }
```

```
TWTY_UNI512 = $000e; { Means Item is a TW_UNI512...added 1.9 }
```

```
{ $EXTERNALSYM TWTY_UNI512 }
```

```
{ ***** }
```

```
* Capability Constants
```

```
*
```

```
{ ***** }
```

```
{ ICAP_BITORDER values (BO_ means Bit Order) }
```

```
TWBO_LSBFIRST = 0;
```

```
{ $EXTERNALSYM TWBO_LSBFIRST }
```

```
TWBO_MSBFIRST = 1;
```

```
{ $EXTERNALSYM TWBO_MSBFIRST }
```

```
{ ICAP_COMPRESSION values (CP_ means ComPression ) }
```

```
TWCP_NONE = 0;
```

```
{ $EXTERNALSYM TWCP_NONE }
```

```
TWCP_PACKBITS = 1;
```

```
{ $EXTERNALSYM TWCP_PACKBITS }
```

```
TWCP_GROUP31D = 2; { Follows CCITT spec (no End Of Line) }
```

```
{ $EXTERNALSYM TWCP_GROUP31D }
```

```
TWCP_GROUP31DEOL = 3; { Follows CCITT spec (has End Of Line) }
```

```
{ $EXTERNALSYM TWCP_GROUP31DEOL }
```

```
TWCP_GROUP32D = 4; { Follows CCITT spec (use cap for K Factor) }
```

```
{ $EXTERNALSYM TWCP_GROUP32D }
```

```
TWCP_GROUP4 = 5; { Follows CCITT spec }
```

```
{ $EXTERNALSYM TWCP_GROUP4 }
```

```
TWCP_JPEG = 6; { Use capability for more info }
```

```
{ $EXTERNALSYM TWCP_JPEG }
```

```
TWCP_LZW = 7; { Must license from Unisys and IBM to use }
```

```
{ $EXTERNALSYM TWCP_LZW }
```

```
TWCP_JBIG = 8; { For Bitonal images -- Added 1.7 KHL }
```

```
{ $EXTERNALSYM TWCP_JBIG }
```

```
{ Added 1.8 }
```

```
TWCP_PNG = 9;
```

```
{ $EXTERNALSYM TWCP_PNG }
```

```

TWCP_RLE4   = 10;
{$EXTERNALSYM TWCP_RLE4}
TWCP_RLE8   = 11;
{$EXTERNALSYM TWCP_RLE8}
TWCP_BITFIELDS = 12;
{$EXTERNALSYM TWCP_BITFIELDS}

```

```
{ ICAP_IMAGEFILEFORMAT values (FF_ means File Format) }
```

```

TWFF_TIFF   = 0; { Tagged Image File Format }
{$EXTERNALSYM TWFF_TIFF}
TWFF_PICT   = 1; { Macintosh PICT }
{$EXTERNALSYM TWFF_PICT}
TWFF_BMP    = 2; { Windows Bitmap }
{$EXTERNALSYM TWFF_BMP}
TWFF_XBM    = 3; { X-Windows Bitmap }
{$EXTERNALSYM TWFF_XBM}
TWFF_JFIF   = 4; { JPEG File Interchange Format }
{$EXTERNALSYM TWFF_JFIF}
TWFF_FPX    = 5; { Flash Pix }
{$EXTERNALSYM TWFF_FPX}
TWFF_TIFFMULTI= 6; { Multi-page tiff file }
{$EXTERNALSYM TWFF_TIFFMULTI}
TWFF_PNG    = 7;
{$EXTERNALSYM TWFF_PNG}
TWFF_SPIFF  = 8;
{$EXTERNALSYM TWFF_SPIFF}
TWFF_EXIF   = 9;
{$EXTERNALSYM TWFF_EXIF}

```

```
{ ICAP_FILTER values (FT_ means Filter Type) }
```

```

TWFT_RED    = 0;
{$EXTERNALSYM TWFT_RED}
TWFT_GREEN  = 1;
{$EXTERNALSYM TWFT_GREEN}
TWFT_BLUE   = 2;
{$EXTERNALSYM TWFT_BLUE}
TWFT_NONE   = 3;
{$EXTERNALSYM TWFT_NONE}
TWFT_WHITE  = 4;
{$EXTERNALSYM TWFT_WHITE}
TWFT_CYAN   = 5;
{$EXTERNALSYM TWFT_CYAN}
TWFT_MAGENTA = 6;
{$EXTERNALSYM TWFT_MAGENTA}
TWFT_YELLOW = 7;
{$EXTERNALSYM TWFT_YELLOW}

```

```
TWFT_BLACK = 8;
{$EXTERNALSYM TWFT_BLACK}
```

```
{ ICAP_LIGHTPATH values (LP_ means Light Path) }
TWLP_REFLECTIVE = 0;
{$EXTERNALSYM TWLP_REFLECTIVE}
TWLP_TRANSMISSIVE = 1;
{$EXTERNALSYM TWLP_TRANSMISSIVE}
```

```
{ ICAP_LIGHTSOURCE values (LS_ means Light Source) }
TWLS_RED = 0;
{$EXTERNALSYM TWLS_RED}
TWLS_GREEN = 1;
{$EXTERNALSYM TWLS_GREEN}
TWLS_BLUE = 2;
{$EXTERNALSYM TWLS_BLUE}
TWLS_NONE = 3;
{$EXTERNALSYM TWLS_NONE}
TWLS_WHITE = 4;
{$EXTERNALSYM TWLS_WHITE}
TWLS_UV = 5;
{$EXTERNALSYM TWLS_UV}
TWLS_IR = 6;
{$EXTERNALSYM TWLS_IR}
```

```
{ ICAP_ORIENTATION values (OR_ means ORientation) }
TWOR_ROT0 = 0;
{$EXTERNALSYM TWOR_ROT0}
TWOR_ROT90 = 1;
{$EXTERNALSYM TWOR_ROT90}
TWOR_ROT180 = 2;
{$EXTERNALSYM TWOR_ROT180}
TWOR_ROT270 = 3;
{$EXTERNALSYM TWOR_ROT270}
TWOR_PORTRAIT = TWOR_ROT0;
{$EXTERNALSYM TWOR_PORTRAIT}
TWOR_LANDSCAPE = TWOR_ROT270;
{$EXTERNALSYM TWOR_LANDSCAPE}
```

```
{ ICAP_PLANARCHUNKY values (PC_ means Planar/Chunky ) }
TWPC_CHUNKY = 0;
{$EXTERNALSYM TWPC_CHUNKY}
TWPC_PLANAR = 1;
{$EXTERNALSYM TWPC_PLANAR}
```

```
{ ICAP_PIXELFLAVOR values (PF_ means Pixel Flavor) }
TWPF_CHOCOLATE = 0; { zero pixel represents darkest shade }
{$EXTERNALSYM TWPF_CHOCOLATE}
```

```

TWPF_VANILLA = 1; { zero pixel represents lightest shade }
{$EXTERNALSYM TWPF_VANILLA}

{ ICAP_PIXELTYPE values (PT_ means Pixel Type) }
TWPT_BW = 0; { Black and White }
{$EXTERNALSYM TWPT_BW}
TWPT_GRAY = 1;
{$EXTERNALSYM TWPT_GRAY}
TWPT_RGB = 2;
{$EXTERNALSYM TWPT_RGB}
TWPT_PALETTE = 3;
{$EXTERNALSYM TWPT_PALETTE}
TWPT_CMY = 4;
{$EXTERNALSYM TWPT_CMY}
TWPT_CMYK = 5;
{$EXTERNALSYM TWPT_CMYK}
TWPT_YUV = 6;
{$EXTERNALSYM TWPT_YUV}
TWPT_YUVK = 7;
{$EXTERNALSYM TWPT_YUVK}
TWPT_CIEXYZ = 8;
{$EXTERNALSYM TWPT_CIEXYZ}

{ ICAP_SUPPORTEDSIZES values (SS_ means Supported Sizes) }
TWSS_NONE = 0;
{$EXTERNALSYM TWSS_NONE}
TWSS_A4LETTER = 1;
{$EXTERNALSYM TWSS_A4LETTER}
TWSS_B5LETTER = 2;
{$EXTERNALSYM TWSS_B5LETTER}
TWSS_USLETTER = 3;
{$EXTERNALSYM TWSS_USLETTER}
TWSS_USLEGAL = 4;
{$EXTERNALSYM TWSS_USLEGAL}
{ Added 1.5 }
TWSS_A5 = 5;
{$EXTERNALSYM TWSS_A5}
TWSS_B4 = 6;
{$EXTERNALSYM TWSS_B4}
TWSS_B6 = 7;
{$EXTERNALSYM TWSS_B6}
// TWSS_B = 8;

{ Added 1.7 }
TWSS_USLEDGER = 9;
{$EXTERNALSYM TWSS_USLEDGER}
TWSS_USEXECUTIVE = 10;
{$EXTERNALSYM TWSS_USEXECUTIVE}

```

```

TWSS_A3      = 11;
{$EXTERNALSYM TWSS_A3}
TWSS_B3      = 12;
{$EXTERNALSYM TWSS_B3}
TWSS_A6      = 13;
{$EXTERNALSYM TWSS_A6}
TWSS_C4      = 14;
{$EXTERNALSYM TWSS_C4}
TWSS_C5      = 15;
{$EXTERNALSYM TWSS_C5}
TWSS_C6      = 16;
{$EXTERNALSYM TWSS_C6}

```

```

{ Added 1.8 }
TWSS_4A0     = 17;
{$EXTERNALSYM TWSS_4A0}
TWSS_2A0     = 18;
{$EXTERNALSYM TWSS_2A0}
TWSS_A0      = 19;
{$EXTERNALSYM TWSS_A0}
TWSS_A1      = 20;
{$EXTERNALSYM TWSS_A1}
TWSS_A2      = 21;
{$EXTERNALSYM TWSS_A2}
TWSS_A4      = TWSS_A4LETTER;
{$EXTERNALSYM TWSS_A4}
TWSS_A7      = 22;
{$EXTERNALSYM TWSS_A7}
TWSS_A8      = 23;
{$EXTERNALSYM TWSS_A8}
TWSS_A9      = 24;
{$EXTERNALSYM TWSS_A9}
TWSS_A10     = 25;
{$EXTERNALSYM TWSS_A10}
TWSS_ISOB0   = 26;
{$EXTERNALSYM TWSS_ISOB0}
TWSS_ISOB1   = 27;
{$EXTERNALSYM TWSS_ISOB1}
TWSS_ISOB2   = 28;
{$EXTERNALSYM TWSS_ISOB2}
TWSS_ISOB3   = TWSS_B3;
{$EXTERNALSYM TWSS_ISOB3}
TWSS_ISOB4   = TWSS_B4;
{$EXTERNALSYM TWSS_ISOB4}
TWSS_ISOB5   = 29;
{$EXTERNALSYM TWSS_ISOB5}
TWSS_ISOB6   = TWSS_B6;
{$EXTERNALSYM TWSS_ISOB6}

```

```

TWSS_ISO7      = 30;
{$EXTERNALSYM TWSS_ISO7}
TWSS_ISO8      = 31;
{$EXTERNALSYM TWSS_ISO8}
TWSS_ISO9      = 32;
{$EXTERNALSYM TWSS_ISO9}
TWSS_ISO10     = 33;
{$EXTERNALSYM TWSS_ISO10}
TWSS_JISB0     = 34;
{$EXTERNALSYM TWSS_JISB0}
TWSS_JISB1     = 35;
{$EXTERNALSYM TWSS_JISB1}
TWSS_JISB2     = 36;
{$EXTERNALSYM TWSS_JISB2}
TWSS_JISB3     = 37;
{$EXTERNALSYM TWSS_JISB3}
TWSS_JISB4     = 38;
{$EXTERNALSYM TWSS_JISB4}
TWSS_JISB5     = TWSS_B5LETTER;
{$EXTERNALSYM TWSS_JISB5}
TWSS_JISB6     = 39;
{$EXTERNALSYM TWSS_JISB6}
TWSS_JISB7     = 40;
{$EXTERNALSYM TWSS_JISB7}
TWSS_JISB8     = 41;
{$EXTERNALSYM TWSS_JISB8}
TWSS_JISB9     = 42;
{$EXTERNALSYM TWSS_JISB9}
TWSS_JISB10    = 43;
{$EXTERNALSYM TWSS_JISB10}
TWSS_C0        = 44;
{$EXTERNALSYM TWSS_C0}
TWSS_C1        = 45;
{$EXTERNALSYM TWSS_C1}
TWSS_C2        = 46;
{$EXTERNALSYM TWSS_C2}
TWSS_C3        = 47;
{$EXTERNALSYM TWSS_C3}
TWSS_C7        = 48;
{$EXTERNALSYM TWSS_C7}
TWSS_C8        = 49;
{$EXTERNALSYM TWSS_C8}
TWSS_C9        = 50;
{$EXTERNALSYM TWSS_C9}
TWSS_C10       = 51;
{$EXTERNALSYM TWSS_C10}
TWSS_USSTATEMENT = 52;
{$EXTERNALSYM TWSS_USSTATEMENT}

```

```
TWSS_BUSINESSCARD= 53;
{$EXTERNALSYM TWSS_BUSINESSCARD}
```

```
{ ICAP_XFERMECH values (SX_ means Setup XFer) }
TWSX_NATIVE = 0;
{$EXTERNALSYM TWSX_NATIVE}
TWSX_FILE = 1;
{$EXTERNALSYM TWSX_FILE}
TWSX_MEMORY = 2;
{$EXTERNALSYM TWSX_MEMORY}
TWSX_FILE2 = 3; { added 1.9 }
{$EXTERNALSYM TWSX_FILE2}
```

```
{ ICAP_UNITS values (UN_ means UNits) }
TWUN_INCHES = 0;
{$EXTERNALSYM TWUN_INCHES}
TWUN_CENTIMETERS = 1;
{$EXTERNALSYM TWUN_CENTIMETERS}
TWUN_PICAS = 2;
{$EXTERNALSYM TWUN_PICAS}
TWUN_POINTS = 3;
{$EXTERNALSYM TWUN_POINTS}
TWUN_TWIPS = 4;
{$EXTERNALSYM TWUN_TWIPS}
TWUN_PIXELS = 5;
{$EXTERNALSYM TWUN_PIXELS}
```

```
{ Added 1.5 }
{ ICAP_BITDEPTHREDUCTION values (BR_ means Bitdepth Reduction) }
TWBR_THRESHOLD = 0;
{$EXTERNALSYM TWBR_THRESHOLD}
TWBR_HALFTONE = 1;
{$EXTERNALSYM TWBR_HALFTONE}
TWBR_CUSTHALFTONE = 2;
{$EXTERNALSYM TWBR_CUSTHALFTONE}
TWBR_DIFFUSION = 3;
{$EXTERNALSYM TWBR_DIFFUSION}
```

```
{ Added 1.7 }
{ ICAP_DUPLEX values }
TWDX_NONE = 0;
{$EXTERNALSYM TWDX_NONE}
TWDX_1PASSDUPLEX = 1;
{$EXTERNALSYM TWDX_1PASSDUPLEX}
TWDX_2PASSDUPLEX = 2;
{$EXTERNALSYM TWDX_2PASSDUPLEX}
```

```
{ Added 1.7 }
```

```

{ TWEI_BARCODETYPE values }
TWBT_3OF9          = 0;
{$EXTERNALSYM TWBT_3OF9}
TWBT_2OF5INTERLEAVED = 1;
{$EXTERNALSYM TWBT_2OF5INTERLEAVED}
TWBT_2OF5NONINTERLEAVED= 2;
{$EXTERNALSYM TWBT_2OF5NONINTERLEAVED}
TWBT_CODE93        = 3;
{$EXTERNALSYM TWBT_CODE93}
TWBT_CODE128        = 4;
{$EXTERNALSYM TWBT_CODE128}
TWBT_UCC128         = 5;
{$EXTERNALSYM TWBT_UCC128}
TWBT_CODABAR        = 6;
{$EXTERNALSYM TWBT_CODABAR}
TWBT_UPCA           = 7;
{$EXTERNALSYM TWBT_UPCA}
TWBT_UPCE           = 8;
{$EXTERNALSYM TWBT_UPCE}
TWBT_EAN8           = 9;
{$EXTERNALSYM TWBT_EAN8}
TWBT_EAN13          = 10;
{$EXTERNALSYM TWBT_EAN13}
TWBT_POSTNET        = 11;
{$EXTERNALSYM TWBT_POSTNET}
TWBT_PDF417         = 12;
{$EXTERNALSYM TWBT_PDF417}

{ Added 1.8 }
TWBT_2OF5INDUSTRIAL = 13;
{$EXTERNALSYM TWBT_2OF5INDUSTRIAL}
TWBT_2OF5MATRIX      = 14;
{$EXTERNALSYM TWBT_2OF5MATRIX}
TWBT_2OF5DATALOGIC   = 15;
{$EXTERNALSYM TWBT_2OF5DATALOGIC}
TWBT_2OF5IATA        = 16;
{$EXTERNALSYM TWBT_2OF5IATA}
TWBT_3OF9FULLASCII   = 17;
{$EXTERNALSYM TWBT_3OF9FULLASCII}
TWBT_CODABARWITHSTARTSTOP = 18;
{$EXTERNALSYM TWBT_CODABARWITHSTARTSTOP}
TWBT_MAXICODE        = 19;
{$EXTERNALSYM TWBT_MAXICODE}

{ Added 1.7 }
{ TWEI_DESKEWSTATUS values }
TWDSK_SUCCESS        = 0;
{$EXTERNALSYM TWDSK_SUCCESS}

```

```

TWDSK_REPORTONLY = 1;
{$EXTERNALSYM TWDSK_REPORTONLY}
TWDSK_FAIL      = 2;
{$EXTERNALSYM TWDSK_FAIL}
TWDSK_DISABLED  = 3;
{$EXTERNALSYM TWDSK_DISABLED}

```

```

{ Added 1.7 }
{ TWEI_PATCHCODE values }
TWPCH_PATCH1  = 0;
{$EXTERNALSYM TWPCH_PATCH1}
TWPCH_PATCH2  = 1;
{$EXTERNALSYM TWPCH_PATCH2}
TWPCH_PATCH3  = 2;
{$EXTERNALSYM TWPCH_PATCH3}
TWPCH_PATCH4  = 3;
{$EXTERNALSYM TWPCH_PATCH4}
TWPCH_PATCH6  = 4;
{$EXTERNALSYM TWPCH_PATCH6}
TWPCH_PATCHT  = 5;
{$EXTERNALSYM TWPCH_PATCHT}

```

```

{ Added 1.7 }
{ CAP_JOBCONTROL values }
TWJC_NONE     = 0;
{$EXTERNALSYM TWJC_NONE}
TWJC_JSIC     = 1;
{$EXTERNALSYM TWJC_JSIC}
TWJC_J SIS    = 2;
{$EXTERNALSYM TWJC_J SIS}
TWJC_JSXC     = 3;
{$EXTERNALSYM TWJC_JSXC}
TWJC_JSXS     = 4;
{$EXTERNALSYM TWJC_JSXS}

```

```

{ Added 1.7 }
{ TWEI_BARCODEROTATION values (BCOR_ means barcode rotation) }
TWBCOR_ROT0   = 0;
{$EXTERNALSYM TWBCOR_ROT0}
TWBCOR_ROT90  = 1;
{$EXTERNALSYM TWBCOR_ROT90}
TWBCOR_ROT180 = 2;
{$EXTERNALSYM TWBCOR_ROT180}
TWBCOR_ROT270 = 3;
{$EXTERNALSYM TWBCOR_ROT270}
TWBCOR_ROTX   = 4;
{$EXTERNALSYM TWBCOR_ROTX}

```

```

{ Added 1.8 }
{ ACAP_AUDIOFILEFORMAT values (AF_ means audio format) }
  TWAF_WAV = 0;
  {$EXTERNALSYM TWAF_WAV}
  TWAF_AIFF = 1;
  {$EXTERNALSYM TWAF_AIFF}
  TWAF_AU = 3;
  {$EXTERNALSYM TWAF_AU}
  TWAF_SND = 4;
  {$EXTERNALSYM TWAF_SND}

{ CAP_ALARMS values (AL_ means alarms) }
  TWAL_ALARM = 0;
  {$EXTERNALSYM TWAL_ALARM}
  TWAL_FEEDERERROR = 1;
  {$EXTERNALSYM TWAL_FEEDERERROR}
  TWAL_FEEDERWARNING = 2;
  {$EXTERNALSYM TWAL_FEEDERWARNING}
  TWAL_BARCODE = 3;
  {$EXTERNALSYM TWAL_BARCODE}
  TWAL_DOUBLEFEED = 4;
  {$EXTERNALSYM TWAL_DOUBLEFEED}
  TWAL_JAM = 5;
  {$EXTERNALSYM TWAL_JAM}
  TWAL_PATCHCODE = 6;
  {$EXTERNALSYM TWAL_PATCHCODE}
  TWAL_POWER = 7;
  {$EXTERNALSYM TWAL_POWER}
  TWAL_SKEW = 8;
  {$EXTERNALSYM TWAL_SKEW}

{ CAP_CLEARBUFFERS values (CB_ means clear buffers) }
  TWCB_AUTO = 0;
  {$EXTERNALSYM TWCB_AUTO}
  TWCB_CLEAR = 1;
  {$EXTERNALSYM TWCB_CLEAR}
  TWCB_NOCLEAR = 2;
  {$EXTERNALSYM TWCB_NOCLEAR}

{ CAP_DEVICEEVENT values (DE_ means device event) }
  TWDE_CUSTOMEVENTS = $8000;
  {$EXTERNALSYM TWDE_CUSTOMEVENTS}
  TWDE_CHECKAUTOMATICCAPTURE = 0;
  {$EXTERNALSYM TWDE_CHECKAUTOMATICCAPTURE}
  TWDE_CHECKBATTERY = 1;
  {$EXTERNALSYM TWDE_CHECKBATTERY}
  TWDE_CHECKDEVICEONLINE = 2;
  {$EXTERNALSYM TWDE_CHECKDEVICEONLINE}

```

```

TWDE_CHECKFLASH      = 3;
{$EXTERNALSYM TWDE_CHECKFLASH}
TWDE_CHECKPOWERSUPPLY = 4;
{$EXTERNALSYM TWDE_CHECKPOWERSUPPLY}
TWDE_CHECKRESOLUTION  = 5;
{$EXTERNALSYM TWDE_CHECKRESOLUTION}
TWDE_DEVICEADDED      = 6;
{$EXTERNALSYM TWDE_DEVICEADDED}
TWDE_DEVICEOFFLINE    = 7;
{$EXTERNALSYM TWDE_DEVICEOFFLINE}
TWDE_DEVICEREADY      = 8;
{$EXTERNALSYM TWDE_DEVICEREADY}
TWDE_DEVICEREMOVED    = 9;
{$EXTERNALSYM TWDE_DEVICEREMOVED}
TWDE_IMAGECAPTURED    = 10;
{$EXTERNALSYM TWDE_IMAGECAPTURED}
TWDE_IMAGEDELETED     = 11;
{$EXTERNALSYM TWDE_IMAGEDELETED}
TWDE_PAPERDOUBLEFEED  = 12;
{$EXTERNALSYM TWDE_PAPERDOUBLEFEED}
TWDE_PAPERJAM         = 13;
{$EXTERNALSYM TWDE_PAPERJAM}
TWDE_LAMPFAILURE      = 14;
{$EXTERNALSYM TWDE_LAMPFAILURE}
TWDE_POWERSAVE        = 15;
{$EXTERNALSYM TWDE_POWERSAVE}
TWDE_POWERSAVENOTIFY  = 16;
{$EXTERNALSYM TWDE_POWERSAVENOTIFY}

```

{ CAP_FEEDERALIGNMENT values (FA_ means feeder alignment) }

```

TWFA_NONE = 0;
{$EXTERNALSYM TWFA_NONE}
TWFA_LEFT  = 1;
{$EXTERNALSYM TWFA_LEFT}
TWFA_CENTER = 2;
{$EXTERNALSYM TWFA_CENTER}
TWFA_RIGHT = 3;
{$EXTERNALSYM TWFA_RIGHT}

```

{ CAP_FEEDERORDER values (FO_ means feeder order) }

```

TWFO_FIRSTPAGEFIRST = 0;
{$EXTERNALSYM TWFO_FIRSTPAGEFIRST}
TWFO_LASTPAGEFIRST  = 1;
{$EXTERNALSYM TWFO_LASTPAGEFIRST}

```

{ CAP_FILESYSTEM values (FS_ means file system) }

```

TWFS_FILESYSTEM = 0;
{$EXTERNALSYM TWFS_FILESYSTEM}

```

```

TWFS_RECURSIVEDELETE = 1;
{$EXTERNALSYM TWFS_RECURSIVEDELETE}

{ CAP_POWERSUPPLY values (PS_ means power supply) }
TWPS_EXTERNAL = 0;
{$EXTERNALSYM TWPS_EXTERNAL}
TWPS_BATTERY = 1;
{$EXTERNALSYM TWPS_BATTERY}

{ CAP_PRINTER values (PR_ means printer) }
TWPR_IMPRINTERTOPBEFORE = 0;
{$EXTERNALSYM TWPR_IMPRINTERTOPBEFORE}
TWPR_IMPRINTERTOPAFTER = 1;
{$EXTERNALSYM TWPR_IMPRINTERTOPAFTER}
TWPR_IMPRINTERBOTTOMBEFORE = 2;
{$EXTERNALSYM TWPR_IMPRINTERBOTTOMBEFORE}
TWPR_IMPRINTERBOTTOMAFTER = 3;
{$EXTERNALSYM TWPR_IMPRINTERBOTTOMAFTER}
TWPR_ENDORSERTOPBEFORE = 4;
{$EXTERNALSYM TWPR_ENDORSERTOPBEFORE}
TWPR_ENDORSERTOPAFTER = 5;
{$EXTERNALSYM TWPR_ENDORSERTOPAFTER}
TWPR_ENDORSERBOTTOMBEFORE = 6;
{$EXTERNALSYM TWPR_ENDORSERBOTTOMBEFORE}
TWPR_ENDORSERBOTTOMAFTER = 7;
{$EXTERNALSYM TWPR_ENDORSERBOTTOMAFTER}

{ CAP_PRINTERMODE values (PM_ means printer mode) }
TWPM_SINGLESTRING = 0;
{$EXTERNALSYM TWPM_SINGLESTRING}
TWPM_MULTISTRING = 1;
{$EXTERNALSYM TWPM_MULTISTRING}
TWPM_COMPOUNDSTRING = 2;
{$EXTERNALSYM TWPM_COMPOUNDSTRING}

{ ICAP_BARCODESEARCHMODE values (TWBD_ means search) }
TWBD_HORZ = 0;
{$EXTERNALSYM TWBD_HORZ}
TWBD_VERT = 1;
{$EXTERNALSYM TWBD_VERT}
TWBD_HORZVERT = 2;
{$EXTERNALSYM TWBD_HORZVERT}
TWBD_VERTHORZ = 3;
{$EXTERNALSYM TWBD_VERTHORZ}

{ ICAP_FLASHUSED2 values (FL_ means flash) }
TWFL_NONE = 0;
{$EXTERNALSYM TWFL_NONE}

```

```

TWFL_OFF   = 1;
{$EXTERNALSYM TWFL_OFF}
TWFL_ON    = 2;
{$EXTERNALSYM TWFL_ON}
TWFL_AUTO  = 3;
{$EXTERNALSYM TWFL_AUTO}
TWFL_REDEYE = 4;
{$EXTERNALSYM TWFL_REDEYE}

```

```

{ ICAP_FLIPROTATION values (FR_ means flip rotation) }
TWFR_BOOK  = 0;
{$EXTERNALSYM TWFR_BOOK}
TWFR_FANFOLD = 1;
{$EXTERNALSYM TWFR_FANFOLD}

```

```

{ ICAP_IMAGEFILTER values (IF_ means image filter) }
TWIF_NONE  = 0;
{$EXTERNALSYM TWIF_NONE}
TWIF_AUTO  = 1;
{$EXTERNALSYM TWIF_AUTO}
TWIF_LOWPASS = 2;
{$EXTERNALSYM TWIF_LOWPASS}
TWIF_BANDPASS = 3;
{$EXTERNALSYM TWIF_BANDPASS}
TWIF_HIGHPASS = 4;
{$EXTERNALSYM TWIF_HIGHPASS}
TWIF_TEXT    = TWIF_BANDPASS;
{$EXTERNALSYM TWIF_TEXT}
TWIF_FINELINE = TWIF_HIGHPASS;
{$EXTERNALSYM TWIF_FINELINE}

```

```

{ ICAP_NOISEFILTER values (NF_ means noise filter) }
TWNF_NONE   = 0;
{$EXTERNALSYM TWNF_NONE}
TWNF_AUTO   = 1;
{$EXTERNALSYM TWNF_AUTO}
TWNF_LONEPIXEL = 2;
{$EXTERNALSYM TWNF_LONEPIXEL}
TWNF_MAJORITYRULE = 3;
{$EXTERNALSYM TWNF_MAJORITYRULE}

```

```

{ ICAP_OVERSCAN values (OV_ means overscan) }
TWOV_NONE   = 0;
{$EXTERNALSYM TWOV_NONE}
TWOV_AUTO   = 1;
{$EXTERNALSYM TWOV_AUTO}
TWOV_TOPBOTTOM = 2;
{$EXTERNALSYM TWOV_TOPBOTTOM}

```

```

TWOV_LEFTRIGHT = 3;
{$EXTERNALSYM TWOV_LEFTRIGHT}
TWOV_ALL      = 4;
{$EXTERNALSYM TWOV_ALL}

{ TW_FILESYSTEM.FileType values (FT_ means file type) }
TWFY_CAMERA    = 0;
{$EXTERNALSYM TWFY_CAMERA}
TWFY_CAMERATOP  = 1;
{$EXTERNALSYM TWFY_CAMERATOP}
TWFY_CAMERABOTTOM = 2;
{$EXTERNALSYM TWFY_CAMERABOTTOM}
TWFY_CAMERAPREVIEW = 3;
{$EXTERNALSYM TWFY_CAMERAPREVIEW}
TWFY_DOMAIN     = 4;
{$EXTERNALSYM TWFY_DOMAIN}
TWFY_HOST       = 5;
{$EXTERNALSYM TWFY_HOST}
TWFY_DIRECTORY  = 6;
{$EXTERNALSYM TWFY_DIRECTORY}
TWFY_IMAGE      = 7;
{$EXTERNALSYM TWFY_IMAGE}
TWFY_UNKNOWN    = 8;
{$EXTERNALSYM TWFY_UNKNOWN}

{ ICAP_JPEGQUALITY values (JQ_ means jpeg quality) }
TWJQ_UNKNOWN    = -4;
{$EXTERNALSYM TWJQ_UNKNOWN}
TWJQ_LOW        = -3;
{$EXTERNALSYM TWJQ_LOW}
TWJQ_MEDIUM     = -2;
{$EXTERNALSYM TWJQ_MEDIUM}
TWJQ_HIGH       = -1;
{$EXTERNALSYM TWJQ_HIGH}

{*****
* Country Constants                                     *
***** }

TWCY_AFGHANISTAN = 1001;
{$EXTERNALSYM TWCY_AFGHANISTAN}
TWCY_ALGERIA     = 213;
{$EXTERNALSYM TWCY_ALGERIA}
TWCY_AMERICANSAMOA = 684;
{$EXTERNALSYM TWCY_AMERICANSAMOA}
TWCY_ANDORRA     = 033;
{$EXTERNALSYM TWCY_ANDORRA}
TWCY_ANGOLA      = 1002;

```

```

{$EXTERNALSYM TWCY_ANGOLA}
TWCY_ANGUILLA    = 8090;
{$EXTERNALSYM TWCY_ANGUILLA}
TWCY_ANTIGUA     = 8091;
{$EXTERNALSYM TWCY_ANTIGUA}
TWCY_ARGENTINA   = 54;
{$EXTERNALSYM TWCY_ARGENTINA}
TWCY_ARUBA       = 297;
{$EXTERNALSYM TWCY_ARUBA}
TWCY_ASCENSIONI  = 247;
{$EXTERNALSYM TWCY_ASCENSIONI}
TWCY_AUSTRALIA   = 61;
{$EXTERNALSYM TWCY_AUSTRALIA}
TWCY_AUSTRIA     = 43;
{$EXTERNALSYM TWCY_AUSTRIA}
TWCY_BAHAMAS     = 8092;
{$EXTERNALSYM TWCY_BAHAMAS}
TWCY_BAHRAIN     = 973;
{$EXTERNALSYM TWCY_BAHRAIN}
TWCY_BANGLADESH  = 880;
{$EXTERNALSYM TWCY_BANGLADESH}
TWCY_BARBADOS    = 8093;
{$EXTERNALSYM TWCY_BARBADOS}
TWCY_BELGIUM     = 32;
{$EXTERNALSYM TWCY_BELGIUM}
TWCY_BELIZE      = 501;
{$EXTERNALSYM TWCY_BELIZE}
TWCY_BENIN       = 229;
{$EXTERNALSYM TWCY_BENIN}
TWCY_BERMUDA     = 8094;
{$EXTERNALSYM TWCY_BERMUDA}
TWCY_BHUTAN      = 1003;
{$EXTERNALSYM TWCY_BHUTAN}
TWCY_BOLIVIA     = 591;
{$EXTERNALSYM TWCY_BOLIVIA}
TWCY_BOTSWANA    = 267;
{$EXTERNALSYM TWCY_BOTSWANA}
TWCY_BRITAIN     = 6;
{$EXTERNALSYM TWCY_BRITAIN}
TWCY_BRITVIRGINIS = 8095;
{$EXTERNALSYM TWCY_BRITVIRGINIS}
TWCY_BRAZIL      = 55;
{$EXTERNALSYM TWCY_BRAZIL}
TWCY_BRUNEI      = 673;
{$EXTERNALSYM TWCY_BRUNEI}
TWCY_BULGARIA    = 359;
{$EXTERNALSYM TWCY_BULGARIA}
TWCY_BURKINAFASO = 1004;

```

```

{$EXTERNALSYM TWCY_BURKINAFASO}
TWCY_BURMA      = 1005;
{$EXTERNALSYM TWCY_BURMA}
TWCY_BURUNDI    = 1006;
{$EXTERNALSYM TWCY_BURUNDI}
TWCY_CAMAROON   = 237;
{$EXTERNALSYM TWCY_CAMAROON}
TWCY_CANADA     = 2;
{$EXTERNALSYM TWCY_CANADA}
TWCY_CAPEVERDEIS = 238;
{$EXTERNALSYM TWCY_CAPEVERDEIS}
TWCY_CAYMANIS   = 8096;
{$EXTERNALSYM TWCY_CAYMANIS}
TWCY_CENTRALAFREP = 1007;
{$EXTERNALSYM TWCY_CENTRALAFREP}
TWCY_CHAD       = 1008;
{$EXTERNALSYM TWCY_CHAD}
TWCY_CHILE      = 56;
{$EXTERNALSYM TWCY_CHILE}
TWCY_CHINA      = 86;
{$EXTERNALSYM TWCY_CHINA}
TWCY_CHRISTMASIS = 1009;
{$EXTERNALSYM TWCY_CHRISTMASIS}
TWCY_COCOSIS    = 1009;
{$EXTERNALSYM TWCY_COCOSIS}
TWCY_COLOMBIA   = 57;
{$EXTERNALSYM TWCY_COLOMBIA}
TWCY_COMOROS    = 1010;
{$EXTERNALSYM TWCY_COMOROS}
TWCY_CONGO      = 1011;
{$EXTERNALSYM TWCY_CONGO}
TWCY_COOKIS     = 1012;
{$EXTERNALSYM TWCY_COOKIS}
TWCY_COSTARICA  = 506 ;
{$EXTERNALSYM TWCY_COSTARICA}
TWCY_CUBA       = 005;
{$EXTERNALSYM TWCY_CUBA}
TWCY_CYPRIUS    = 357;
{$EXTERNALSYM TWCY_CYPRIUS}
TWCY_CZECHOSLOVAKIA = 42;
{$EXTERNALSYM TWCY_CZECHOSLOVAKIA}
TWCY_DENMARK    = 45;
{$EXTERNALSYM TWCY_DENMARK}
TWCY_DJIBOUTI   = 1013;
{$EXTERNALSYM TWCY_DJIBOUTI}
TWCY_DOMINICA   = 8097;
{$EXTERNALSYM TWCY_DOMINICA}
TWCY_DOMINICANREP = 8098;

```

```

{$EXTERNALSYM TWCY_DOMINCANREP}
TWCY_EASTERIS    = 1014;
{$EXTERNALSYM TWCY_EASTERIS}
TWCY_ECUADOR     = 593;
{$EXTERNALSYM TWCY_ECUADOR}
TWCY_EGYPT       = 20;
{$EXTERNALSYM TWCY_EGYPT}
TWCY_ELSALVADOR  = 503;
{$EXTERNALSYM TWCY_ELSALVADOR}
TWCY_EQGUINEA    = 1015;
{$EXTERNALSYM TWCY_EQGUINEA}
TWCY_ETHIOPIA    = 251;
{$EXTERNALSYM TWCY_ETHIOPIA}
TWCY_FALKLANDIS  = 1016;
{$EXTERNALSYM TWCY_FALKLANDIS}
TWCY_FAEROEIS    = 298;
{$EXTERNALSYM TWCY_FAEROEIS}
TWCY_FIJIISLANDS = 679;
{$EXTERNALSYM TWCY_FIJIISLANDS}
TWCY_FINLAND     = 358;
{$EXTERNALSYM TWCY_FINLAND}
TWCY_FRANCE      = 33;
{$EXTERNALSYM TWCY_FRANCE}
TWCY_FRANTILLES  = 596;
{$EXTERNALSYM TWCY_FRANTILLES}
TWCY_FRGUIANA    = 594;
{$EXTERNALSYM TWCY_FRGUIANA}
TWCY_FRPOLYNEISA = 689;
{$EXTERNALSYM TWCY_FRPOLYNEISA}
TWCY_FUTANAIS    = 1043;
{$EXTERNALSYM TWCY_FUTANAIS}
TWCY_GABON       = 241;
{$EXTERNALSYM TWCY_GABON}
TWCY_GAMBIA      = 220;
{$EXTERNALSYM TWCY_GAMBIA}
TWCY_GERMANY     = 49;
{$EXTERNALSYM TWCY_GERMANY}
TWCY_GHANA       = 233;
{$EXTERNALSYM TWCY_GHANA}
TWCY_GIBRALTER   = 350;
{$EXTERNALSYM TWCY_GIBRALTER}
TWCY_GREECE      = 30;
{$EXTERNALSYM TWCY_GREECE}
TWCY_GREENLAND   = 299;
{$EXTERNALSYM TWCY_GREENLAND}
TWCY_GRENADA     = 8099;
{$EXTERNALSYM TWCY_GRENADA}
TWCY_GRENEDES    = 8015;

```

```

{$EXTERNALSYM TWCY_GRENEDINES}
TWCY_GUADELOUPE = 590;
{$EXTERNALSYM TWCY_GUADELOUPE}
TWCY_GUAM = 671;
{$EXTERNALSYM TWCY_GUAM}
TWCY_GUANTANAMOBAY = 5399;
{$EXTERNALSYM TWCY_GUANTANAMOBAY}
TWCY_GUATEMALA = 502;
{$EXTERNALSYM TWCY_GUATEMALA}
TWCY_GUINEA = 224;
{$EXTERNALSYM TWCY_GUINEA}
TWCY_GUINEABISSAU = 1017;
{$EXTERNALSYM TWCY_GUINEABISSAU}
TWCY_GUYANA = 592;
{$EXTERNALSYM TWCY_GUYANA}
TWCY_HAITI = 509;
{$EXTERNALSYM TWCY_HAITI}
TWCY_HONDURAS = 504;
{$EXTERNALSYM TWCY_HONDURAS}
TWCY_HONGKONG = 852;
{$EXTERNALSYM TWCY_HONGKONG}
TWCY_HUNGARY = 36;
{$EXTERNALSYM TWCY_HUNGARY}
TWCY_ICELAND = 354;
{$EXTERNALSYM TWCY_ICELAND}
TWCY_INDIA = 91;
{$EXTERNALSYM TWCY_INDIA}
TWCY_INDONESIA = 62;
{$EXTERNALSYM TWCY_INDONESIA}
TWCY_IRAN = 98;
{$EXTERNALSYM TWCY_IRAN}
TWCY_IRAQ = 964;
{$EXTERNALSYM TWCY_IRAQ}
TWCY_IRELAND = 353;
{$EXTERNALSYM TWCY_IRELAND}
TWCY_ISRAEL = 972;
{$EXTERNALSYM TWCY_ISRAEL}
TWCY_ITALY = 39;
{$EXTERNALSYM TWCY_ITALY}
TWCY_IVORYCOAST = 225;
{$EXTERNALSYM TWCY_IVORYCOAST}
TWCY_JAMAICA = 8010;
{$EXTERNALSYM TWCY_JAMAICA}
TWCY_JAPAN = 81;
{$EXTERNALSYM TWCY_JAPAN}
TWCY_JORDAN = 962;
{$EXTERNALSYM TWCY_JORDAN}
TWCY_KENYA = 254;

```

```

{$EXTERNALSYM TWCY_KENYA}
TWCY_KIRIBATI    = 1018;
{$EXTERNALSYM TWCY_KIRIBATI}
TWCY_KOREA       = 82;
{$EXTERNALSYM TWCY_KOREA}
TWCY_KUWAIT      = 965;
{$EXTERNALSYM TWCY_KUWAIT}
TWCY_LAOS        = 1019;
{$EXTERNALSYM TWCY_LAOS}
TWCY_LEBANON     = 1020;
{$EXTERNALSYM TWCY_LEBANON}
TWCY_LIBERIA     = 231;
{$EXTERNALSYM TWCY_LIBERIA}
TWCY_LIBYA       = 218;
{$EXTERNALSYM TWCY_LIBYA}
TWCY_LIECHTENSTEIN = 41;
{$EXTERNALSYM TWCY_LIECHTENSTEIN}
TWCY_LUXENBOURG  = 352;
{$EXTERNALSYM TWCY_LUXENBOURG}
TWCY_MACAO       = 853;
{$EXTERNALSYM TWCY_MACAO}
TWCY_MADAGASCAR  = 1021;
{$EXTERNALSYM TWCY_MADAGASCAR}
TWCY_MALAWI      = 265;
{$EXTERNALSYM TWCY_MALAWI}
TWCY_MALAYSIA    = 60;
{$EXTERNALSYM TWCY_MALAYSIA}
TWCY_MALDIVES    = 960;
{$EXTERNALSYM TWCY_MALDIVES}
TWCY_MALI        = 1022;
{$EXTERNALSYM TWCY_MALI}
TWCY_MALTA       = 356;
{$EXTERNALSYM TWCY_MALTA}
TWCY_MARSHALLIS  = 692;
{$EXTERNALSYM TWCY_MARSHALLIS}
TWCY_MAUERITANIA = 1023;
{$EXTERNALSYM TWCY_MAUERITANIA}
TWCY_MAUERITIUS  = 230;
{$EXTERNALSYM TWCY_MAUERITIUS}
TWCY_MEXICO      = 3;
{$EXTERNALSYM TWCY_MEXICO}
TWCY_MICRONESIA  = 691;
{$EXTERNALSYM TWCY_MICRONESIA}
TWCY_MIQUELON    = 508;
{$EXTERNALSYM TWCY_MIQUELON}
TWCY_MONACO      = 33;
{$EXTERNALSYM TWCY_MONACO}
TWCY_MONGOLIA    = 1024;

```

```

{$EXTERNALSYM TWCY_MONGOLIA}
TWCY_MONTSEERRAT = 8011;
{$EXTERNALSYM TWCY_MONTSEERRAT}
TWCY_MOROCCO = 212;
{$EXTERNALSYM TWCY_MOROCCO}
TWCY_MOZAMBIQUE = 1025;
{$EXTERNALSYM TWCY_MOZAMBIQUE}
TWCY_NAMIBIA = 264;
{$EXTERNALSYM TWCY_NAMIBIA}
TWCY_NAURU = 1026;
{$EXTERNALSYM TWCY_NAURU}
TWCY_NEPAL = 977;
{$EXTERNALSYM TWCY_NEPAL}
TWCY_NETHERLANDS = 31;
{$EXTERNALSYM TWCY_NETHERLANDS}
TWCY_NETHANTILLES = 599;
{$EXTERNALSYM TWCY_NETHANTILLES}
TWCY_NEVIS = 8012;
{$EXTERNALSYM TWCY_NEVIS}
TWCY_NEWCALEDONIA = 687;
{$EXTERNALSYM TWCY_NEWCALEDONIA}
TWCY_NEWZEALAND = 64;
{$EXTERNALSYM TWCY_NEWZEALAND}
TWCY_NICARAGUA = 505;
{$EXTERNALSYM TWCY_NICARAGUA}
TWCY_NIGER = 227;
{$EXTERNALSYM TWCY_NIGER}
TWCY_NIGERIA = 234;
{$EXTERNALSYM TWCY_NIGERIA}
TWCY_NIUE = 1027;
{$EXTERNALSYM TWCY_NIUE}
TWCY_NORFOLKI = 1028;
{$EXTERNALSYM TWCY_NORFOLKI}
TWCY_NORWAY = 47;
{$EXTERNALSYM TWCY_NORWAY}
TWCY_OMAN = 968;
{$EXTERNALSYM TWCY_OMAN}
TWCY_PAKISTAN = 92;
{$EXTERNALSYM TWCY_PAKISTAN}
TWCY_PALAU = 1029;
{$EXTERNALSYM TWCY_PALAU}
TWCY_PANAMA = 507;
{$EXTERNALSYM TWCY_PANAMA}
TWCY_PARAGUAY = 595;
{$EXTERNALSYM TWCY_PARAGUAY}
TWCY_PERU = 51;
{$EXTERNALSYM TWCY_PERU}
TWCY_PHILLIPPINES = 63;

```

```

{$EXTERNALSYM TWCY_PHILLIPPINES}
TWCY_PITCAIRNIS = 1030;
{$EXTERNALSYM TWCY_PITCAIRNIS}
TWCY_PNEWGUINEA = 675;
{$EXTERNALSYM TWCY_PNEWGUINEA}
TWCY_POLAND = 48;
{$EXTERNALSYM TWCY_POLAND}
TWCY_PORTUGAL = 351;
{$EXTERNALSYM TWCY_PORTUGAL}
TWCY_QATAR = 974;
{$EXTERNALSYM TWCY_QATAR}
TWCY_REUNIONI = 1031;
{$EXTERNALSYM TWCY_REUNIONI}
TWCY_ROMANIA = 40;
{$EXTERNALSYM TWCY_ROMANIA}
TWCY_RWANDA = 250;
{$EXTERNALSYM TWCY_RWANDA}
TWCY_SAIPAN = 670;
{$EXTERNALSYM TWCY_SAIPAN}
TWCY_SANMARINO = 39;
{$EXTERNALSYM TWCY_SANMARINO}
TWCY_SAOTOME = 1033;
{$EXTERNALSYM TWCY_SAOTOME}
TWCY_SAUDIARABIA = 966;
{$EXTERNALSYM TWCY_SAUDIARABIA}
TWCY_SENEGAL = 221;
{$EXTERNALSYM TWCY_SENEGAL}
TWCY_SEYCHELLESIS = 1034;
{$EXTERNALSYM TWCY_SEYCHELLESIS}
TWCY_SIERRALEONE = 1035;
{$EXTERNALSYM TWCY_SIERRALEONE}
TWCY_SINGAPORE = 65;
{$EXTERNALSYM TWCY_SINGAPORE}
TWCY_SOLOMONIS = 1036;
{$EXTERNALSYM TWCY_SOLOMONIS}
TWCY_SOMALI = 1037;
{$EXTERNALSYM TWCY_SOMALI}
TWCY_SOUTHAFRICA = 27 ;
{$EXTERNALSYM TWCY_SOUTHAFRICA}
TWCY_SPAIN = 34;
{$EXTERNALSYM TWCY_SPAIN}
TWCY_SRILANKA = 94;
{$EXTERNALSYM TWCY_SRILANKA}
TWCY_STHELENA = 1032;
{$EXTERNALSYM TWCY_STHELENA}
TWCY_STKITTS = 8013;
{$EXTERNALSYM TWCY_STKITTS}
TWCY_STLUCIA = 8014;

```

```

{$EXTERNALSYM TWCY_STLUCIA}
TWCY_STPIERRE    = 508;
{$EXTERNALSYM TWCY_STPIERRE}
TWCY_STVINCENT   = 8015;
{$EXTERNALSYM TWCY_STVINCENT}
TWCY_SUDAN       = 1038;
{$EXTERNALSYM TWCY_SUDAN}
TWCY_SURINAME    = 597;
{$EXTERNALSYM TWCY_SURINAME}
TWCY_SWAZILAND   = 268;
{$EXTERNALSYM TWCY_SWAZILAND}
TWCY_SWEDEN      = 46;
{$EXTERNALSYM TWCY_SWEDEN}
TWCY_SWITZERLAND = 41;
{$EXTERNALSYM TWCY_SWITZERLAND}
TWCY_SYRIA       = 1039;
{$EXTERNALSYM TWCY_SYRIA}
TWCY_TAIWAN      = 886;
{$EXTERNALSYM TWCY_TAIWAN}
TWCY_TANZANIA    = 255;
{$EXTERNALSYM TWCY_TANZANIA}
TWCY_THAILAND    = 66;
{$EXTERNALSYM TWCY_THAILAND}
TWCY_TOBAGO      = 8016;
{$EXTERNALSYM TWCY_TOBAGO}
TWCY_TOGO        = 228;
{$EXTERNALSYM TWCY_TOGO}
TWCY_TONGAIS     = 676;
{$EXTERNALSYM TWCY_TONGAIS}
TWCY_TRINIDAD    = 8016;
{$EXTERNALSYM TWCY_TRINIDAD}
TWCY_TUNISIA     = 216;
{$EXTERNALSYM TWCY_TUNISIA}
TWCY_TURKEY      = 90;
{$EXTERNALSYM TWCY_TURKEY}
TWCY_TURKSCAICOS = 8017;
{$EXTERNALSYM TWCY_TURKSCAICOS}
TWCY_TUVALU      = 1040;
{$EXTERNALSYM TWCY_TUVALU}
TWCY_UGANDA      = 256;
{$EXTERNALSYM TWCY_UGANDA}
TWCY_USSR        = 7;
{$EXTERNALSYM TWCY_USSR}
TWCY_UAEMIRATES  = 971;
{$EXTERNALSYM TWCY_UAEMIRATES}
TWCY_UNITEDKINGDOM = 44;
{$EXTERNALSYM TWCY_UNITEDKINGDOM}
TWCY_USA         = 1;

```

```

{$EXTERNALSYM TWCY_USA}
TWCY_URUGUAY      = 598;
{$EXTERNALSYM TWCY_URUGUAY}
TWCY_VANUATU      = 1041;
{$EXTERNALSYM TWCY_VANUATU}
TWCY_VATICANCITY  = 39;
{$EXTERNALSYM TWCY_VATICANCITY}
TWCY_VENEZUELA    = 58;
{$EXTERNALSYM TWCY_VENEZUELA}
TWCY_WAKE          = 1042;
{$EXTERNALSYM TWCY_WAKE}
TWCY_WALLISIS     = 1043;
{$EXTERNALSYM TWCY_WALLISIS}
TWCY_WESTERNSAHARA = 1044;
{$EXTERNALSYM TWCY_WESTERNSAHARA}
TWCY_WESTERNSAMOA = 1045;
{$EXTERNALSYM TWCY_WESTERNSAMOA}
TWCY_YEMEN        = 1046;
{$EXTERNALSYM TWCY_YEMEN}
TWCY_YUGOSLAVIA   = 38;
{$EXTERNALSYM TWCY_YUGOSLAVIA}
TWCY_ZAIRE         = 243;
{$EXTERNALSYM TWCY_ZAIRE}
TWCY_ZAMBIA        = 260;
{$EXTERNALSYM TWCY_ZAMBIA}
TWCY_ZIMBABWE     = 263;
{$EXTERNALSYM TWCY_ZIMBABWE}

```

{ Added for 1.8 }

```

TWCY_ALBANIA      = 355;
{$EXTERNALSYM TWCY_ALBANIA}
TWCY_ARMENIA      = 374;
{$EXTERNALSYM TWCY_ARMENIA}
TWCY_AZERBAIJAN   = 994;
{$EXTERNALSYM TWCY_AZERBAIJAN}
TWCY_BELARUS      = 375;
{$EXTERNALSYM TWCY_BELARUS}
TWCY_BOSNIAHERZGO = 387;
{$EXTERNALSYM TWCY_BOSNIAHERZGO}
TWCY_CAMBODIA     = 855;
{$EXTERNALSYM TWCY_CAMBODIA}
TWCY_CROATIA      = 385;
{$EXTERNALSYM TWCY_CROATIA}
TWCY_CZECHREPUBLIC = 420;
{$EXTERNALSYM TWCY_CZECHREPUBLIC}
TWCY_DIEGOGARCIA  = 246;
{$EXTERNALSYM TWCY_DIEGOGARCIA}
TWCY_ERITREA      = 291;

```

```

{$EXTERNALSYM TWCY_ERITREA}
TWCY_ESTONIA    = 372;
{$EXTERNALSYM TWCY_ESTONIA}
TWCY_GEORGIA    = 995;
{$EXTERNALSYM TWCY_GEORGIA}
TWCY_LATVIA     = 371;
{$EXTERNALSYM TWCY_LATVIA}
TWCY_LESOTHO    = 266;
{$EXTERNALSYM TWCY_LESOTHO}
TWCY_LITHUANIA  = 370;
{$EXTERNALSYM TWCY_LITHUANIA}
TWCY_MACEDONIA  = 389;
{$EXTERNALSYM TWCY_MACEDONIA}
TWCY_MAYOTTEIS  = 269;
{$EXTERNALSYM TWCY_MAYOTTEIS}
TWCY_MOLDOVA    = 373;
{$EXTERNALSYM TWCY_MOLDOVA}
TWCY_MYANMAR     = 95 ;
{$EXTERNALSYM TWCY_MYANMAR}
TWCY_NORTHKOREA = 850;
{$EXTERNALSYM TWCY_NORTHKOREA}
TWCY_PUERTORICO = 787;
{$EXTERNALSYM TWCY_PUERTORICO}
TWCY_RUSSIA      = 7 ;
{$EXTERNALSYM TWCY_RUSSIA}
TWCY_SERBIA      = 381;
{$EXTERNALSYM TWCY_SERBIA}
TWCY_SLOVAKIA    = 421;
{$EXTERNALSYM TWCY_SLOVAKIA}
TWCY_SLOVENIA    = 386;
{$EXTERNALSYM TWCY_SLOVENIA}
TWCY_SOUTHKOREA  = 82 ;
{$EXTERNALSYM TWCY_SOUTHKOREA}
TWCY_UKRAINE     = 380;
{$EXTERNALSYM TWCY_UKRAINE}
TWCY_USVIRGINIS  = 340;
{$EXTERNALSYM TWCY_USVIRGINIS}
TWCY_VIETNAM     = 84 ;
{$EXTERNALSYM TWCY_VIETNAM}

```

```

{*****
* Language Constants                                     *
***** }

```

```

TWLG_DAN    = 0; { Danish }
{$EXTERNALSYM TWLG_DAN}
TWLG_DUT    = 1; { Dutch }
{$EXTERNALSYM TWLG_DUT}

```

```

TWLG_ENG    = 2; { International English }
{$EXTERNALSYM TWLG_ENG}
TWLG_FCF    = 3; { French Canadian }
{$EXTERNALSYM TWLG_FCF}
TWLG_FIN    = 4; { Finnish }
{$EXTERNALSYM TWLG_FIN}
TWLG_FRN    = 5; { French }
{$EXTERNALSYM TWLG_FRN}
TWLG_GER    = 6; { German }
{$EXTERNALSYM TWLG_GER}
TWLG_ICE    = 7; { Icelandic }
{$EXTERNALSYM TWLG_ICE}
TWLG_ITN    = 8; { Italian }
{$EXTERNALSYM TWLG_ITN}
TWLG_NOR    = 9; { Norwegian }
{$EXTERNALSYM TWLG_NOR}
TWLG_POR    = 10; { Portuguese }
{$EXTERNALSYM TWLG_POR}
TWLG_SPA    = 11; { Spanish }
{$EXTERNALSYM TWLG_SPA}
TWLG_SWE    = 12; { Swedish }
{$EXTERNALSYM TWLG_SWE}
TWLG_USA    = 13; { U.S. English }
{$EXTERNALSYM TWLG_USA}

```

```
{ Added for 1.8 }
```

```

TWLG_USERLOCALE    = -1;
{$EXTERNALSYM TWLG_USERLOCALE}
TWLG_AFRIKAANS      = 14;
{$EXTERNALSYM TWLG_AFRIKAANS}
TWLG_ALBANIA        = 15;
{$EXTERNALSYM TWLG_ALBANIA}
TWLG_ARABIC         = 16;
{$EXTERNALSYM TWLG_ARABIC}
TWLG_ARABIC_ALGERIA = 17;
{$EXTERNALSYM TWLG_ARABIC_ALGERIA}
TWLG_ARABIC_BAHRAIN = 18;
{$EXTERNALSYM TWLG_ARABIC_BAHRAIN}
TWLG_ARABIC_EGYPT   = 19;
{$EXTERNALSYM TWLG_ARABIC_EGYPT}
TWLG_ARABIC_IRAQ     = 20;
{$EXTERNALSYM TWLG_ARABIC_IRAQ}
TWLG_ARABIC_JORDAN   = 21;
{$EXTERNALSYM TWLG_ARABIC_JORDAN}
TWLG_ARABIC_KUWAIT   = 22;
{$EXTERNALSYM TWLG_ARABIC_KUWAIT}
TWLG_ARABIC_LEBANON  = 23;
{$EXTERNALSYM TWLG_ARABIC_LEBANON}

```

TWLG_ARABIC_LIBYA = 24;
 {\$EXTERNALSYM TWLG_ARABIC_LIBYA}
 TWLG_ARABIC_MOROCCO = 25;
 {\$EXTERNALSYM TWLG_ARABIC_MOROCCO}
 TWLG_ARABIC_OMAN = 26;
 {\$EXTERNALSYM TWLG_ARABIC_OMAN}
 TWLG_ARABIC_QATAR = 27;
 {\$EXTERNALSYM TWLG_ARABIC_QATAR}
 TWLG_ARABIC_SAUDIARABIA = 28;
 {\$EXTERNALSYM TWLG_ARABIC_SAUDIARABIA}
 TWLG_ARABIC_SYRIA = 29;
 {\$EXTERNALSYM TWLG_ARABIC_SYRIA}
 TWLG_ARABIC_TUNISIA = 30;
 {\$EXTERNALSYM TWLG_ARABIC_TUNISIA}
 TWLG_ARABIC_UAE = 31; { United Arabic Emirates }
 {\$EXTERNALSYM TWLG_ARABIC_UAE}
 TWLG_ARABIC_YEMEN = 32;
 {\$EXTERNALSYM TWLG_ARABIC_YEMEN}
 TWLG_BASQUE = 33;
 {\$EXTERNALSYM TWLG_BASQUE}
 TWLG_BYELORUSSIAN = 34;
 {\$EXTERNALSYM TWLG_BYELORUSSIAN}
 TWLG_BULGARIAN = 35;
 {\$EXTERNALSYM TWLG_BULGARIAN}
 TWLG_CATALAN = 36;
 {\$EXTERNALSYM TWLG_CATALAN}
 TWLG_CHINESE = 37;
 {\$EXTERNALSYM TWLG_CHINESE}
 TWLG_CHINESE_HONGKONG = 38;
 {\$EXTERNALSYM TWLG_CHINESE_HONGKONG}
 TWLG_CHINESE_PRC = 39; { People's Republic of China }
 {\$EXTERNALSYM TWLG_CHINESE_PRC}
 TWLG_CHINESE_SINGAPORE = 40;
 {\$EXTERNALSYM TWLG_CHINESE_SINGAPORE}
 TWLG_CHINESE_SIMPLIFIED = 41;
 {\$EXTERNALSYM TWLG_CHINESE_SIMPLIFIED}
 TWLG_CHINESE_TAIWAN = 42;
 {\$EXTERNALSYM TWLG_CHINESE_TAIWAN}
 TWLG_CHINESE_TRADITIONAL = 43;
 {\$EXTERNALSYM TWLG_CHINESE_TRADITIONAL}
 TWLG_CROATIA = 44;
 {\$EXTERNALSYM TWLG_CROATIA}
 TWLG_CZECH = 45;
 {\$EXTERNALSYM TWLG_CZECH}
 TWLG_DANISH = TWLG_DAN;
 {\$EXTERNALSYM TWLG_DANISH}
 TWLG_DUTCH = TWLG_DUT;
 {\$EXTERNALSYM TWLG_DUTCH}

```

TWLG_DUTCH_BELGIAN    = 46;
{$EXTERNALSYM TWLG_DUTCH_BELGIAN}
TWLG_ENGLISH          = TWLG_ENG;
{$EXTERNALSYM TWLG_ENGLISH}
TWLG_ENGLISH_AUSTRALIAN = 47;
{$EXTERNALSYM TWLG_ENGLISH_AUSTRALIAN}
TWLG_ENGLISH_CANADIAN  = 48;
{$EXTERNALSYM TWLG_ENGLISH_CANADIAN}
TWLG_ENGLISH_IRELAND   = 49;
{$EXTERNALSYM TWLG_ENGLISH_IRELAND}
TWLG_ENGLISH_NEWZEALAND = 50;
{$EXTERNALSYM TWLG_ENGLISH_NEWZEALAND}
TWLG_ENGLISH_SOUTHAFRICA = 51;
{$EXTERNALSYM TWLG_ENGLISH_SOUTHAFRICA}
TWLG_ENGLISH_UK        = 52;
{$EXTERNALSYM TWLG_ENGLISH_UK}
TWLG_ENGLISH_USA       = TWLG_USA;
{$EXTERNALSYM TWLG_ENGLISH_USA}
TWLG_ESTONIAN          = 53;
{$EXTERNALSYM TWLG_ESTONIAN}
TWLG_FAEROESE          = 54;
{$EXTERNALSYM TWLG_FAEROESE}
TWLG_FARSI             = 55;
{$EXTERNALSYM TWLG_FARSI}
TWLG_FINNISH           = TWLG_FIN;
{$EXTERNALSYM TWLG_FINNISH}
TWLG_FRENCH            = TWLG_FRN;
{$EXTERNALSYM TWLG_FRENCH}
TWLG_FRENCH_BELGIAN    = 56;
{$EXTERNALSYM TWLG_FRENCH_BELGIAN}
TWLG_FRENCH_CANADIAN   = TWLG_FCF;
{$EXTERNALSYM TWLG_FRENCH_CANADIAN}
TWLG_FRENCH_LUXEMBOURG = 57;
{$EXTERNALSYM TWLG_FRENCH_LUXEMBOURG}
TWLG_FRENCH_SWISS      = 58;
{$EXTERNALSYM TWLG_FRENCH_SWISS}
TWLG_GERMAN            = TWLG_GER;
{$EXTERNALSYM TWLG_GERMAN}
TWLG_GERMAN_AUSTRIAN   = 59;
{$EXTERNALSYM TWLG_GERMAN_AUSTRIAN}
TWLG_GERMAN_LUXEMBOURG = 60;
{$EXTERNALSYM TWLG_GERMAN_LUXEMBOURG}
TWLG_GERMAN_LIECHTENSTEIN = 61;
{$EXTERNALSYM TWLG_GERMAN_LIECHTENSTEIN}
TWLG_GERMAN_SWISS      = 62;
{$EXTERNALSYM TWLG_GERMAN_SWISS}
TWLG_GREEK             = 63;
{$EXTERNALSYM TWLG_GREEK}

```

```

TWLG_HEBREW          = 64;
{$EXTERNALSYM TWLG_HEBREW}
TWLG_HUNGARIAN       = 65;
{$EXTERNALSYM TWLG_HUNGARIAN}
TWLG_ICELANDIC       = TWLG_ICE;
{$EXTERNALSYM TWLG_ICELANDIC}
TWLG_INDONESIAN      = 66;
{$EXTERNALSYM TWLG_INDONESIAN}
TWLG_ITALIAN         = TWLG_ITN;
{$EXTERNALSYM TWLG_ITALIAN}
TWLG_ITALIAN_SWISS   = 67;
{$EXTERNALSYM TWLG_ITALIAN_SWISS}
TWLG_JAPANESE        = 68;
{$EXTERNALSYM TWLG_JAPANESE}
TWLG_KOREAN          = 69;
{$EXTERNALSYM TWLG_KOREAN}
TWLG_KOREAN_JOHAB    = 70;
{$EXTERNALSYM TWLG_KOREAN_JOHAB}
TWLG_LATVIAN         = 71;
{$EXTERNALSYM TWLG_LATVIAN}
TWLG_LITHUANIAN      = 72;
{$EXTERNALSYM TWLG_LITHUANIAN}
TWLG_NORWEGIAN       = TWLG_NOR;
{$EXTERNALSYM TWLG_NORWEGIAN}
TWLG_NORWEGIAN_BOKMAL = 73;
{$EXTERNALSYM TWLG_NORWEGIAN_BOKMAL}
TWLG_NORWEGIAN_NYNORSK = 74;
{$EXTERNALSYM TWLG_NORWEGIAN_NYNORSK}
TWLG_POLISH          = 75;
{$EXTERNALSYM TWLG_POLISH}
TWLG_PORTUGUESE      = TWLG_POR;
{$EXTERNALSYM TWLG_PORTUGUESE}
TWLG_PORTUGUESE_BRAZIL = 76;
{$EXTERNALSYM TWLG_PORTUGUESE_BRAZIL}
TWLG_ROMANIAN        = 77;
{$EXTERNALSYM TWLG_ROMANIAN}
TWLG_RUSSIAN         = 78;
{$EXTERNALSYM TWLG_RUSSIAN}
TWLG_SERBIAN_LATIN   = 79;
{$EXTERNALSYM TWLG_SERBIAN_LATIN}
TWLG_SLOVAK          = 80;
{$EXTERNALSYM TWLG_SLOVAK}
TWLG_SLOVENIAN       = 81;
{$EXTERNALSYM TWLG_SLOVENIAN}
TWLG_SPANISH         = TWLG_SPA;
{$EXTERNALSYM TWLG_SPANISH}
TWLG_SPANISH_MEXICAN = 82;
{$EXTERNALSYM TWLG_SPANISH_MEXICAN}

```

```

TWLG_SPANISH_MODERN    = 83;
{$EXTERNALSYM TWLG_SPANISH_MODERN}
TWLG_SWEDISH           = TWLG_SWE;
{$EXTERNALSYM TWLG_SWEDISH}
TWLG_THAI              = 84;
{$EXTERNALSYM TWLG_THAI}
TWLG_TURKISH           = 85;
{$EXTERNALSYM TWLG_TURKISH}
TWLG_UKRANIAN          = 86;
{$EXTERNALSYM TWLG_UKRANIAN}

```

{ More stuff added for 1.8 }

```

TWLG_ASSAMESE          = 87;
{$EXTERNALSYM TWLG_ASSAMESE}
TWLG_BENGALI           = 88;
{$EXTERNALSYM TWLG_BENGALI}
TWLG_BIHARI            = 89;
{$EXTERNALSYM TWLG_BIHARI}
TWLG_BODO              = 90;
{$EXTERNALSYM TWLG_BODO}
TWLG_DOGRI             = 91;
{$EXTERNALSYM TWLG_DOGRI}
TWLG_GUJARATI          = 92;
{$EXTERNALSYM TWLG_GUJARATI}
TWLG_HARYANVI          = 93;
{$EXTERNALSYM TWLG_HARYANVI}
TWLG_HINDI             = 94;
{$EXTERNALSYM TWLG_HINDI}
TWLG_KANNADA           = 95;
{$EXTERNALSYM TWLG_KANNADA}
TWLG_KASHMIRI          = 96;
{$EXTERNALSYM TWLG_KASHMIRI}
TWLG_MALAYALAM         = 97;
{$EXTERNALSYM TWLG_MALAYALAM}
TWLG_MARATHI           = 98;
{$EXTERNALSYM TWLG_MARATHI}
TWLG_MARWARI           = 99;
{$EXTERNALSYM TWLG_MARWARI}
TWLG_MEGHALAYAN        = 100;
{$EXTERNALSYM TWLG_MEGHALAYAN}
TWLG_MIZO              = 101;
{$EXTERNALSYM TWLG_MIZO}
TWLG_NAGA              = 102;
{$EXTERNALSYM TWLG_NAGA}
TWLG_ORISSI            = 103;
{$EXTERNALSYM TWLG_ORISSI}
TWLG_PUNJABI           = 104;
{$EXTERNALSYM TWLG_PUNJABI}

```

```

TWLG_PUSHTU          = 105;
{$EXTERNALSYM TWLG_PUSHTU}
TWLG_SERBIAN_CYRILLIC = 106;
{$EXTERNALSYM TWLG_SERBIAN_CYRILLIC}
TWLG_SIKKIMI         = 107;
{$EXTERNALSYM TWLG_SIKKIMI}
TWLG_SWEDISH_FINLAND = 108;
{$EXTERNALSYM TWLG_SWEDISH_FINLAND}
TWLG_TAMIL           = 109;
{$EXTERNALSYM TWLG_TAMIL}
TWLG_TELUGU          = 110;
{$EXTERNALSYM TWLG_TELUGU}
TWLG_TRIPURI         = 111;
{$EXTERNALSYM TWLG_TRIPURI}
TWLG_URDU            = 112;
{$EXTERNALSYM TWLG_URDU}
TWLG_VIETNAMESE      = 113;
{$EXTERNALSYM TWLG_VIETNAMESE}

```

```

{*****
* Data Groups                                     *
***** }

```

```

{ More Data Groups may be added in the future.
* Possible candidates include text, vector graphics, sound, etc.
* NOTE: Data Group constants must be powers of 2 as they are used
*   as bitflags when Application asks DSM to present a list of DSs.
}

```

```

DG_CONTROL = $0001; { data pertaining to control }
{$EXTERNALSYM DG_CONTROL}
DG_IMAGE   = $0002; { data pertaining to raster images }
{$EXTERNALSYM DG_IMAGE}
{ Added 1.8 }
DG_AUDIO   = $0004; { data pertaining to audio }
{$EXTERNALSYM DG_AUDIO}

```

```

{*****
* Data Argument Types                             *
***** }

```

```

{ SDH - 03/23/95 - WATCH }
{ The thunker requires knowledge about size of data being passed in the }
{ lpData parameter to DS_Entry (which is not readily available due to }
{ type LPVOID. Thus, we key off the DAT_ argument to determine the size. }
{ This has a couple implications: }
{ 1) Any additional DAT_ features require modifications to the thunk code }
{ for thunker support. }

```

```
{ 2) Any applications which use the custom capabilities are not supported }
{ under thinking since we have no way of knowing what size data (if }
{ any) is being passed. }
```

```
DAT_NULL          = $0000; { No data or structure. }
{$EXTERNALSYM DAT_NULL}
DAT_CUSTOMBASE    = $8000; { Base of custom DATs. }
{$EXTERNALSYM DAT_CUSTOMBASE}
```

```
{ Data Argument Types for the DG_CONTROL Data Group. }
DAT_CAPABILITY    = $0001; { TW_CAPABILITY }
{$EXTERNALSYM DAT_CAPABILITY}
DAT_EVENT         = $0002; { TW_EVENT }
{$EXTERNALSYM DAT_EVENT}
DAT_IDENTITY      = $0003; { TW_IDENTITY }
{$EXTERNALSYM DAT_IDENTITY}
DAT_PARENT       = $0004; { TW_HANDLE, application win handle in Windows }
{$EXTERNALSYM DAT_PARENT}
DAT_PENDINGXFER   = $0005; { TW_PENDINGXFER }
{$EXTERNALSYM DAT_PENDINGXFER}
DAT_SETUPMEMXFER  = $0006; { TW_SETUPMEMXFER }
{$EXTERNALSYM DAT_SETUPMEMXFER}
DAT_SETUPFILEXFER = $0007; { TW_SETUPFILEXFER }
{$EXTERNALSYM DAT_SETUPFILEXFER}
DAT_STATUS       = $0008; { TW_STATUS }
{$EXTERNALSYM DAT_STATUS}
DAT_USERINTERFACE = $0009; { TW_USERINTERFACE }
{$EXTERNALSYM DAT_USERINTERFACE}
DAT_XFERGROUP     = $000a; { TW_UINT32 }
{$EXTERNALSYM DAT_XFERGROUP}
{ SDH - 03/21/95 - TWUNK }
{ Additional message required for thunker to request the special }
{ identity information. }
DAT_TWUNKIDENTITY = $000b; { TW_TWUNKIDENTITY }
{$EXTERNALSYM DAT_TWUNKIDENTITY}
DAT_CUSTOMDSDATA  = $000c; { TW_CUSTOMDSDATA. }
{$EXTERNALSYM DAT_CUSTOMDSDATA}
```

```
{ Added 1.8 }
DAT_DEVICEEVENT   = $000d; { TW_DEVICEEVENT }
{$EXTERNALSYM DAT_DEVICEEVENT}
DAT_FILESYSTEM    = $000e; { TW_FILESYSTEM }
{$EXTERNALSYM DAT_FILESYSTEM}
DAT_PASSTHRU      = $000f; { TW_PASSTHRU }
{$EXTERNALSYM DAT_PASSTHRU}
```

```
{ Data Argument Types for the DG_IMAGE Data Group. }
DAT_IMAGEINFO     = $0101; { TW_IMAGEINFO }
```

```
{ $EXTERNALSYM DAT_IMAGEINFO }
DAT_IMAGELAYOUT = $0102; { TW_IMAGELAYOUT }
{ $EXTERNALSYM DAT_IMAGELAYOUT }
DAT_IMAGEMEMXFER = $0103; { TW_IMAGEMEMXFER }
{ $EXTERNALSYM DAT_IMAGEMEMXFER }
DAT_IMAGENATIVEXFER = $0104; { TW_UINT32 loword is hDIB, PICHandle }
{ $EXTERNALSYM DAT_IMAGENATIVEXFER }
DAT_IMAGEFILEXFER = $0105; { Null data }
{ $EXTERNALSYM DAT_IMAGEFILEXFER }
DAT_CIECOLOR = $0106; { TW_CIECOLOR }
{ $EXTERNALSYM DAT_CIECOLOR }
DAT_GRAYRESPONSE = $0107; { TW_GRAYRESPONSE }
{ $EXTERNALSYM DAT_GRAYRESPONSE }
DAT_RGBRESPONSE = $0108; { TW_RGBRESPONSE }
{ $EXTERNALSYM DAT_RGBRESPONSE }
DAT_JPEGCOMPRESSION = $0109; { TW_JPEGCOMPRESSION }
{ $EXTERNALSYM DAT_JPEGCOMPRESSION }
DAT_PALETTE8 = $010a; { TW_PALETTE8 }
{ $EXTERNALSYM DAT_PALETTE8 }
DAT_EXTIMAGEINFO = $010b; { TW_EXTIMAGEINFO -- for 1.7 Spec. }
{ $EXTERNALSYM DAT_EXTIMAGEINFO }
```

```
{ Added 1.8 }
```

```
{ Data Argument Types for the DG_AUDIO Data Group. }
```

```
DAT_AUDIOFILEXFER = $0201; { Null data }
{ $EXTERNALSYM DAT_AUDIOFILEXFER }
DAT_AUDIOINFO = $0202; { TW_AUDIOINFO }
{ $EXTERNALSYM DAT_AUDIOINFO }
DAT_AUDIONATIVEXFER = $0203; { TW_UINT32 handle to WAV, (AIFF Mac) }
{ $EXTERNALSYM DAT_AUDIONATIVEXFER }
```

```
{ Added 1.9 }
```

```
DAT_SETUPFILEXFER2 = $0301; { New file xfer operation }
{ $EXTERNALSYM DAT_SETUPFILEXFER2 }
```

```
{ *****
* Messages *
***** }
```

```
{ All message constants are unique.
```

```
* Messages are grouped according to which DATs they are used with. }
```

```
MSG_NULL = $0000; { Used in TW_EVENT structure }
{ $EXTERNALSYM MSG_NULL }
MSG_CUSTOMBASE = $8000; { Base of custom messages }
{ $EXTERNALSYM MSG_CUSTOMBASE }
```

```
{ Generic messages may be used with any of several DATs. }
```

```

MSG_GET          = $0001; { Get one or more values }
{$EXTERNALSYM MSG_GET}
MSG_GETCURRENT   = $0002; { Get current value }
{$EXTERNALSYM MSG_GETCURRENT}
MSG_GETDEFAULT   = $0003; { Get default (e.g. power up) value }
{$EXTERNALSYM MSG_GETDEFAULT}
MSG_GETFIRST     = $0004; { Get first of a series of items, e.g. DSs }
{$EXTERNALSYM MSG_GETFIRST}
MSG_GETNEXT      = $0005; { Iterate through a series of items. }
{$EXTERNALSYM MSG_GETNEXT}
MSG_SET          = $0006; { Set one or more values }
{$EXTERNALSYM MSG_SET}
MSG_RESET        = $0007; { Set current value to default value }
{$EXTERNALSYM MSG_RESET}
MSG_QUERYUPPORT  = $0008; { Get supported operations on the cap. }
{$EXTERNALSYM MSG_QUERYUPPORT}

{ Messages used with DAT_NULL }
MSG_XFERREADY    = $0101; { The data source has data ready }
{$EXTERNALSYM MSG_XFERREADY}
MSG_CLOSEDREQ    = $0102; { Request for Application. to close DS }
{$EXTERNALSYM MSG_CLOSEDREQ}
MSG_CLOSEDOK     = $0103; { Tell the Application. to save the state. }
{$EXTERNALSYM MSG_CLOSEDOK}
{ Added 1.8 }
MSG_DEVICEEVENT  = $0104; { Some event has taken place }
{$EXTERNALSYM MSG_DEVICEEVENT}

{ Messages used with a pointer to a DAT_STATUS structure }
MSG_CHECKSTATUS  = $0201; { Get status information }
{$EXTERNALSYM MSG_CHECKSTATUS}

{ Messages used with a pointer to DAT_PARENT data }
MSG_OPENDSM      = $0301; { Open the DSM }
{$EXTERNALSYM MSG_OPENDSM}
MSG_CLOSEDSM     = $0302; { Close the DSM }
{$EXTERNALSYM MSG_CLOSEDSM}

{ Messages used with a pointer to a DAT_IDENTITY structure }
MSG_OPENDS       = $0401; { Open a data source }
{$EXTERNALSYM MSG_OPENDS}
MSG_CLOSEDS      = $0402; { Close a data source }
{$EXTERNALSYM MSG_CLOSEDS}
MSG_USERSELECT   = $0403; { Put up a dialog of all DS }
{$EXTERNALSYM MSG_USERSELECT}

{ Messages used with a pointer to a DAT_USERINTERFACE structure }
MSG_DISABLED     = $0501; { Disable data transfer in the DS }

```

```

{$EXTERNALSYM MSG_DISABLED}
MSG_ENABLEDS      = $0502; { Enable data transfer in the DS }
{$EXTERNALSYM MSG_ENABLEDS}
MSG_ENABLEDSUIONLY = $0503; { Enable for saving DS state only. }
{$EXTERNALSYM MSG_ENABLEDSUIONLY}

{ Messages used with a pointer to a DAT_EVENT structure }
MSG_PROCESSEVENT  = $0601;
{$EXTERNALSYM MSG_PROCESSEVENT}

{ Messages used with a pointer to a DAT_PENDINGXFERS structure }
MSG_ENDXFER       = $0701;
{$EXTERNALSYM MSG_ENDXFER}
MSG_STOPFEEDER    = $0702;
{$EXTERNALSYM MSG_STOPFEEDER}

{ Added 1.8 }
{ Messages used with a pointer to a DAT_FILESYSTEM structure }
MSG_CHANGEDIRECTORY = $0801;
{$EXTERNALSYM MSG_CHANGEDIRECTORY}
MSG_CREATEDIRECTORY = $0802;
{$EXTERNALSYM MSG_CREATEDIRECTORY}
MSG_DELETE         = $0803;
{$EXTERNALSYM MSG_DELETE}
MSG_FORMATMEDIA    = $0804;
{$EXTERNALSYM MSG_FORMATMEDIA}
MSG_GETCLOSE       = $0805;
{$EXTERNALSYM MSG_GETCLOSE}
MSG_GETFIRSTFILE   = $0806;
{$EXTERNALSYM MSG_GETFIRSTFILE}
MSG_GETINFO        = $0807;
{$EXTERNALSYM MSG_GETINFO}
MSG_GETNEXTFILE    = $0808;
{$EXTERNALSYM MSG_GETNEXTFILE}
MSG_RENAME         = $0809;
{$EXTERNALSYM MSG_RENAME}
MSG_COPY           = $080A;
{$EXTERNALSYM MSG_COPY}
MSG_AUTOMATICCAPTUREDDIRECTORY = $080B;
{$EXTERNALSYM MSG_AUTOMATICCAPTUREDDIRECTORY}

{ Messages used with a pointer to a DAT_PASSTHRU structure }
MSG_PASSTHRU      = $0901;
{$EXTERNALSYM MSG_PASSTHRU}

{*****
* Capabilities                                     *
*****}

```

CAP_CUSTOMBASE = \$8000; { Base of custom capabilities }
 {\$EXTERNALSYM CAP_CUSTOMBASE}

{ all data sources are REQUIRED to support these caps }

CAP_XFERCOUNT = \$0001;
 {\$EXTERNALSYM CAP_XFERCOUNT}

{ image data sources are REQUIRED to support these caps }

ICAP_COMPRESSION = \$0100;
 {\$EXTERNALSYM ICAP_COMPRESSION}
 ICAP_PIXELTYPE = \$0101;
 {\$EXTERNALSYM ICAP_PIXELTYPE}
 ICAP_UNITS = \$0102; { default is TWUN_INCHES }
 {\$EXTERNALSYM ICAP_UNITS}
 ICAP_XFERMECH = \$0103;
 {\$EXTERNALSYM ICAP_XFERMECH}

{ all data sources MAY support these caps }

CAP_AUTHOR = \$1000;
 {\$EXTERNALSYM CAP_AUTHOR}
 CAP_CAPTION = \$1001;
 {\$EXTERNALSYM CAP_CAPTION}
 CAP_FEEDERENABLED = \$1002;
 {\$EXTERNALSYM CAP_FEEDERENABLED}
 CAP_FEEDERLOADED = \$1003;
 {\$EXTERNALSYM CAP_FEEDERLOADED}
 CAP_TIMEDATE = \$1004;
 {\$EXTERNALSYM CAP_TIMEDATE}
 CAP_SUPPORTEDCAPS = \$1005;
 {\$EXTERNALSYM CAP_SUPPORTEDCAPS}
 CAP_EXTENDED CAPS = \$1006;
 {\$EXTERNALSYM CAP_EXTENDED CAPS}
 CAP_AUTOFEED = \$1007;
 {\$EXTERNALSYM CAP_AUTOFEED}
 CAP_CLEARPAGE = \$1008;
 {\$EXTERNALSYM CAP_CLEARPAGE}
 CAP_FEEDPAGE = \$1009;
 {\$EXTERNALSYM CAP_FEEDPAGE}
 CAP_REWINDPAGE = \$100a;
 {\$EXTERNALSYM CAP_REWINDPAGE}
 CAP_INDICATORS = \$100b; { Added 1.1 }
 {\$EXTERNALSYM CAP_INDICATORS}
 CAP_SUPPORTEDCAPSEXT = \$100c; { Added 1.6 }
 {\$EXTERNALSYM CAP_SUPPORTEDCAPSEXT}
 CAP_PAPERDETECTABLE = \$100d; { Added 1.6 }
 {\$EXTERNALSYM CAP_PAPERDETECTABLE}
 CAP_UICONTROLLABLE = \$100e; { Added 1.6 }

```

{$EXTERNALSYM CAP_UICONTROLLABLE}
CAP_DEVICEONLINE          = $100f; { Added 1.6 }
{$EXTERNALSYM CAP_DEVICEONLINE}
CAP_AUTOSCAN              = $1010; { Added 1.6 }
{$EXTERNALSYM CAP_AUTOSCAN}
CAP_THUMBNAILSENABLED     = $1011; { Added 1.7 }
{$EXTERNALSYM CAP_THUMBNAILSENABLED}
CAP_DUPLEX                 = $1012; { Added 1.7 }
{$EXTERNALSYM CAP_DUPLEX}
CAP_DUPLEXENABLED         = $1013; { Added 1.7 }
{$EXTERNALSYM CAP_DUPLEXENABLED}
CAP_ENABLEDSUIONLY        = $1014; { Added 1.7 }
{$EXTERNALSYM CAP_ENABLEDSUIONLY}
CAP_CUSTOMDSDATA          = $1015; { Added 1.7 }
{$EXTERNALSYM CAP_CUSTOMDSDATA}
CAP_ENDORSER              = $1016; { Added 1.7 }
{$EXTERNALSYM CAP_ENDORSER}
CAP_JOBCONTROL            = $1017; { Added 1.7 }
{$EXTERNALSYM CAP_JOBCONTROL}
CAP_ALARMS                = $1018; { Added 1.8 }
{$EXTERNALSYM CAP_ALARMS}
CAP_ALARMVOLUME           = $1019; { Added 1.8 }
{$EXTERNALSYM CAP_ALARMVOLUME}
CAP_AUTOMATICCAPTURE      = $101a; { Added 1.8 }
{$EXTERNALSYM CAP_AUTOMATICCAPTURE}
CAP_TIMEBEFOREFIRSTCAPTURE = $101b; { Added 1.8 }
{$EXTERNALSYM CAP_TIMEBEFOREFIRSTCAPTURE}
CAP_TIMEBETWEENCAPTURES   = $101c; { Added 1.8 }
{$EXTERNALSYM CAP_TIMEBETWEENCAPTURES}
CAP_CLEARBUFFERS          = $101d; { Added 1.8 }
{$EXTERNALSYM CAP_CLEARBUFFERS}
CAP_MAXBATCHBUFFERS       = $101e; { Added 1.8 }
{$EXTERNALSYM CAP_MAXBATCHBUFFERS}
CAP_DEVICETIMEDATE        = $101f; { Added 1.8 }
{$EXTERNALSYM CAP_DEVICETIMEDATE}
CAP_POWERSUPPLY           = $1020; { Added 1.8 }
{$EXTERNALSYM CAP_POWERSUPPLY}
CAP_CAMERAPREVIEWUI       = $1021; { Added 1.8 }
{$EXTERNALSYM CAP_CAMERAPREVIEWUI}
CAP_DEVICEEVENT           = $1022; { Added 1.8 }
{$EXTERNALSYM CAP_DEVICEEVENT}
CAP_SERIALNUMBER          = $1024; { Added 1.8 }
{$EXTERNALSYM CAP_SERIALNUMBER}
CAP_PRINTER               = $1026; { Added 1.8 }
{$EXTERNALSYM CAP_PRINTER}
CAP_PRINTERENABLED        = $1027; { Added 1.8 }
{$EXTERNALSYM CAP_PRINTERENABLED}
CAP_PRINTERINDEX          = $1028; { Added 1.8 }

```

```

{$EXTERNALSYM CAP_PRINTERINDEX}
CAP_PRINTERMODE          = $1029; { Added 1.8 }
{$EXTERNALSYM CAP_PRINTERMODE}
CAP_PRINTERSTRING        = $102a; { Added 1.8 }
{$EXTERNALSYM CAP_PRINTERSTRING}
CAP_PRINTERSUFFIX         = $102b; { Added 1.8 }
{$EXTERNALSYM CAP_PRINTERSUFFIX}
CAP_LANGUAGE              = $102c; { Added 1.8 }
{$EXTERNALSYM CAP_LANGUAGE}
CAP_FEEDERALIGNMENT       = $102d; { Added 1.8 }
{$EXTERNALSYM CAP_FEEDERALIGNMENT}
CAP_FEEDERORDER           = $102e; { Added 1.8 }
{$EXTERNALSYM CAP_FEEDERORDER}
CAP_REACQUIREALLOWED     = $1030; { Added 1.8 }
{$EXTERNALSYM CAP_REACQUIREALLOWED}
CAP_BATTERYMINUTES        = $1032; { Added 1.8 }
{$EXTERNALSYM CAP_BATTERYMINUTES}
CAP_BATTERYPERCENTAGE     = $1033; { Added 1.8 }
{$EXTERNALSYM CAP_BATTERYPERCENTAGE}

```

{ image data sources MAY support these caps }

```

ICAP_AUTOBRIGHT          = $1100;
{$EXTERNALSYM ICAP_AUTOBRIGHT}
ICAP_BRIGHTNESS          = $1101;
{$EXTERNALSYM ICAP_BRIGHTNESS}
ICAP_CONTRAST             = $1103;
{$EXTERNALSYM ICAP_CONTRAST}
ICAP_CUSTHALFTONE        = $1104;
{$EXTERNALSYM ICAP_CUSTHALFTONE}
ICAP_EXPOSURETIME        = $1105;
{$EXTERNALSYM ICAP_EXPOSURETIME}
ICAP_FILTER               = $1106;
{$EXTERNALSYM ICAP_FILTER}
ICAP_FLASHUSED            = $1107;
{$EXTERNALSYM ICAP_FLASHUSED}
ICAP_GAMMA                = $1108;
{$EXTERNALSYM ICAP_GAMMA}
ICAP_HALFTONES            = $1109;
{$EXTERNALSYM ICAP_HALFTONES}
ICAP_HIGHLIGHT            = $110a;
{$EXTERNALSYM ICAP_HIGHLIGHT}
ICAP_IMAGEFILEFORMAT      = $110c;
{$EXTERNALSYM ICAP_IMAGEFILEFORMAT}
ICAP_LAMPSTATE            = $110d;
{$EXTERNALSYM ICAP_LAMPSTATE}
ICAP_LIGHTSOURCE          = $110e;
{$EXTERNALSYM ICAP_LIGHTSOURCE}
ICAP_ORIENTATION          = $1110;

```

```

{$EXTERNALSYM ICAP_ORIENTATION}
ICAP_PHYSICALWIDTH      = $1111;
{$EXTERNALSYM ICAP_PHYSICALWIDTH}
ICAP_PHYSICALHEIGHT     = $1112;
{$EXTERNALSYM ICAP_PHYSICALHEIGHT}
ICAP_SHADOW             = $1113;
{$EXTERNALSYM ICAP_SHADOW}
ICAP_FRAMES              = $1114;
{$EXTERNALSYM ICAP_FRAMES}
ICAP_XNATIVERESOLUTION  = $1116;
{$EXTERNALSYM ICAP_XNATIVERESOLUTION}
ICAP_YNATIVERESOLUTION  = $1117;
{$EXTERNALSYM ICAP_YNATIVERESOLUTION}
ICAP_XRESOLUTION         = $1118;
{$EXTERNALSYM ICAP_XRESOLUTION}
ICAP_YRESOLUTION        = $1119;
{$EXTERNALSYM ICAP_YRESOLUTION}
ICAP_MAXFRAMES           = $111a;
{$EXTERNALSYM ICAP_MAXFRAMES}
ICAP_TILES               = $111b;
{$EXTERNALSYM ICAP_TILES}
ICAP_BITORDER            = $111c;
{$EXTERNALSYM ICAP_BITORDER}
ICAP_CCITTKFACTOR       = $111d;
{$EXTERNALSYM ICAP_CCITTKFACTOR}
ICAP_LIGHTPATH           = $111e;
{$EXTERNALSYM ICAP_LIGHTPATH}
ICAP_PIXELFLAVOR        = $111f;
{$EXTERNALSYM ICAP_PIXELFLAVOR}
ICAP_PLANARCHUNKY       = $1120;
{$EXTERNALSYM ICAP_PLANARCHUNKY}
ICAP_ROTATION            = $1121;
{$EXTERNALSYM ICAP_ROTATION}
ICAP_SUPPORTEDSIZES      = $1122;
{$EXTERNALSYM ICAP_SUPPORTEDSIZES}
ICAP_THRESHOLD           = $1123;
{$EXTERNALSYM ICAP_THRESHOLD}
ICAP_XSCALING            = $1124;
{$EXTERNALSYM ICAP_XSCALING}
ICAP_YSCALING            = $1125;
{$EXTERNALSYM ICAP_YSCALING}
ICAP_BITORDERCODES      = $1126;
{$EXTERNALSYM ICAP_BITORDERCODES}
ICAP_PIXELFLAVORCODES   = $1127;
{$EXTERNALSYM ICAP_PIXELFLAVORCODES}
ICAP_JPEGPIXELTYPE       = $1128;
{$EXTERNALSYM ICAP_JPEGPIXELTYPE}
ICAP_TIMEFILL            = $112a;

```

```

{$EXTERNALSYM ICAP_TIMEFILL}
ICAP_BITDEPTH = $112b;
{$EXTERNALSYM ICAP_BITDEPTH}
ICAP_BITDEPTHRREDUCTION = $112c; { Added 1.5 }
{$EXTERNALSYM ICAP_BITDEPTHRREDUCTION}
ICAP_UNDEFINEDIMAGESIZE = $112d; { Added 1.6 }
{$EXTERNALSYM ICAP_UNDEFINEDIMAGESIZE}
ICAP_IMAGEDATASET = $112e; { Added 1.7 }
{$EXTERNALSYM ICAP_IMAGEDATASET}
ICAP_EXTIMAGEINFO = $112f; { Added 1.7 }
{$EXTERNALSYM ICAP_EXTIMAGEINFO}
ICAP_MINIMUMHEIGHT = $1130; { Added 1.7 }
{$EXTERNALSYM ICAP_MINIMUMHEIGHT}
ICAP_MINIMUMWIDTH = $1131; { Added 1.7 }
{$EXTERNALSYM ICAP_MINIMUMWIDTH}
ICAP_FLIPROTATION = $1136; { Added 1.8 }
{$EXTERNALSYM ICAP_FLIPROTATION}
ICAP_BARCODEDETECTIONENABLED = $1137; { Added 1.8 }
{$EXTERNALSYM ICAP_BARCODEDETECTIONENABLED}
ICAP_SUPPORTEDBARCODETYPES = $1138; { Added 1.8 }
{$EXTERNALSYM ICAP_SUPPORTEDBARCODETYPES}
ICAP_BARCODEMAXSEARCHPRIORITIES = $1139; { Added 1.8 }
{$EXTERNALSYM ICAP_BARCODEMAXSEARCHPRIORITIES}
ICAP_BARCODESEARCHPRIORITIES = $113a; { Added 1.8 }
{$EXTERNALSYM ICAP_BARCODESEARCHPRIORITIES}
ICAP_BARCODESEARCHMODE = $113b; { Added 1.8 }
{$EXTERNALSYM ICAP_BARCODESEARCHMODE}
ICAP_BARCODEMAXRETRIES = $113c; { Added 1.8 }
{$EXTERNALSYM ICAP_BARCODEMAXRETRIES}
ICAP_BARCODETIMEOUT = $113d; { Added 1.8 }
{$EXTERNALSYM ICAP_BARCODETIMEOUT}
ICAP_ZOOMFACTOR = $113e; { Added 1.8 }
{$EXTERNALSYM ICAP_ZOOMFACTOR}
ICAP_PATCHCODEDETECTIONENABLED = $113f; { Added 1.8 }
{$EXTERNALSYM ICAP_PATCHCODEDETECTIONENABLED}
ICAP_SUPPORTEDPATCHCODETYPES = $1140; { Added 1.8 }
{$EXTERNALSYM ICAP_SUPPORTEDPATCHCODETYPES}
ICAP_PATCHCODEMAXSEARCHPRIORITIES = $1141; { Added 1.8 }
{$EXTERNALSYM ICAP_PATCHCODEMAXSEARCHPRIORITIES}
ICAP_PATCHCODESEARCHPRIORITIES = $1142; { Added 1.8 }
{$EXTERNALSYM ICAP_PATCHCODESEARCHPRIORITIES}
ICAP_PATCHCODESEARCHMODE = $1143; { Added 1.8 }
{$EXTERNALSYM ICAP_PATCHCODESEARCHMODE}
ICAP_PATCHCODEMAXRETRIES = $1144; { Added 1.8 }
{$EXTERNALSYM ICAP_PATCHCODEMAXRETRIES}
ICAP_PATCHCODETIMEOUT = $1145; { Added 1.8 }
{$EXTERNALSYM ICAP_PATCHCODETIMEOUT}
ICAP_FLASHUSED2 = $1146; { Added 1.8 }

```

```

{$EXTERNALSYM ICAP_FLASHUSED2}
ICAP_IMAGEFILTER          = $1147; { Added 1.8 }
{$EXTERNALSYM ICAP_IMAGEFILTER}
ICAP_NOISEFILTER          = $1148; { Added 1.8 }
{$EXTERNALSYM ICAP_NOISEFILTER}
ICAP_OVERSCAN             = $1149; { Added 1.8 }
{$EXTERNALSYM ICAP_OVERSCAN}
ICAP_AUTOMATICBORDERDETECTION = $1150; { Added 1.8 }
{$EXTERNALSYM ICAP_AUTOMATICBORDERDETECTION}
ICAP_AUTOMATICDESKEW      = $1151; { Added 1.8 }
{$EXTERNALSYM ICAP_AUTOMATICDESKEW}
ICAP_AUTOMATICROTATE      = $1152; { Added 1.8 }
{$EXTERNALSYM ICAP_AUTOMATICROTATE}
ICAP_JPEGQUALITY          = $1153; { Added 1.9 }
{$EXTERNALSYM ICAP_JPEGQUALITY}

```

```

{ image data sources MAY support these audio caps }
ACAP_AUDIOFILEFORMAT      = $1201; { Added 1.8 }
{$EXTERNALSYM ACAP_AUDIOFILEFORMAT}
ACAP_XFERMECH              = $1202; { Added 1.8 }
{$EXTERNALSYM ACAP_XFERMECH}

```

```

{ -----

```

Version 1.7: Following is Extended Image Info Attributes.
 July 1997
 KHL

```

----- }

```

```

TWEI_BARCODEX             = $1200;
{$EXTERNALSYM TWEI_BARCODEX}
TWEI_BARCODEY             = $1201;
{$EXTERNALSYM TWEI_BARCODEY}
TWEI_BARCODETEXT          = $1202;
{$EXTERNALSYM TWEI_BARCODETEXT}
TWEI_BARCODETYPE          = $1203;
{$EXTERNALSYM TWEI_BARCODETYPE}
TWEI_DESHADETOP           = $1204;
{$EXTERNALSYM TWEI_DESHADETOP}
TWEI_DESHADELEFT          = $1205;
{$EXTERNALSYM TWEI_DESHADELEFT}
TWEI_DESHADEHEIGHT        = $1206;
{$EXTERNALSYM TWEI_DESHADEHEIGHT}
TWEI_DESHADEWIDTH         = $1207;
{$EXTERNALSYM TWEI_DESHADEWIDTH}
TWEI_DESHADESIZE          = $1208;
{$EXTERNALSYM TWEI_DESHADESIZE}

```

```

TWEI_SPECKLESREMOVED = $1209;
{$EXTERNALSYM TWEI_SPECKLESREMOVED}
TWEI_HORZLINEXCOORD = $120A;
{$EXTERNALSYM TWEI_HORZLINEXCOORD}
TWEI_HORZLINEYCOORD = $120B;
{$EXTERNALSYM TWEI_HORZLINEYCOORD}
TWEI_HORZLINELENGTH = $120C;
{$EXTERNALSYM TWEI_HORZLINELENGTH}
TWEI_HORZLINETHICKNESS = $120D;
{$EXTERNALSYM TWEI_HORZLINETHICKNESS}
TWEI_VERTLINEXCOORD = $120E;
{$EXTERNALSYM TWEI_VERTLINEXCOORD}
TWEI_VERTLINEYCOORD = $120F;
{$EXTERNALSYM TWEI_VERTLINEYCOORD}
TWEI_VERTLINELENGTH = $1210;
{$EXTERNALSYM TWEI_VERTLINELENGTH}
TWEI_VERTLINETHICKNESS = $1211;
{$EXTERNALSYM TWEI_VERTLINETHICKNESS}
TWEI_PATCHCODE = $1212;
{$EXTERNALSYM TWEI_PATCHCODE}
TWEI_ENDORSEDTEXT = $1213;
{$EXTERNALSYM TWEI_ENDORSEDTEXT}
TWEI_FORMCONFIDENCE = $1214;
{$EXTERNALSYM TWEI_FORMCONFIDENCE}
TWEI_FORMTEMPLATEMATCH = $1215;
{$EXTERNALSYM TWEI_FORMTEMPLATEMATCH}
TWEI_FORMTEMPLATEPAGEMATCH = $1216;
{$EXTERNALSYM TWEI_FORMTEMPLATEPAGEMATCH}
TWEI_FORMHORZDOCOFFSET = $1217;
{$EXTERNALSYM TWEI_FORMHORZDOCOFFSET}
TWEI_FORMVERTDOCOFFSET = $1218;
{$EXTERNALSYM TWEI_FORMVERTDOCOFFSET}
TWEI_BARCODECOUNT = $1219;
{$EXTERNALSYM TWEI_BARCODECOUNT}
TWEI_BARCODECONFIDENCE = $121A;
{$EXTERNALSYM TWEI_BARCODECONFIDENCE}
TWEI_BARCODEROTATION = $121B;
{$EXTERNALSYM TWEI_BARCODEROTATION}
TWEI_BARCODETEXTLENGTH = $121C;
{$EXTERNALSYM TWEI_BARCODETEXTLENGTH}
TWEI_DESHADECOUNT = $121D;
{$EXTERNALSYM TWEI_DESHADECOUNT}
TWEI_DESHADEBLACKCOUNTOLD = $121E;
{$EXTERNALSYM TWEI_DESHADEBLACKCOUNTOLD}
TWEI_DESHADEBLACKCOUNTNEW = $121F;
{$EXTERNALSYM TWEI_DESHADEBLACKCOUNTNEW}
TWEI_DESHADEBLACKRLMIN = $1220;
{$EXTERNALSYM TWEI_DESHADEBLACKRLMIN}

```

```

TWEI_DESHADEBLACKRLMAX  = $1221;
{$EXTERNALSYM TWEI_DESHADEBLACKRLMAX}
TWEI_DESHADEWHITECOUNTOLD = $1222;
{$EXTERNALSYM TWEI_DESHADEWHITECOUNTOLD}
TWEI_DESHADEWHITECOUNTNEW = $1223;
{$EXTERNALSYM TWEI_DESHADEWHITECOUNTNEW}
TWEI_DESHADEWHITERLMIN  = $1224;
{$EXTERNALSYM TWEI_DESHADEWHITERLMIN}
TWEI_DESHADEWHITERLAVE  = $1225;
{$EXTERNALSYM TWEI_DESHADEWHITERLAVE}
TWEI_DESHADEWHITERLMAX  = $1226;
{$EXTERNALSYM TWEI_DESHADEWHITERLMAX}
TWEI_BLACKSPECKLESREMOVED = $1227;
{$EXTERNALSYM TWEI_BLACKSPECKLESREMOVED}
TWEI WhitespecklesRemoved = $1228;
{$EXTERNALSYM TWEI WhitespecklesRemoved}
TWEI_HORZLINECOUNT  = $1229;
{$EXTERNALSYM TWEI_HORZLINECOUNT}
TWEI_VERTLINECOUNT  = $122A;
{$EXTERNALSYM TWEI_VERTLINECOUNT}
TWEI_DESKEWSTATUS  = $122B;
{$EXTERNALSYM TWEI_DESKEWSTATUS}
TWEI_SKEWORIGINALANGLE = $122C;
{$EXTERNALSYM TWEI_SKEWORIGINALANGLE}
TWEI_SKEWFINALANGLE  = $122D;
{$EXTERNALSYM TWEI_SKEWFINALANGLE}
TWEI_SKEWCONFIDENCE  = $122E;
{$EXTERNALSYM TWEI_SKEWCONFIDENCE}
TWEI_SKEWWINDOWX1  = $122F;
{$EXTERNALSYM TWEI_SKEWWINDOWX1}
TWEI_SKEWWINDOWY1  = $1230;
{$EXTERNALSYM TWEI_SKEWWINDOWY1}
TWEI_SKEWWINDOWX2  = $1231;
{$EXTERNALSYM TWEI_SKEWWINDOWX2}
TWEI_SKEWWINDOWY2  = $1232;
{$EXTERNALSYM TWEI_SKEWWINDOWY2}
TWEI_SKEWWINDOWX3  = $1233;
{$EXTERNALSYM TWEI_SKEWWINDOWX3}
TWEI_SKEWWINDOWY3  = $1234;
{$EXTERNALSYM TWEI_SKEWWINDOWY3}
TWEI_SKEWWINDOWX4  = $1235;
{$EXTERNALSYM TWEI_SKEWWINDOWX4}
TWEI_SKEWWINDOWY4  = $1236;
{$EXTERNALSYM TWEI_SKEWWINDOWY4}
TWEI_BOOKNAME      = $1238; { added 1.9 }
{$EXTERNALSYM TWEI_BOOKNAME}
TWEI_CHAPTERNUMBER  = $1239; { added 1.9 }
{$EXTERNALSYM TWEI_CHAPTERNUMBER}

```

```

TWEI_DOCUMENTNUMBER = $123A; { added 1.9 }
{$EXTERNALSYM TWEI_DOCUMENTNUMBER}
TWEI_PAGENUMBER     = $123B; { added 1.9 }
{$EXTERNALSYM TWEI_PAGENUMBER}
TWEI_CAMERA         = $123C; { added 1.9 }
{$EXTERNALSYM TWEI_CAMERA}
TWEI_FRAMENUMBER    = $123D; { added 1.9 }
{$EXTERNALSYM TWEI_FRAMENUMBER}
TWEI_FRAME          = $123E; { added 1.9 }
{$EXTERNALSYM TWEI_FRAME}
TWEI_PIXELFLAVOR    = $123F; { added 1.9 }
{$EXTERNALSYM TWEI_PIXELFLAVOR}

```

```

TWEJ_NONE           = $0000;
{$EXTERNALSYM TWEJ_NONE}
TWEJ_MIDSEPARATOR   = $0001;
{$EXTERNALSYM TWEJ_MIDSEPARATOR}
TWEJ_PATCH1         = $0002;
{$EXTERNALSYM TWEJ_PATCH1}
TWEJ_PATCH2         = $0003;
{$EXTERNALSYM TWEJ_PATCH2}
TWEJ_PATCH3         = $0004;
{$EXTERNALSYM TWEJ_PATCH3}
TWEJ_PATCH4         = $0005;
{$EXTERNALSYM TWEJ_PATCH4}
TWEJ_PATCH6         = $0006;
{$EXTERNALSYM TWEJ_PATCH6}
TWEJ_PATCHT         = $0007;
{$EXTERNALSYM TWEJ_PATCHT}

```

```

{ Added 1.8 }
{ TW_PASSTHRU.Direction values }
TWDR_GET            = 1;
{$EXTERNALSYM TWDR_GET}
TWDR_SET            = 2;
{$EXTERNALSYM TWDR_SET}

```

```

{*****
*      Return Codes and Condition Codes section      *
*****}

```

```

{ Return Codes: DSM_Entry and DS_Entry may return any one of these values. }
TWRC_CUSTOMBASE    = $8000;
{$EXTERNALSYM TWRC_CUSTOMBASE}

```

```

TWRC_SUCCESS        = 0;
{$EXTERNALSYM TWRC_SUCCESS}
TWRC_FAILURE        = 1; { Application may get TW_STATUS for info on failure }

```

```

{$EXTERNALSYM TWRC_FAILURE}
TWRC_CHECKSTATUS = 2; { "tried hard": ; get status }
{$EXTERNALSYM TWRC_CHECKSTATUS}
TWRC_CANCEL      = 3;
{$EXTERNALSYM TWRC_CANCEL}
TWRC_DSEVENT     = 4;
{$EXTERNALSYM TWRC_DSEVENT}
TWRC_NOTDSEVENT  = 5;
{$EXTERNALSYM TWRC_NOTDSEVENT}
TWRC_XFERDONE    = 6;
{$EXTERNALSYM TWRC_XFERDONE}
TWRC_ENDOFLIST   = 7; { After MSG_GETNEXT if nothing left }
{$EXTERNALSYM TWRC_ENDOFLIST}
TWRC_INFONOTSUPPORTED = 8;
{$EXTERNALSYM TWRC_INFONOTSUPPORTED}
TWRC_DATANOTAVAILABLE = 9;
{$EXTERNALSYM TWRC_DATANOTAVAILABLE}

```

{ Condition Codes: Application gets these by doing DG_CONTROL DAT_STATUS MSG_GET. }

```

TWCC_CUSTOMBASE = $8000;
{$EXTERNALSYM TWCC_CUSTOMBASE}

```

```

TWCC_SUCCESS      = 0; { It worked! }
{$EXTERNALSYM TWCC_SUCCESS}
TWCC BUMMER       = 1; { Failure due to unknown causes }
{$EXTERNALSYM TWCC BUMMER}
TWCC_LOWMEMORY    = 2; { Not enough memory to perform operation }
{$EXTERNALSYM TWCC_LOWMEMORY}
TWCC_NODS         = 3; { No Data Source }
{$EXTERNALSYM TWCC_NODS}
TWCC_MAXCONNECTIONS = 4; { DS is connected to max possible applications }
{$EXTERNALSYM TWCC_MAXCONNECTIONS}
TWCC_OPERATIONERROR = 5; { DS or DSM reported error, application shouldn't }
{$EXTERNALSYM TWCC_OPERATIONERROR}
TWCC_BADCAP       = 6; { Unknown capability }
{$EXTERNALSYM TWCC_BADCAP}
TWCC_BADPROTOCOL  = 9; { Unrecognized MSG DG DAT combination }
{$EXTERNALSYM TWCC_BADPROTOCOL}
TWCC_BADVALUE     = 10; { Data parameter out of range }
{$EXTERNALSYM TWCC_BADVALUE}
TWCC_SEQERROR     = 11; { DG DAT MSG out of expected sequence }
{$EXTERNALSYM TWCC_SEQERROR}
TWCC_BADDEST      = 12; { Unknown destination Application/Source in
DSM_Entry }
{$EXTERNALSYM TWCC_BADDEST}
TWCC_CAPUNSUPPORTED = 13; { Capability not supported by source }
{$EXTERNALSYM TWCC_CAPUNSUPPORTED}

```

```

TWCC_CAPBADOPERATION = 14; { Operation not supported by capability }
{$EXTERNALSYM TWCC_CAPBADOPERATION}
TWCC_CAPSEQERROR = 15; { Capability has dependancy on other capability }
{$EXTERNALSYM TWCC_CAPSEQERROR}

```

```
{ Added 1.8 }
```

```

TWCC_DENIED = 16; { File System operation is denied (file is protected) }
{$EXTERNALSYM TWCC_DENIED}
TWCC_FILEEXISTS = 17; { Operation failed because file already exists. }
{$EXTERNALSYM TWCC_FILEEXISTS}
TWCC_FILENOTFOUND = 18; { File not found }
{$EXTERNALSYM TWCC_FILENOTFOUND}
TWCC_NOTEMPTY = 19; { Operation failed because directory is not empty }
{$EXTERNALSYM TWCC_NOTEMPTY}
TWCC_PAPERJAM = 20; { The feeder is jammed }
{$EXTERNALSYM TWCC_PAPERJAM}
TWCC_PAPERDOUBLEFEED = 21; { The feeder detected multiple pages }
{$EXTERNALSYM TWCC_PAPERDOUBLEFEED}
TWCC_FILEWRITEERROR = 22; { Error writing the file (meant for things like disk full
conditions) }
{$EXTERNALSYM TWCC_FILEWRITEERROR}
TWCC_CHECKDEVICEONLINE = 23; { The device went offline prior to or during this
operation }
{$EXTERNALSYM TWCC_CHECKDEVICEONLINE}

```

```
{ bit patterns: for query the operation that are supported by the data source on a
capability }
```

```
{ Application gets these through
```

```
DG_CONTROL/DAT_CAPABILITY/MSG_QUERY SUPPORT }
```

```
{ Added 1.6 }
```

```

TWQC_GET = $0001;
{$EXTERNALSYM TWQC_GET}
TWQC_SET = $0002;
{$EXTERNALSYM TWQC_SET}
TWQC_GETDEFAULT = $0004;
{$EXTERNALSYM TWQC_GETDEFAULT}
TWQC_GETCURRENT = $0008;
{$EXTERNALSYM TWQC_GETCURRENT}
TWQC_RESET = $0010;
{$EXTERNALSYM TWQC_RESET}

```

```
{*****
```

```
* Entry Points *
```

```
***** }
```

```
{*****
```

```
* Function: DSM_Entry, the only entry point into the Data Source Manager.
```

```

*
* Parameters:
* pOrigin Identifies the source module of the message. This could
*   identify an Application, a Source, or the Source Manager.
*
* pDest Identifies the destination module for the message.
*   This could identify an application or a data source.
*   If this is NULL, the message goes to the Source Manager.
*
* DG The Data Group.
*   Example: DG_IMAGE.
*
* DAT The Data Attribute Type.
*   Example: DAT_IMAGEMEMXFER.
*
* MSG The message. Messages are interpreted by the destination module
*   with respect to the Data Group and the Data Attribute Type.
*   Example: MSG_GET.
*
* pData A pointer to the data structure or variable identified
*   by the Data Attribute Type.
*   Example: (TW_MEMREF)&ImageMemXfer
*   where ImageMemXfer is a TW_IMAGEMEMXFER structure.
*
* Returns:
* ReturnCode
*   Example: TWRC_SUCCESS.
*
***** }
type
{$EXTERNALSYM DSMENTRYPROC}
DSMENTRYPROC = function(pOrigin: pTW_IDENTITY; pDest: pTW_IDENTITY;
  DG: TW_UINT32; DAT: TW_UINT16; MSG: TW_UINT16;
  pData: TW_MEMREF): TW_UINT16; stdcall;

TDSMEntryProc = DSMENTRYPROC;

var
  DSM_Entry: TDSMEntryProc = nil;

{*****
* Function: DS_Entry, the entry point provided by a Data Source.
*
* Parameters:
* pOrigin Identifies the source module of the message. This could
*   identify an application or the Data Source Manager.
*
* DG The Data Group.

```

```

*   Example: DG_IMAGE.
*
* DAT   The Data Attribute Type.
*   Example: DAT_IMAGEMEMXFER.
*
* MSG   The message. Messages are interpreted by the data source
*   with respect to the Data Group and the Data Attribute Type.
*   Example: MSG_GET.
*
* pData A pointer to the data structure or variable identified
*   by the Data Attribute Type.
*   Example: (TW_MEMREF)&ImageMemXfer
*   where ImageMemXfer is a TW_IMAGEMEMXFER structure.
*
* Returns:
* ReturnCode
*   Example: TWRC_SUCCESS.
*
* Note:
* The DSPROC type is only used by an application when it calls
* a Data Source directly, bypassing the Data Source Manager.
*
***** }
type
  DSEENTRYPROC = function(pOrigin: pTW_IDENTITY; DG: TW_UINT32; DAT:
TW_UINT16;
    MSG: TW_UINT16; pData: TW_MEMREF): TW_UINT16; stdcall;
  {$EXTERNALSYM DSEENTRYPROC}

  TDSEEntryProc = DSEENTRYPROC;

var
  DS_Entry: TDSEEntryProc = nil;

implementation

end.

```