MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INPE - INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INICIAÇÃO CIENTÍFICA (CNPq)
RELATÓRIO FINAL

SOFTWARE DE REDUÇÃO DE DADOS TELEMÉTRICOS DO PROJETO MASCO

Orientador: Thyrso Villela

Elaborado por: Erika Trench Sestari

INPE

São José dos Campos
Junho de 1997

Sumário

1 - Introdução:

As observações em alta energia (raios-X e raios-gama) são uma ferramenta de uso importante para entender os processos físicos que ocorrem no Universo.

Atualmente, no Brasil, o único grupo que desenvolve pesquisa experimental na área de astronomia de raios-X duros e raios-gama de baixa energia é o grupo de astrofísica de altas energias da Divisão de Astrofísica do INPE. Tendo em vista a importância das observações em alta energia, essa Divisão decidiu construir um telescópio imageador de raios-X e raios-gama utilizando a técnica de imageamento conhecida como máscara Codificada, daí o nome preliminar do projeto: MASCO, que tem como objetivo principal obter imagens do céu em altas energias.

Este projeto, além do objetivo principal, proporcionará ao INPE, e por conseguinte à comunidade astronômica nacional, a capacidade de dispor da tecnologia para construir plataformas capazes de levar instrumentos astronômicos a grandes altitudes e efetuar observações praticamente livres da influência da atmosfera, já que em altas energias é fundamental que os instrumentos operem em grandes altitudes.

No caso de observações em alta energia, as observações precisam ser feitas a bordo de veículos espaciais (balões, foguetes e satélites) já que a atmosfera terrestre absorve essa radiação eletromagnética.

Através das imagens obtidas pelo telescópio MASCO será possível realizar o estudo de objetos astrofísicos pois as imagens possibilitarão distinguir individualmente os mesmos e obter seus espectros descontaminados da contribuição de outros objetos e do ruído de fundo no mesmo campo de visada.

Dentro do projeto MASCO existem várias possibilidades de trabalho sendo uma delas o desenvolvimento do software de redução de dados do mesmo, que será uma importante ferramenta para análise dos dados do experimento.

Após o vôo, através deste software será possível visualizar a evolução do balão (latitude, longitude e altitude em determinadas horas) além de colaborar na análise de grandezas tais como pressão e temperatura, o que facilitará a observação do comportamento do experimento e confecção de tabelas e gráficos.

O software de redução de dados telemétricos apresentado neste relatório lerá arquivos binários gerados pelo programa de aquisição da telemetria do experimento (masco.vi *LabView*) e irá convertê-los para um formato adequado à análise dos dados.

O programa *masco.vi* gera automaticamente uma seqüência de arquivos binários durante o processo de monitoração do Frame (quadro de dados) que é enviado pelos microcontroladores de

bordo, HO (Housekeeping) e TC (Telecomando).O conjunto de dados gravados nestes arquivos é uma cópia do Frame enviado pelo HO.

O software foi realizado em 4 etapas. Inicialmente, foi desenvolvido um programa em linguagem C que realiza as seguintes tarefas: pedir ao usuário o nome do arquivo binário de entrada a ser convertido, verificar a existência deste arquivo, abri-lo, converter os dados para a base decimal e gerar o arquivo de saída em modo texto ASCII, em um formato adequado para análise.

Na etapa seguinte foi feita a leitura da hora (formato: hh:mm:ss) e hora decimal, ambas baseadas no GPS, sendo verificados erros tais como frame com palavras a menos, arquivos com menor número de frames e arquivos inexistentes. Além disso foram calculadas as variáveis Altitude, Latitude e Longitude sendo essas variáveis introduzidas ao formato já existente, ficando este, alterado para o formato apresentado logo abaixo:

Formato do Arquivo de Saída:

HORA	HORAdecimal	No Frame	Latitude	Longitude	Altitude	Palavra 2	...	Palavra74
[hh:mm:ss]	[baseada no GPS]	[Palavra73]				[Palavra2]	...	[Palavra74]
.	.	.				.		.
.	.	.				.		.
.	.	.				.		.
[hh:mm:ss]	[baseada no GPS]	[Palavra73]				[Palavra2]	...	[Palvra74]

Em seguida, o programa foi adaptado para funcionar com um tipo diferente de entrada, com menor número de frames e palavras por arquivo. Nesse estágio, o programa sofreu grandes alterações para acomodar tipos diferentes de entrada, permitindo fácil adaptação para novos tipos.

A etapa final do desenvolvimento do software consistiu na elaboração da interface gráfica do programa, utilizando como recurso o compilador Visual C++ da Microsoft.

A utilização de um compilador versátil como o Visual C foi necessária não na parte de tratamento de dados do programa mas para o melhor controle da interface com o usuário. Ele facilitou a confecção de diálogos e janelas em ambiente Windows para uma mais fácil compreensão e utilização do programa, requisitando o mínimo de informações do seu usuário e respondendo de maneira simples e clara.

Com o software de redução de dados telemétricos do projeto MASCO, é possível visualizar a evolução do experimento com relação à latitude, longitude e altitude em determinadas horas. Ou seja, por meio deste software de redução de dados, será possível a construção de tabelas e gráficos relativos a essas variáveis . Além disso, o software também mostra as palavras relativas a pressão, temperatura, tensões elétricas (tensão das baterias), codificadores de posição e estados digitais convertidas para decimal, o que facilita a análise de dados.

2 - Análise do Material Bibliográfico:

A presente análise será dividida em duas partes, uma referente ao Projeto MASCO e outra referente ao software de redução de dados telemétricos.

Recentemente, com a entrada em operação do Compton Gamma Ray Observatory (CGRO) e do GRANAT, as observações em altas energias foram finalmente incorporadas à astrofísica como uma ferramenta de uso indispensável para entender os processos físicos que ocorrem no Universo.

Uma das mais empolgantes áreas da astrofísica moderna é a da astronomia de raios-X duros e raios-gama de baixa energia. Isto se deve à descoberta de uma rica fenomenologia em uma grande variedade de objetos astrofísicos, cuja emissão de radiação eletromagnética está associada a processos não-térmicos altamente energéticos. O grande avanço tecnológico recente na área permitem o desenvolvimento de detectores mais sensíveis e técnicas telescópicas mais avançadas, que fazem com que este campo da astrofísica evolua em uma velocidade vertiginosa.

O campo da astrofísica de raios-X duros e raios-gama de baixa energia é extremamente fértil e rico, o que pode ser comprovado pela grande quantidade de descobertas recentes na área. Dentre os diversos objetos astrofísicos que deverão ser observados e estudados pelo projeto MASCO, detacam-se o Centro e Plano Galácticos e os núcleos ativos de galáxias.

Foi utilizado no projeto a técnica de imageamento conhecida como Máscara Codificada, já utilizada por aproximadamente trinta telescópios de raios-X, que empregaram os padrões de máscara mais diversos: Fresnel, "pinhole", URA e HURA.(Cook,W.R. *et al. 1984)*

O telescópio MASCO utilizará um padrão novo de máscara codificada: o MURA - Modified Uniformily Redundant Array - (Gottesman e Fenimore, 1989) e será o primeiro experimento a utilizar este padrão. Esta nova família da arranjos binários tem propriedades e relação sinal/ruído iguais às dos arranjos uniformemente redundantes. Tendo em vista sua sesibilidade e capacidade de imageamento, o Projeto MASCO será bastante competitivo em nível internacional.

Os MURA's são uma classe mais geral de padrões de máscara codificada da qual os URA's são um sub-conjunto. As dimensões (p,q) de um URA (Fenimore e Cannon, 1978) têm as seguintes propriedades: a) são números primos e b) obedecem uma relação do tipo $p-q = 2$. Assim, as dimensões básicas de um padrão URA são definidas por dois números primos consecutivos, permitindo somente a construção de uma forma retangular para os arranjos. Os MURA's, por outro lado, podem ser construídos de acordo com a relação $p-q = 0$, ou seja, é possível construir uma máscara de forma quadrada. Esta é uma característica excelente do padrão MURA, uma vez

que ela pode ser melhor adaptado à forma da superfície do detector, que em geral é circular. Na configuração quadrada, os MURA's podem ser construídos com qualquer dimensão L que seja um número primo, que obedeça a relação L = 4m + 1, m = 1,2,3,..., e que tenha um "throughput" (L-1)/2L, onde L = p^2 é o número de elementos do padrão básico, já que p = q. Com a introdução dos MURA's todos os números primos podem ser usados para gerar máscaras codificadas. A Figura 1 abaixo mostra o padrão MURA que é empregado no telescópio MASCO e a figura 2 mostra a Máscara Codificada.

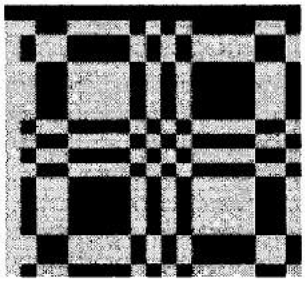


Figura 1:padrão MURA.

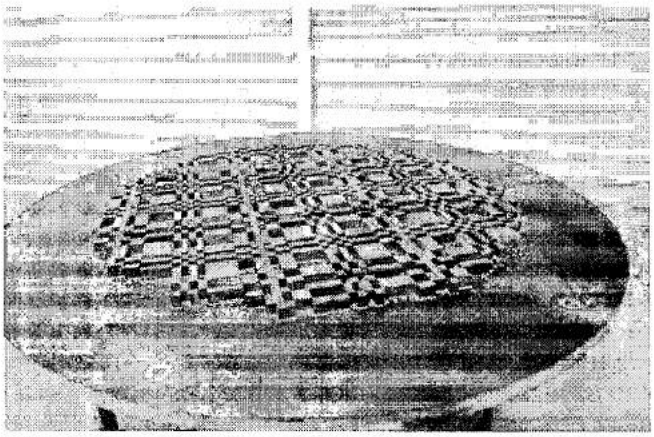


Figura 2: A Máscara Codificada.

A máscara e sua antimáscara são montadas numa mesma peça. A antimáscara é semelhante a máscara, a diferença está apenas nas posições dos espaços vazios e cheios, na antimáscara eles são trocados, de forma que se a máscara for colocada sobre a antimáscara os espaço vazios desta coincidem exatamente com os cheios da outra. Antes disso, em experimentos

do mesmo tipo, a máscara tinha que ficar em uma peça e a antimáscara em outra peça, dificultando a troca de uma pela outra. Nesse experimento basta virar a máscara de 90° que obtém-se a antimáscara. Além disso, utilizar a anti-máscara melhora a relação sinal/ruído. Esse método de utilizar máscara e anti-máscara em uma mesma peça já foi implementado e testado com sucesso no experimento TIMAX (Braga *et al.*,1991).

Inicialmente o experimento será lançado em uma região já conhecida do espaço, o que possibilitará verificar seu funcionamento, pois nesse caso já existe uma determinação das posições das fontes de alta energia que poderão ser comparadas com as posições determinadas pelo experimento.

O projeto MASCO, além de obter imagens do céu em alta energia, também tentará colaborar com a determinação da variação de fluxos de energia, até hoje não compreendida pelos astrofísicos.

Com relação ao softwarede redução de dados, durante o desenvolvimento do código do programa foi utilizado o livro de linguagem C denominado Linguagem C:Teoria & Programas (Mesquita, T.J.,1992) além do próprio help do borland C. Durante a implementação da interface do programa utilizou-se o manual do visual C++ fornecido pela Microsoft (1993) e o livro Learning C++ (Graham, N.,1996).

Tanto na implementação do código do programa quanto no desenvolvimento da interface, os livros citados serviram de apoio para um melhor entendimento da linguagem C e da linguagem C++, além de ensinar a trabalhar com o Visual C++.

3 - Procedimentos Metodológicos:

(Análise do desenvolvimento do software de redução de dados)

3.1 - Desenvolvimento:

Para facilitar o desenvolvimento do software de redução de dados telemétricos, sua elaboração foi dividida em 4 etapas ou módulos:

- **1ª Etapa:**

Nesta etapa do desenvolvimento do software foi dado início a um programa em linguagem C que deverá realizar as seguintes tarefas: pedir ao usuário o nome do arquivo binário de entrada a ser convertido (nome e formato apresentados no **Anexo 1 e Anexo 2**, respectivamente), abrir este arquivo, converter os dados para a base decimal e gerar o arquivo de saída em modo texto ASCII, em um formato adequado para análise, descrito logo abaixo:

Formato do Arquivo de Saída:

No Frame	Palavra 2	Palavra3	...	Palavra 74
[Palavra73]	[Palavra2]	[Palavra3]	...	[Palavra74]
.	.	.		.
.	.	.		.
.	.	.		.
[Palavra73]	[Palavra2]	[Palvra3]]	...	[Palavra74]

- Total de linhas em função do número de Frames convertidos.
- Todos os valores numéricos em decimal.
- GPS: sistema de localização baseado em satélites, "Global Pointing System".

Neste módulo do programa foi desenvolvida uma verificação para existência ou não do arquivo a ser lido, caso ele não exista é enviada uma mensagem de erro ao usuário. Casos deste tipo serão melhor analisados no desenvolvimento da etapa seguinte do programa.

- **2ª Etapa:**

Nesta etapa do desenvolvimento do programa foi feita a leitura da Hora (formato: hh:mm:ss) e da Hora decimal, ambas baseadas no GPSTIME . Para essa leitura foi seguido o Diagrama de Blocos apresentado no **Anexo 3** e **Anexo 4**.

Além disso foram verificados erros tais como frame com palavras a menos, arquivos com menos de 1000 frames e arquivos inexistentes. O formato do arquivo de saída foi ligeiramente alterado, incrementando as variáveis relacionadas a hora no formato descrito anteriormente.

Formato do Arquivo de Saída:

HORA	HORAdecimal	No Frame	Palavra 2	...	Palavra74
[hh:mm:ss]	[baseada no GPS]	[Palavra73]	[Palavra2]	...	[Palavra74]
.	.	.			.
.	.	.			.
.	.	.			.
[hh:mm:ss]	[baseada no GPS]	[Palavra73]	[Palavra2]	...	[Palvra74]

Além disso, nesta etapa do desenvolvimento do programa foram calculadas as variáveis Altitude, Latitude e Longitude.

Seguindo o Diagrama de Blocos mostrado no **Anexo 5** foi implementada a variável Altitude.

Através dos Diagramas de Blocos mostrados nos **Anexo 3 e Anexo 6** pode-se implementar a Latitude; No **Anexo 3 e Anexo 7** são apresentados os Diagramas de Blocos que serão utilizados no cálculo efetivo da Longitude.

Essas três variáveis calculadas foram introduzidas ao formato já existente, ficando este alterado para o formato apresentado logo abaixo:

Formato do Arquivo de Saída:

HORA	HORAdecimal	No Frame	Latitude	Longitude	Altitude	Palavra 2	...	Palavra74
[hh:mm:ss]	[baseada no GPS]	[Palavra73]				[Palavra2]	...	[Palavra74]
.	.	.			.	.		.
.	.	.				.		.
.	.	.				.		.
[hh:mm:ss]	[baseada no GPS]	[Palavra73]				[Palavra2]	...	[Palvra74]

- **3ª Etapa:**
-

Em seguida, o programa foi expandido para aceitar outro formato além do descrito no **anexo 2** (novo formato descrito no **anexo 8**). Nesse novo formato o número de frames diminuiu para 500, o tamanho dos frames diminuiu e alguns dados foram movidos de posição.

Nessa etapa foi feita uma reformulação no tratamento de dados de modo a tornar seletivo o tipo de arquivo lido. O programa reconhece o tipo e lê os dados segundo o seu formato. Isso torna fácil possíveis expansões futuras, para a utilização de novos tipos de arquivos.

- **4ª Etapa:**

Finalmente, após o término da parte lógica do programa, iniciamos o desenvolvimento da interface gráfica com o usuário baseado no ambiente Windows.

Foi criado um sistema de menu e diálogos de modo a automatizar tarefas como a conversão de múltiplos arquivos. Para isso foram criadas classes gráficas derivadas das classes padrão de Windows (Microsoft Fondation Classes) como classes de janela, menú e diálogo. Através de conceitos como herança e encapsulamento de classes, foi montada uma estrutura de classes gráficas, para o detalhamento da interface, e lógica, para a conversão do arquivo. Nessa última estrutura foram inseridas as funções originais do programa, desenvolvidas nas etapas anteriores.

3.2 - Descrição Técnica Final:

Para uma melhor compreensão do código de programa do software de redução de dados telemétricos que segue em anexo, logo abaixo será mostrado um resumo explicativo das principais funções globais utilizadas.

- Função : leitura

A função de leitura faz a abertura do arquivo em binário e lê os frames contidos em cada um desses arquivos. Nela são feitas verificações do tipo arquivo inexistente, frame com palavras a menos e arquivos com falta de frames, sendo que para cada um desses eventos é retornado um valor.

- Função : gravação

A função de gravação cria no arquivo de saída o cabeçário que segue o formato mostrado na 2ª etapa e grava neste cabeçário os dados obtidos.

- Função : Printtime

Seguindo os Diagramas de Blocos mostrados no **Anexo 3** e no **Anexo 4** é feito o cálculo da Hora do Frame em formato decimal e hh:mm:ss.

- Função : Latitude

Seguindo os Diagramas de Blocos mostrados no **Anexo 3** e no **Anexo 6** é feito o cálculo da Latitude do Balão.

- Função : Longitude

Seguindo os Diagramas de Blocos mostrados no **Anexo 3** e no **Anexo 7** é feito o cálculo da Longitude do Balão.

- Função : Altitude

Seguindo o Diagrama de Bloco mostrado no **Anexo 5** é feito o cálculo da Altitude do Balão.

- Função : Conv

No Quadro de Dados HouseKeeping apresentado em anexo estão especificadas 74 palavras, porém, nas palavras 50 até 74 é feita uma redistribuição de bytes, explicada no **Anexo 2**. Essas palavras, para poderem ser utilizadas corretamente, são convertidas de longword para word no vetor de dados, sendo este o objetivo da função Conv. Para formatos diferentes deste, deve-se especificar através da variável *NumWords* o número de dados que deve ser tratado como word no frame. No **Anexo 9** temos a descrição de outro tipo de entrada, onde o número de words no frame é 26.

- Função : FileSize

Essa função reconhece o tipo de arquivo de dados selecionado e altera as devidas variáveis para valores de modo que o programa possa reconhecer adequadamente o formato, ela funciona como uma tabela de tipos. Em caso de inserção de novos tipos de arquivos a única função que deve ser alterada será essa.

- Função : TrataErro

Função responsável pelo tratamento de um eventual erro no acesso a arquivos. Essa função associa uma mensagem a ser mostrada ao usuário em forma de diálogo.

À seguir foi feita uma descrição das classes utilizadas:

- Classe : CMascoApp

Essa classe é derivada da classe aplicativo de Windows CWinApp, e à partir dela são criados todos os outros objetos no programa.

Por meio da ferramenta App Studio do Visual C++ criamos classes derivadas dessa como a classe CMainframe, responsável pela janela principal da aplicação, CMascoDoc e CMascoView, também utilizadas para gerar entidades gráficas. Os recursos associados a essas classes foram utilizados modestamente, de modo que não nos ateremos a uma descrição detalhada dessas classes, além de se tratarem de classes comumente utilizadas e cujo código foi criado pela ferramenta.

Além das funções e variáveis associadas à classe original CWinApp, foram adicionadas as seguintes funções:

InitInstance: Inicialização padrão de aplicativo. Cria janela, menú e botões.

OnHelp: Visualiza mensagem de help.

OnAppAbout: Visualiza diálogo CAppAbout.

OnFileOpen: Trata evento de abertura de arquivos, realizando as funções de leitura e gravação.

OnFileSave: Realiza somente gravação com nome novo dado pelo usuário.

A função OnAppAbout cria um objeto do tipo CDialog simples, e o executa modalmente.

À partir da função OnFileOpen é criado um novo objeto da classe CCancelDlg, derivada da classe CDialog, que possui as funções adicionais:

Converte: Abre diálogo de informação e executa funções de leitura e gravação para os arquivos selecionados.

OnCancel: Trata evento Cancel, caso o usuário deseje interromper a operação.

Além dessas, são utilizadas algumas classes comuns de C++, como a classe CString, que implementa o tipo string em C++ e a classe CFileDialog, utilizada para "perguntar" ao usuário os nomes de arquivos a serem convertidos e gravados.

O código de programa apresentado em anexo está devidamente documentado.

4 - Resultados:

O programa foi testado recentemente com arquivos obtidos de outro experimento desenvolvido nos mesmos moldes do projeto MASCO. Neste caso, o arquivo binário tem seu formato descrito no anexo 8 deste relatório, acompanhando o respectivo quadro de dados, e não foi detectado nenhum erro.

O software ainda não foi implantado como ferramenta diária na conversão dos dados. Entretanto todos os testes foram realizados com arquivos reais, recebidos do programa masco.vi, e não ocorreram erros em nenhum desses testes, utilizando os dois formatos de arquivo descritos.

Somente através da utilização diária e constante do programa é que serão encontrados possíveis pontos deficientes, surgindo então idéias de ampliação e melhoria. Nesse ponto o usuário será beneficiado pela grande facilidade de alteração e expansão do projeto.

5 - Conclusão:

Com o que foi desenvolvido do software de redução de dados telemétricos do projeto MASCO, já é possível visualizar a evolução do experimento com relação a latitude, longitude e altitude em determinadas horas, ou seja, por meio desse software de redução de dados, será possível a construção de tabelas e gráficos relativos a essas variáveis . Além disso, o software também mostra as palavras relativas a pressão, temperatura, tensões elétricas (tensão das baterias), codificadores de posição e estados digitais convertidas para decimal, o que facilita análise do experimento.

A decisão de utilização do compilador C++ foi tomada pela facilidade de implementação de entidades gráficas oferecidas por ele. Todo o programa foi feita em linguagem C padrão, e a essa lógica foram acrecidas as rotinas de interface gráfica, algumas criadas pelo próprio compilador. Essas rotinas são definidas pelo seu fabricante (Microsoft) através de classes (MFC), por isso deve-se adotar um compilador C++, já que o compilador C padrão não aceita essa estrutura.

Essa interface visou a facilidade de manuseio do programa e expansão de seus recursos, de modo que, observando o resultado final, consideramos cumprido esse objetivo.

6 - Referências Bibliográficas:

- Villela, T. *et al.*:1994, *Astrophysics and Space Science*, vol. **5**, 269 - 278.

- Braga, J. *et al.*: 1991, *Experimental Astronomy* **2**,101.

- Fenimore, E.E. and Cannon, T.M.:1978, *Applied Optics* **17** (3),337.

- Gottesman, S.R. and Fenimore, E.E.:1989, *Applied Optics* **28**(20),4344.

- Mesquita, T. J. M. , Linguagem C : Teoria & Programas, Ed. Érica.

- Manual do Visual C++ da Microsoft.

- Graham, N.,Learning C++, Ed. McGraw Hill.

- Cook, W.R. *et al.*: 1984, *IEEE Trans. Nucl. Sci.* **31**,129.

7 - Apêndices:

Quadro de Dados (FRAME) HOUSEKEEPING MASCO

- Este FRAME serial é enviado pelo microcomputador de Bordo HOUSE, continuamente sendo recebido pela porta serial do PC.

Principais Características:

- Palavras de 16 Bits ou 2 Bytes.
- FRAME Assíncrono com Baude Rate 9600 bps, com 1 Start Bit, 8 Data Bits, 1 Stop, Parity None.
- As Palavras do grupo analógico de apresentação são provenientes do ADC de 12 Bits do Micro HOUSE, assim o valor da palavra esta entre 0 e 4095 (12 Bits).
- Os números entre parenteses no campo do 1o e 2o byte são os índices das matrizes de dados utilizadas no LABVIEW.

Formato do Quadro de Dados:

Palavra:	Ref. Esquemática:	1o. Byte (No Byte)	2o. Byte (No Byte)	Observação:	Grupo de Apresentação:
1	SYNC	EB	90	Sincronismo	Controle Interno
2	STHO1 (IN_ANA13)	LSB (0)	MSB		Analógico
3	STHO2 (IN_ANA14)	LSB (1)	MSB		Analógico
4	STHO3 (IN_ANA15)	LSB (2)	MSB		Analógico
5	STHO4 (IN_ANA16)	LSB (3)	MSB		Analógico
6	STHO5 (IN_ANA17)	LSB (4)	MSB		Analógico
7	STHO6 (IN_ANA18)	LSB (5)	MSB		Analógico
8	STHO7 (IN_ANA19)	LSB (6)	MSB		Analógico
9	STHO8 (IN_ANA20)	LSB (7)	MSB		Analógico
10	STHO9 (IN_ANA21)	LSB (8)	MSB		Analógico
11	STHO10 (IN_ANA22)	LSB (9)	MSB		Analógico
12	STHO11 (IN_ANA23)	LSB (10)	MSB		Analógico
13	STHOINT (IN_ANA24)	LSB (11)	MSB	Sensor Temperatura Interno Micro HOUSE	Analógico
14	AMTHO1 (IN_ANA5)	LSB (12)	MSB		Analógico
15	AMTHO2 (IN_ANA6)	LSB (13)	MSB		Analógico
16	AMTHO3 (IN_ANA7)	LSB (14)	MSB		Analógico
17	AMTHO4 (IN_ANA8)	LSB (15)	MSB		Analógico
18	AMTHO5 (IN_ANA9)	LSB (16)	MSB		Analógico
19	AMTHO6 (IN_ANA10)	LSB (17)	MSB		Analógico
20	AMTHO7 (IN_ANA11)	LSB (18)	MSB		Analógico
21	AMTHO8 (IN_ANA12)	LSB (19)	MSB		Analógico
22	AMT+5V (IN_ANA1)	LSB (20)	MSB		Analógico
23	AMT+12V (IN_ANA2)	LSB (21)	MSB		Analógico
24	AMT-12V (IN_ANA3)	LSB (22)	MSB		Analógico
25	AMTAT (IN_ANA4)	LSB (23)	MSB		Analógico

26	STTC1 (IN_ANA13)	LSB (24)	MSB		Analógico
27	STTC2 (IN_ANA14)	LSB (25)	MSB		Analógico
28	STTC3 (IN_ANA15)	LSB (26)	MSB		Analógico
29	STTC4 (IN_ANA16)	LSB (27)	MSB		Analógico
30	STTC5 (IN_ANA17)	LSB (28)	MSB		Analógico
31	STTC6 (IN_ANA18)	LSB (29)	MSB		Analógico
32	STTC7 (IN_ANA19)	LSB (30)	MSB		Analógico
33	STTC8 (IN_ANA20)	LSB (31)	MSB		Analógico
34	STTC9 (IN_ANA21)	LSB (32)	MSB		Analógico
35	STTC10 (IN_ANA22)	LSB (33)	MSB		Analógico
36	STTC11 (IN_ANA23)	LSB (34)	MSB		Analógico
37	STTCINT (IN_ANA24)	LSB (35)	MSB	Sensor Temperatura Interno Micro TELEC	Analógico
38	AMTTC1 (IN_ANA1)	LSB (36)	MSB		Analógico
39	AMTTC2 (IN_ANA2)	LSB (37)	MSB		Analógico
40	AMTTC3 (IN_ANA3)	LSB (38)	MSB		Analógico
41	AMTTC4 (IN_ANA4)	LSB (39)	MSB		Analógico
42	AMTTC5 (IN_ANA5)	LSB (40)	MSB		Analógico
43	AMTTC6 (IN_ANA6)	LSB (41)	MSB		Analógico
44	AMTTC7 (IN_ANA7)	LSB (42)	MSB		Analógico
45	AMTTC8 (IN_ANA8)	LSB (43)	MSB		Analógico
46	AMTTC9 (IN_ANA9)	LSB (44)	MSB		Analógico
47	AMTTC10 (IN_ANA10)	LSB (45)	MSB		Analógico
48	AMTTC11 (IN_ANA11)	LSB (46)	MSB		Analógico
49	AMTTC12 (IN_ANA12)	LSB (47)	MSB		Analógico
50	STATUSTC1	Byte1 (0)	Byte2 (1)	Ver Referência 3	STATUS
51	STATUSTC2	Byte3 (2)	Byte4 (3)	Ver Referência 3	STATUS
52	INDIG9-24	Byte1 (4)	Byte2 (5)	Ver Referência 1	Estados (Digital)
53	INDIG25-30	Byte3 (6)	Byte4 (7)	Ver Referência 1	Estados (Digital)
54	GPSVELE1	Byte1 (8)	Byte2 (9)	Ver Referência 2	GPS
55	GPSVELE2	Byte1 (10)	Byte2 (11)	Ver Referência 2	GPS
56	GPSVELN1	Byte1 (12)	Byte2 (13)	Ver Referência 2	GPS
57	GPSVELN2	Byte1 (14)	Byte2 (15)	Ver Referência 2	GPS
58	GPSVELU1	Byte1 (16)	Byte2 (17)	Ver Referência 2	GPS
59	GPSVELU2	Byte1 (18)	Byte2 (19)	Ver Referência 2	GPS
60	GPSLAT1	Byte1 (20)	Byte2 (21)	Ver Referência 2	GPS
61	GPSLAT2	Byte1 (22)	Byte2 (23)	Ver Referência 2	GPS
62	GPSLAT3	Byte1 (24)	Byte2 (25)	Ver Referência 2	GPS
63	GPSLON1	Byte1 (26)	Byte2 (27)	Ver Referência 2	GPS
64	GPSLON2	Byte1 (28)	Byte2 (29)	Ver Referência 2	GPS
65	GPSLON3	Byte1 (30)	Byte2 (31)	Ver Referência 2	GPS
66	GPSALT1	Byte1 (32)	Byte2 (33)	Ver Referência 2	GPS
67	GPSALT2	Byte1 (34)	Byte2 (35)	Ver Referência 2	GPS
68	GPSSTATUS	Byte1 (36)	Byte2 (37)	Ver Referência 2	GPS
69	GPSTIME1	Byte1 (38)	Byte2 (39)	Ver Referência 2	GPS
70	GPSTIME2	Byte1 (40)	Byte2 (41)	Ver Referência 2	GPS
71	GPSTIME3	Byte1 (42)	Byte2 (43)	Ver Referência 2	GPS
72	STATUSGERAL	Byte1 (44)	Byte2 (45)	Ver Referência 5	STATUS
73	CONTADOR	LSB (46)	MSB (47)	Contador de Frames 16 Bits	STATUS
74	CRCFRAME	LSB (48)	MSB (49)	Ver Referência 4	Controle Interno

Anexo 1

Arquivos gerados pelo programa LabView durante recepção do Frame do Housekeeping.

Nomes dos Arquivos:

Os nomes destes arquivos são formatados da seguinte maneira:

1o. Caractere: 'D' Arquivo de dados contendo todo o Frame (Binário)

2o. Caractere: '1 a 9' Representa o número do vôo
'T' Utilizado durante os testes
Obs: Este caractere pode ser alterado pelo usuário no programa de aquisição (LabView)

3o. e 4o. Caractere: '00 a 23' Hora do sistema (microcomputador de aquisição)

5o. a 8o. Caractere: '0000 a 9999' Minuto do sistema em formato decimal multiplicado por 10000

Obs: para calcular o minuto e o segundo da informação da hora deve-se seguir o seguinte procedimento:

1. Converter os caracteres 5o. a 8o. para um numero real (H)
2. $\left(\frac{H}{10000}\cdot 60\right)\text{int} = minuto$
3. $\left(\left(\left(\frac{H}{10000}\cdot 60\right) - minuto\right)\cdot 60\right)\text{int} = segundo$

Exemplo: O 5o. a 8o. dígitos são 5360, minuto = 0.5360x60 = 32.16 = 32 já o segundo = 0.16x60 = 9.6 = 9

Anexo 2

Formato do arquivo binário:

Os delimitadores '[ ]' identificam palavras de 16 bits, os números contidos dentro deste delimitadores fazem referência ao campo *Palavra* do documento **'Quadro de Dados (Frame) HOUSEKEEPING MASCO'.** As palavras 50 a 74 tem seus bytes redistribuídos em palavras de 16 bits, isto é, o primeiro e o segundo byte da uma determinada palavra foram colocados em uma nova palavra de 16 bits como sendo o segundo byte, sendo o primeiro 00. No esquema abaixo os delimitadores '{ }' são usados para delimitar bytes.

[73] [2] [3] ....[49] [50a] [50b] .... [74a] [74b] [{EB}{90}]

{00}{Byte1}

[73] [2] [3] ....[49] [50a] [50b] .... [74a] [74b] [{EB}{90}]

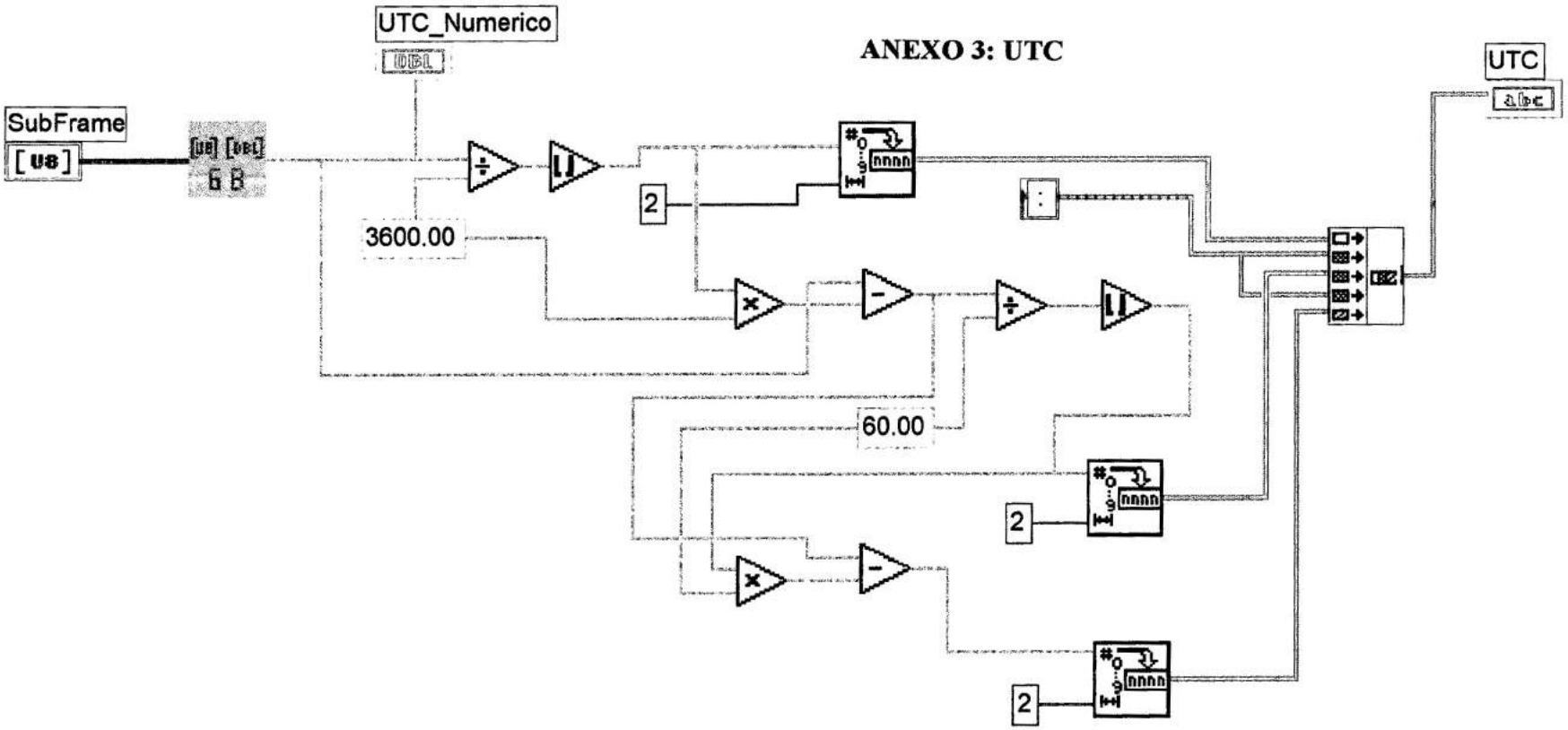
{00}{Byte2}

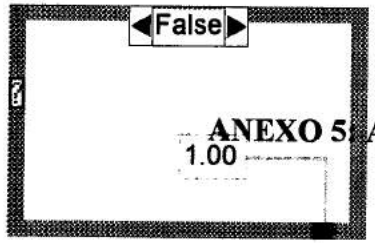
- Considerando que cada Frame no arquivo binário ocupa 200 bytes temos um total de 200 kbytes por arquivo.

alcula_UTC_GPS.vi
6/09/97 10:47 AM

lock Diagram

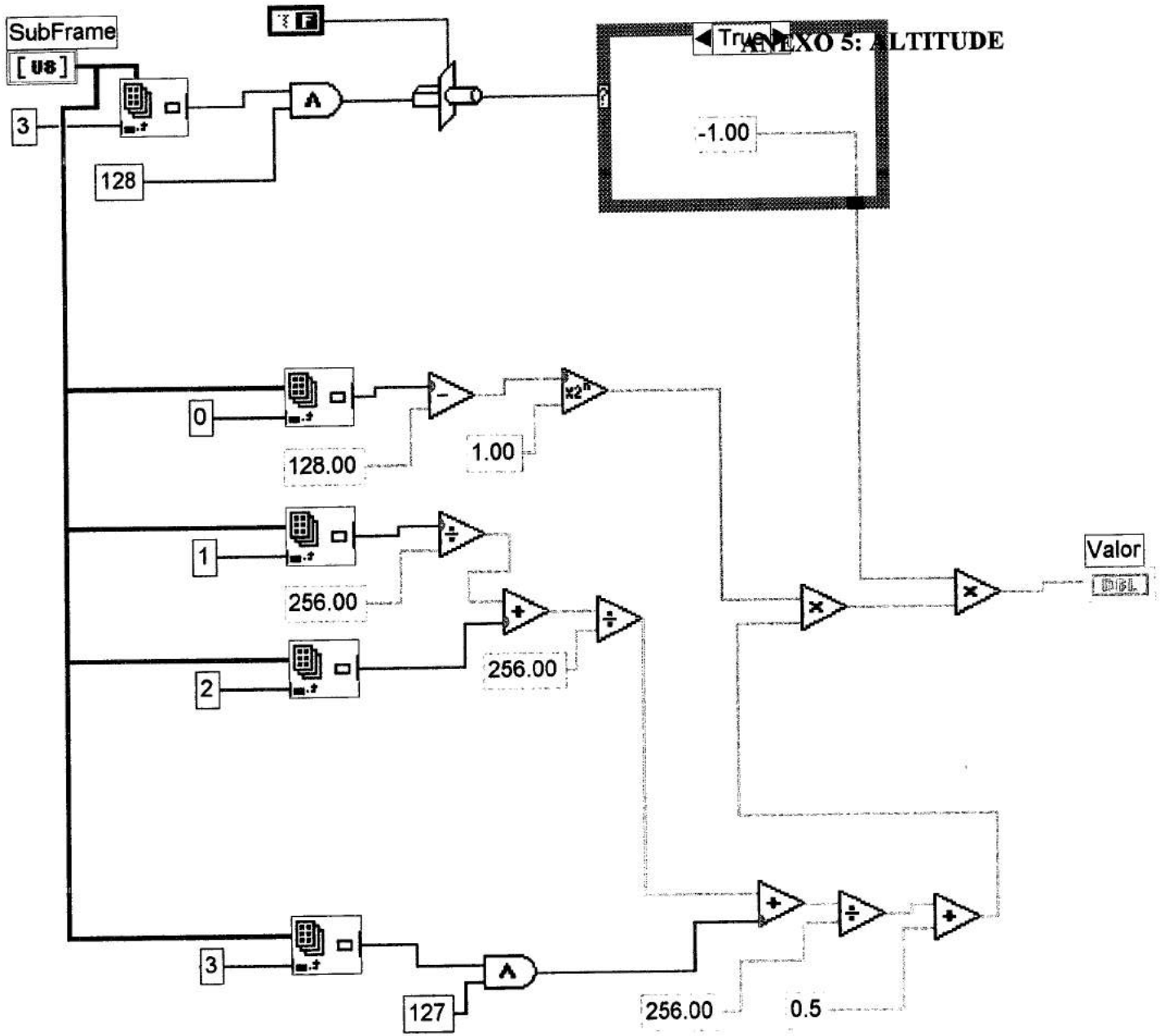
ANEXO 3: UTC




False
ANEXO 5 ALTITUDE
1.00

onverte_GPS_4.vi
6/09/97 10:51 AM

lock Diagram



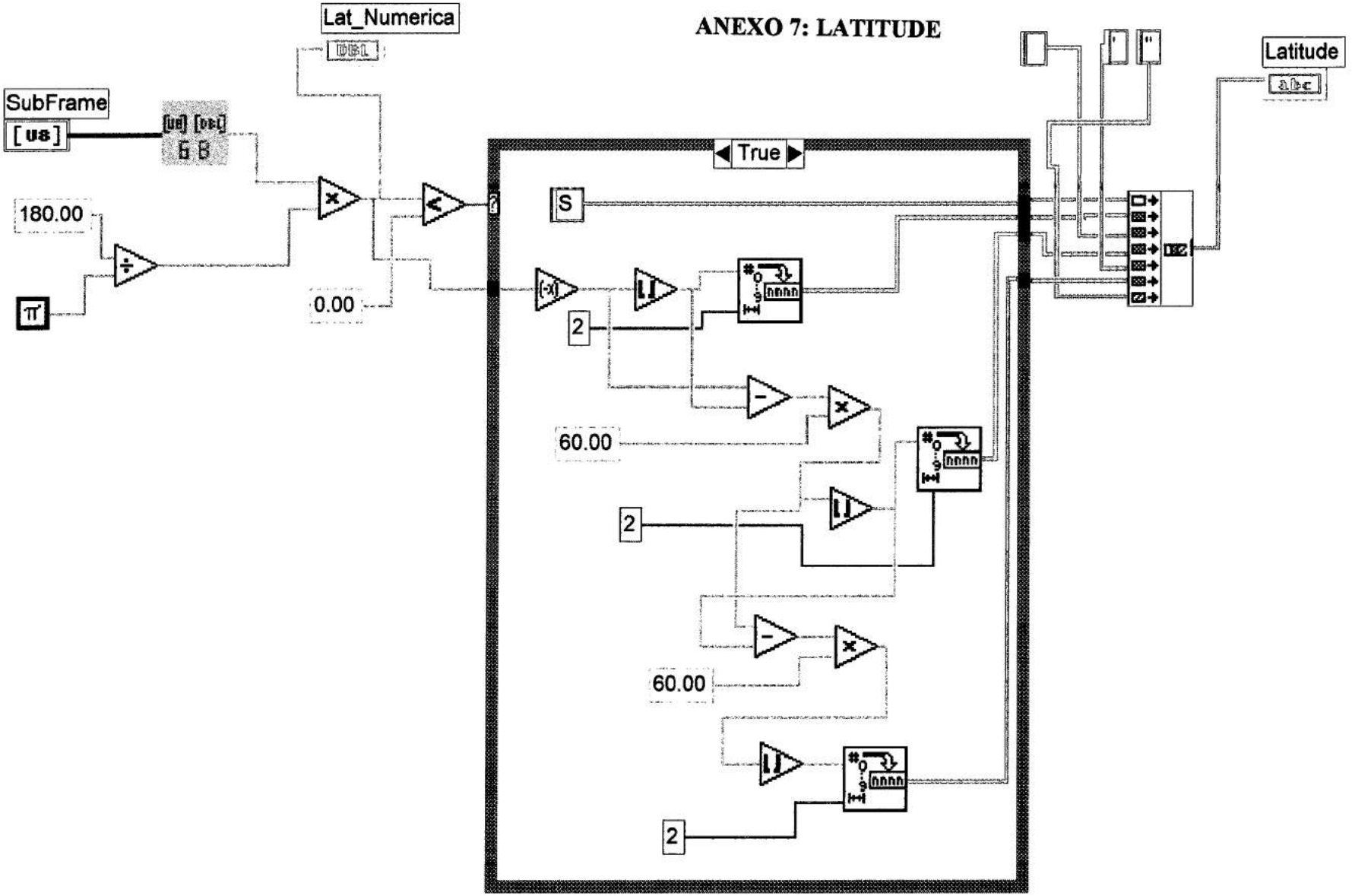
ANEXO 5: ALTITUDE

23

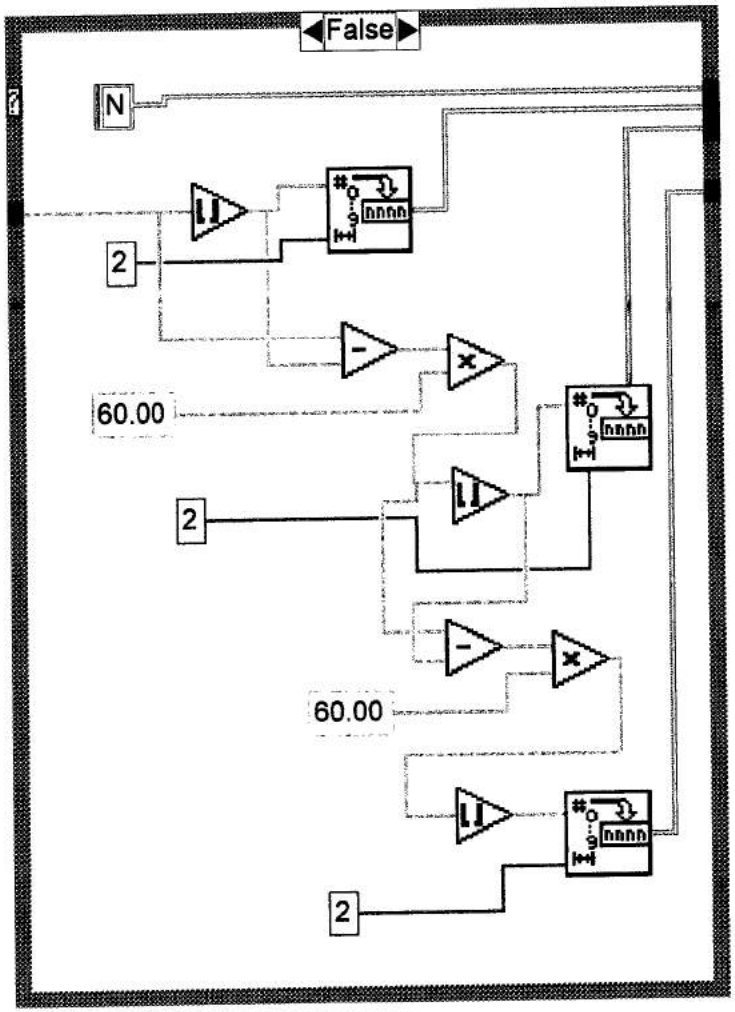
Calcula_Latitude.vi
06/09/97 10:48 AM

Block Diagram

ANEXO 7: LATITUDE

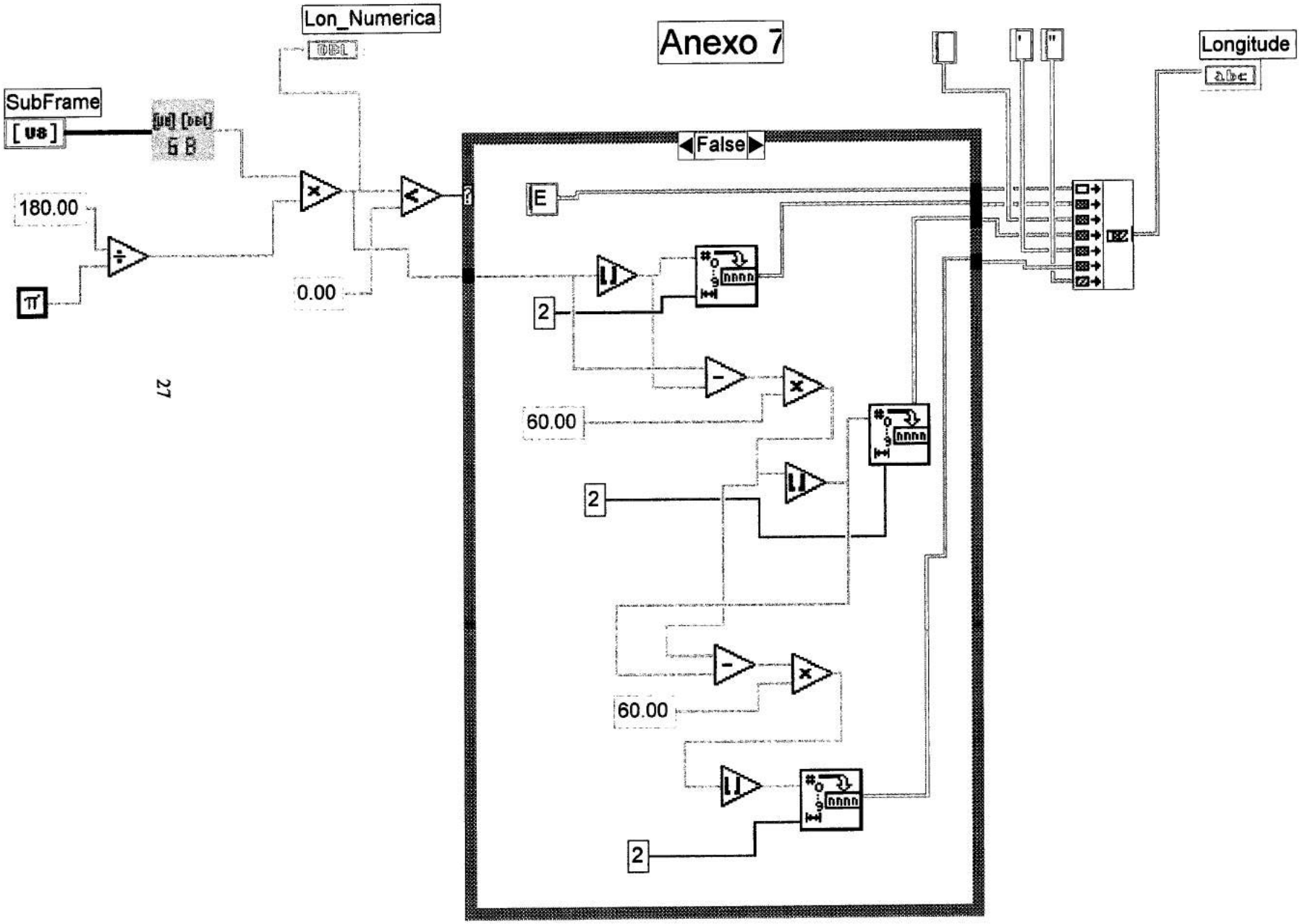


:alcula_Latitude.vi
6/09/97 10:48 AM

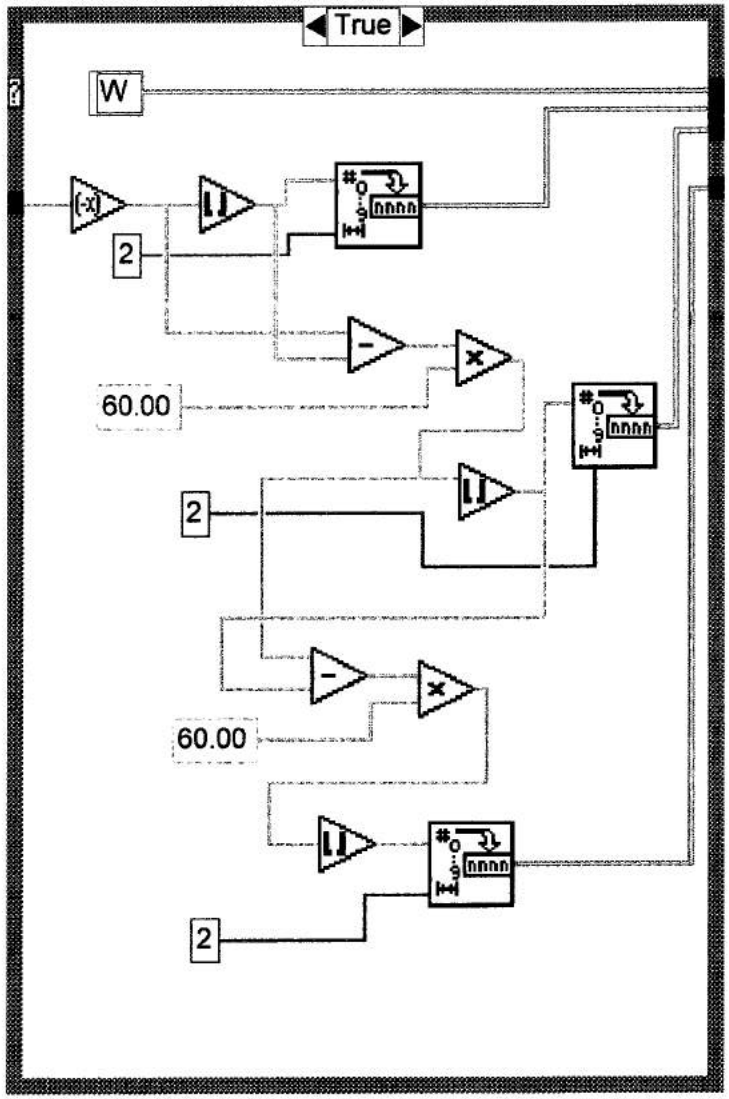


Calcula_Longitude.vi
06/09/97 10:48 AM

Block Diagram



Calcula_Longitude.vi
06/09/97 10:48 AM



Anexo 8

Formato do arquivo binário:

Os delimitadores '[ ]' identificam palavras de 16 bits, os números contidos dentro deste delimitadores fazem referência ao campo *Palavra* do documento **'Quadro de Dados (Frame) GPS/96'.** As palavras 27 a 59 têm seus bytes redistribuídos em palavras de 16 bits, isto é, o primeiro e o segundo byte da uma determinada palavra foram colocados em uma nova palavra de 16 bits como sendo o segundo byte, sendo o primeiro 00. No esquema abaixo os delimitadores '{ }' são usados para delimitar bytes.

[73] [2] [3] ....[26] [27a] [27b] .... [59a] [59b] [{EB}{90}]

[59a] = [{00} {byte1}]
[59b] = [{00} {byte2}]

- O arquivo tem um total de 500 frames.
- Considerando que cada Frame no arquivo binário ocupa 200 bytes temos um total de 200 kbytes por arquivo.

QUADRO DE DADOS (Frame) GPS/96

Palavra:	Ref. Esquemática:	1o. Byte (No Byte)	2o. Byte (No Byte)	Observação:	Grupo de Apresentação:
1	SYNC	EB	90	Sincronismo	Controle Interno
2	AM+5V (IN_ANA12)	LSB (0)	MSB		Analógico
3	AM+12V(IN_ANA13)	LSB (1)	MSB		Analógico
4	AM-12V(IN_ANA14)	LSB (2)	MSB		Analógico
5	AM+5V (IN_ANA15)	LSB (3)	MSB		Analógico
6	AM-15V (IN_ANA16)	LSB (4)	MSB		Analógico
7	CE1:AM+18V(IN_ANA17)	LSB (5)	MSB	Campo Elet.1	Analógico
8	CE1:AM+18V(IN_ANA18)	LSB (6)	MSB	Campo Elet.2	Analógico
9	AM+24V (IN_ANA19)	LSB (7)	MSB		Analógico
10	AM-24V (IN_ANA20)	LSB (8)	MSB		Analógico
11	AM12VGPS (IN_ANA21)	LSB (9)	MSB		Analógico
12	AM12VTR(IN_ANA22)	LSB (10)	MSB		Analógico
13	AM24VLAS(IN_ANA23)	LSB (11)	MSB		Analógico
14	ST1 (IN_ANA1)	LSB (12)	MSB	ST int (uC)	Analógico
15	ST2 (IN_ANA2)	LSB (13)	MSB	ST ext(TX)	Analógico
16	ST3 (IN_ANA3)	LSB (14)	MSB	ST GPS1	Analógico
17	ST4 (IN_ANA4)	LSB (15)	MSB	ST int	Analógico
18	ST5 (IN_ANA5)	LSB (16)	MSB	ST Bat	Analógico
19	ST6 (IN_ANA6)	LSB (17)	MSB	ST ext	Analógico
20	CE1	LSB (18)	MSB	Campo Elet.1	Analógico
21	CE2	LSB (19)	MSB	Campo Elet.2	Analógico
22	SP1000-1 (IN_ANA7)	LSB (20)	MSB		Analógico
23	SP50-1 (IN_ANA8)	LSB (21)	MSB		Analógico
24	SP1000-2 (IN_ANA9)	LSB (22)	MSB		Analógico
25	SP50-2 (IN_ANA10)	LSB (23)	MSB		Analógico
26	SP10 (IN_ANA11)	LSB (24)	MSB		Analógico
27	GPSVELE1	Byte1(0)	Byte2(1)		GPS
28	GPSVELE2	Byte1(2)	Byte2(3)		GPS
29	GPSVELN1	Byte1(4)	Byte2(5)		GPS
30	GPSVELN2	Byte1(6)	Byte2(7)		GPS
31	GPSVELU1	Byte1(8)	Byte2(9)		GPS
32	GPSVELU2	Byte1(10)	Byte2(11)		GPS
33	CE1	Byte1(12)	Byte2(13)	Campo Elet.1	Analógico
34	CE2	Byte1(14)	Byte2(15)	Campo Elet.2	Analógico
35	GPSLAT1	Byte1(16)	Byte2(17)		GPS
36	GPSLAT2	Byte1(18)	Byte2(19)		GPS
37	GPSLAT3	Byte1(20)	Byte2(21)		GPS
38	GPSLON1	Byte1(22)	Byte2(23)		GPS
39	GPSLON2	Byte1(24)	Byte2(25)		GPS
40	GPSLON3	Byte1(26)	Byte2(27)		GPS
41	GPSALT1	Byte1(28)	Byte2(29)		GPS
42	GPSALT2	Byte1(30)	Byte2(31)		GPS
43	GPSSTATUS	Byte1(32)	Byte2(33)		GPS
44	GPSTIME1	Byte1(34)	Byte2(35)		GPS
45	GPSTIME2	Byte1(36)	Byte2(37)		GPS
46	GPSTIME3	Byte1(38)	Byte2(39)		GPS
47	HERRO	Byte1(40)	Byte2(41)		GPS
48	VERRO	Byte1(42)	Byte2(42)		GPS

49	CE1	Byte1(44)	Byte2(45)	Campo Elet.1	Analógico
50	CE2	Byte1(46)	Byte2(47)	Campo Elet.2	Analógico
51	BUSS1-2	Byte1(48)	Byte2(49)	Ver Ref. 1	BUSSOLA
52	BUSS3-4	Byte1(50)	Byte2(51)	Ver Ref. 1	BUSSOLA
53	BUSS5 e 0	Byte1(52)	Byte2(53)	Ver Ref. 1	BUSSOLA
54	STATUSGERAL	Byte1(54)	Byte2(55)	Ver Ref. 2	STATUS
55	CONTADOR FRAMES	LSB(56)	MSB(57)	Cont. 16 Bites	STATUS
56	CRCFRAME	LSB(58)	MSB(59)	N. 1's(P2-54)	Controle Interno
57-59	0's				

Referência 1:

5 bytes referentes ao ângulo medido

Frame em ASCII: $HCHDM,abc.d,M*<checksum><CR><LF>

onde $HCHDM:indicador de direção padrão NMEA
abc.d:ângulo medido
M:indicador de ângulo referente ao N magnético
checksum: XOR'ing dos caracteres entre $ e *

Referência 2:

BYTE1:

Informação referente ao estado do frame do receptor Gps.

MSB							**LSB**
BIT7 G1	BIT6 G2	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0

G1	Função
0	Erro GPS
1	GPS OK

G2	Função
0	Frame GPS está sendo recebido com erros
1	Frame GPS não está sendo recebido

```
////////////////////////////////////////////////////////////////////////////
// Masco.cpp                                                              //
// Objetivo: Ler os arquivos bin rios gerados pelo programa de aquisicao  //
// da telemetria do experimento (masco.vi Lab View), converte-los para um //
// formato adequado a analise dos dados e concatenar estes arquivos quando //
// necessario de acordo com o tempo de voo registrado no proprio frame.   //
////////////////////////////////////////////////////////////////////////////


#include "stdafx.h"
#include "masco.h"
#include "mainfrm.h"
#include "mascodoc.h"
#include "mascovw.h"

#include <stdio.h>          // inclui comandos para gerenciar arquivos
#include <string.h>         // inclui comandos para gerenciar string
#include <math.h>
#include <malloc.h>          // inclui comandos para gerenciar memoria

#include "mascdlg.h"



////////////////////////////////////////////////////////////////////////////

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////////////////////////////////////////////
// CMascoApp

BEGIN_MESSAGE_MAP(CMascoApp, CWinApp)
   ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
   ON_COMMAND(ID_HELP_HELP, OnHelp)
   ON_COMMAND(ID_FILE_OPEN, CMascoApp::OnFileOpen)
   ON_COMMAND(ID_FILE_SAVE_, CMascoApp::OnFileSave)
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
// CMascoApp construction

CMascoApp::CMascoApp()
{
}

////////////////////////////////////////////////////////////////////////////
// The one and only CMascoApp object

CMascoApp NEAR theApp;

////////////////////////////////////////////////////////////////////////////
// CMascoApp initialization

BOOL CMascoApp::InitInstance()
{
   // Standard initialization

   SetDialogBkColor();       // set dialog background color to gray
   LoadStdProfileSettings();  // Load standard INI file options (including MRU)

   AddDocTemplate(new CSingleDocTemplate(IDR_MASCOTYPE,
         RUNTIME_CLASS(CMascoDoc),
```

```
        RUNTIME_CLASS(CMainFrame),     // main SDI frame window
        RUNTIME_CLASS(CMascoView)));

    OnFileNew();

    if (m_lpCmdLine[0] != '\0')
    {
    }

    CWnd* wind=CWnd::GetActiveWindow();
    wind->SetWindowText("Masco Application");

    return TRUE;
}


void CMascoApp::OnAppAbout()
{
    CAboutDlg aboutDlg;
    aboutDlg.DoModal();
}


void CMascoApp::OnHelp()
{

}


/////////////////////////////////////////////////////////////////////////////
// CMascoApp commands


/////////////////////////////////////////////////////////////////////////////
//                                                                         //
// Programa principal.                                                     //
//                                                                         //
// Entrada: Arquivo em binario.                                            //
//          Formato do nome do arquivo de entrada:dt_____.dat.             //
// Saida:   Arquivo em ASCII .                                             //
//          Formato do nome do arquivo de saida:dt_____.out.               //
//          Formato do arquivo em ASCII:                                   //
//          HORA   HORAdecimal  NoFrame  Palavra2 ... Palavra74            //
//                                                                         //
//          * Envia as mensagens de erro tabeladas em trataerro            //
//                                                                         //
/////////////////////////////////////////////////////////////////////////////

void CMascoApp::OnFileOpen()
{
  CString pathname,auxstr;
  int err,ans,init,ind;
  int pos[MAXFILES];


  CFileDialog dialog(TRUE,"*.dat","*.dat",OFN_READONLY,NULL,NULL);
  dialog.m_ofn.Flags=dialog.m_ofn.Flags|OFN_ALLOWMULTISELECT;
  ans=dialog.DoModal();

// printf("\nEntre com o nome do arquivo a ser lido (*.dat):");
// scanf("%s",filename);    // obtem nome do arquivo

  if (ans==IDOK)
  {
    pathname=dialog.GetPathName();
```

```
    if (pathname.Find(' ')==-1)
    {
      ind=pathname.ReverseFind('\\');
      pathname.SetAt(ind,' ');
    }

    pathname+=' ';
    auxstr=pathname;
    filenum=0;
    while ((pos[filenum]=auxstr.Find(' '))!=-1)
      auxstr=auxstr.Right(auxstr.GetLength()-pos[filenum++]-1);

    init=0;
    for (int aux=0;aux<filenum;aux++)
    {
      filelist[aux]=pathname.Mid(init,pos[aux]);
      init+=pos[aux]+1;
    }
    filenum--;
    filelist[0]+="\\";

//  DoWaitCursor(1);
    CCancelDlg canceldlg;
    err=canceldlg.converte(filelist,filenum);

//  DoWaitCursor(0);
    trataerro(err);
  }

  return;
}


/*****************************************************************************************/

void CMascoApp::OnFileSave()
{
  CString auxstr;
  char filename[80];
  int err,ans;

  CFileDialog dialog(FALSE,"*.asc","*.asc",OFN_READONLY,NULL,NULL);
  ans=dialog.DoModal();

  if (ans==IDOK)
  {
    auxstr=dialog.GetPathName();
    strcpy(filename,auxstr);
    DoWaitCursor(1);
      err=gravacao(&filename[0]);
    DoWaitCursor(0);
    trataerro(err);
  }

  return;
}


/*****************************************************************************************/


//////////////////////////////////////////////////////////////////////////
//                                                                      //
// Leitura:  Abre arquivo bin rio e realiza a leitura dos 1000 frames   //
```

```
//          contidos em cada um desses arquivos. Verifica erros do tipo   //
//          arquivo inexistente, frame com palavras a menos e arquivos    //
//          com menos de 1000 frames.                                      //
// Retornos:  ans=0. Retorna True. NÆo ocorreu erro.                       //
//          ans=2. Arquivo inexistente.                                    //
//          ans=3. Faltam palavras no frame.                               //
//          ans=4. Erro no n£mero de frames.                               //
//                                                                         //
/////////////////////////////////////////////////////////////////////////////

int leitura(char* filename)
{
  BYTE byt;
  int lop,nframe;
  FILE *arq;                          // apontador para arquivo, arquivo a ser aberto tipo *arq.
  char auxname[50];
  int ans;
  long tamarq;

  strcpy(auxname,filename);
//  strcat(auxname,".dat");
  ans=255;

  if ((arq=fopen(auxname,"rb"))!=NULL)        // Se arquivo diferente de null ele foi aberto. Continua leitura.
  {
    tamarq=filesize(arq);
    ans=0;
    nframe=1;
    while ((nframe<=Framenum)&&(!feof(arq))&&(ans==0))   // Continua ate terminar de ler todos os frames, nao ocor
rendo fim de
                          // arquivo, nesse caso o numero de frames esta errado.
    {
      lop=1;
      while ((lop<=Datasize)&&(!feof(arq)))       // Continua ate ler todas as palavras, nao ocorrendo fim de arq
uivo,
                                    // nesse caso, o numero de palavras esta errado.
      {
        fread(&byt,1,1,arq);                  // le byte + significativo
        dados[nframe][lop]=byt*256;
        fread(&byt,1,1,arq);                  // le byte - significativo
        dados[nframe][lop]+=byt;
        lop++;                            // incrementa o lop
      }

      if (feof(arq))                        // erro no frame. Faltam palavras.
        ans=3;

      nframe++;                           // incrementa numero dos frames
    }

    if ((feof(arq))&&(ans==0))
      ans=4;                              // erro no tamanho do frame.

    fclose(arq);
  }
  else
    ans=2;                                // arquivo inexistente.
  return ans;
}


/////////////////////////////////////////////////////////////////////////////
//                                                                         //
// Gravacao:  Grava no arquivo de saÍda o cabecario Hora,Hora decimal, No   //
//          Frame e as palavras 2 ate 74.Grava nesse cabecario os dados    //
```

```
//          obtidos.                                                    //
// Retornos:  ans=0. NÆo ocorreu erro.                                  //
//            ans=5. Ocorreu erro na grava‡Æo.                          //
////////////////////////////////////////////////////////////////////////

int gravacao(char* filename)
{
 int lop,nframe,ans;
 float dechour;
 long int dado;
 char time[10];
 FILE *arq;
 char auxname[50];

 ans=0;
 filename[strlen(filename)-4]='\0';
 strcpy(auxname,filename);
 strcat(auxname,".asc");

 if ((arq=fopen(auxname,"w"))!=NULL)          // se for nulo, nao pode abrir arquivo.
 {
  fprintf(arq," HORA      HORA decimal  No Frame    latitude     longitude    altitude "); // inicio do
cabecario

  for (lop=2;lop<75;lop++)                    // joga no arquivo o resto do cabecario
   fprintf(arq," Palavra%2u  ",lop);
  fprintf(arq,"\n");                          // termina linha.

  for (nframe=1;nframe<=Framenum;nframe++)     // percorre todos os frames
  {
   // grava hora
   strcpy(time,printtime(nframe,&dechour));
   fprintf(arq,time);

   // grava hora em decimal
   fprintf(arq,"     %5.5f",dechour);

   // grava numero do frame
    dado=dados[nframe][conv(FrameCont)+1]*256;
    dado+=dados[nframe][conv(FrameCont)];
   fprintf(arq,"     %4.6u",dado);

   // grava latitude
   fprintf(arq,"     %s",latitude(nframe));

   // grava longitude
   fprintf(arq,"   %s",longitude(nframe));

   // grava altitude
   fprintf(arq,"   %s",altitude(nframe));

   for (lop=2;lop<=NumWords;lop++)            // grava 2-49 palavras
     fprintf(arq,"      %4.5u",dados[nframe][lop]);

   for (lop=NumWords+1;lop<=Datasize-1;lop+=2)        // grava 50-74 palavras longas
    {
     dado=dados[nframe][lop+1]*256;
     dado+=dados[nframe][lop];
     fprintf(arq,"      %4.5u",dado);
    }
   fprintf(arq,"\n");
  }
  fclose(arq);
 }
 else
```

```
   ans = 5;                                  // erro na grava‡Æo do arquivo.

  return ans;
}



////////////////////////////////////////////////////////////////////////
//                                                                    //
// Printtime: Calcula a hora do Frame. Formato decimal e hh:mm:ss.    //
// Retornos:  Hora do frame e  hora em decimal.                       //
////////////////////////////////////////////////////////////////////////

char* printtime(int nframe,float* dechour)
{
  char* str = (char*)malloc(10);
  double mantissa,exponent,sign,utct;
  int hour,minute,second;
  int Time = Latitude + 9;

  WORD utctime[7];     // carrega vetor com os valores lidos do arquivo
  utctime[1] = dados[nframe][conv(Time)];
  utctime[2] = dados[nframe][conv(Time) + 1];
  utctime[3] = dados[nframe][conv(Time + 1)];
  utctime[4] = dados[nframe][conv(Time + 1) + 1];
  utctime[5] = dados[nframe][conv(Time + 2)];
  utctime[6] = dados[nframe][conv(Time + 2) + 1];


  if (utctime[6] & 0x80) sign = -1;
  else sign = 1;

  exponent = (unsigned char) utctime[1]-128;
  mantissa = (unsigned char) utctime[2];
  mantissa/ = 256.0;
  mantissa + = (unsigned char) utctime[3];
  mantissa/ = 256.0;
  mantissa + = (unsigned char) utctime[4];
  mantissa/ = 256.0;
  mantissa + = (unsigned char) utctime[5];
  mantissa/ = 256.0;
  mantissa + = (unsigned char) utctime[6] & 0x7f;
  mantissa/ = 256.0;
  mantissa + = 0.5;

  utct = mantissa*pow(2.0,exponent)*sign;

  hour = (int)(utct/3600.0);
  utct- = (hour*3600.0);
  minute = (int)(utct/60.0);
  utct- = (minute*60.0);
  second = (int)(utct);
  if (hour > 24) hour = 0;
  if (minute > 59) minute = 0;
  if (second > 59) second = 0;

  *dechour = hour + (float)(minute/60.0) + (float)(second/3600.0);

  sprintf(str," %02d:%02d:%02d ",hour,minute,second);
  return &str[0];
}


////////////////////////////////////////////////////////////////////////
```

```
//                                                                    //
// Latitude: Calcula a latitude enviada pelo GPS atraves de dados do frame.  //
// Retornos:  latitude.                                               //
////////////////////////////////////////////////////////////////////////

char* latitude(int nframe)
{
  char str[10];
  double mantissa,exponent,sign,utct;
  double latitude_numerica;
  int grau,minuto,segundo;

  WORD utctime[7];     // carrega vetor com os valores lidos do arquivo
  utctime[1]=dados[nframe][conv(Latitude)];
  utctime[2]=dados[nframe][conv(Latitude)+1];
  utctime[3]=dados[nframe][conv(Latitude+1)];
  utctime[4]=dados[nframe][conv(Latitude+1)+1];
  utctime[5]=dados[nframe][conv(Latitude+2)];
  utctime[6]=dados[nframe][conv(Latitude+2)+1];


  if (utctime[6] & 0x80) sign=-1;
  else sign=1;

  exponent=(unsigned char) utctime[1]-128;
  mantissa=(unsigned char) utctime[2];
  mantissa/=256.0;
  mantissa+=(unsigned char) utctime[3];
  mantissa/=256.0;
  mantissa+=(unsigned char) utctime[4];
  mantissa/=256.0;
  mantissa+=(unsigned char) utctime[5];
  mantissa/=256.0;
  mantissa+=(unsigned char) utctime[6] & 0x7f;
  mantissa/=256.0;
  mantissa+=0.5;

  utct=mantissa*pow(2.0,exponent)*sign;

  latitude_numerica=utct*(180/PI);

  if (latitude_numerica<0)

     {
       latitude_numerica=latitude_numerica*-1;
       grau=(int)(latitude_numerica);
       latitude_numerica=latitude_numerica - grau;
       minuto=(int) (latitude_numerica*60.00);
       latitude_numerica=latitude_numerica*60.00 - minuto;
       segundo=(int)(latitude_numerica * 60.00);

         if (grau>360) grau=0;
         if (minuto>59) minuto=0;
         if (segundo>59) segundo=0;

  sprintf(str," S %02d*%02d'%02d'' ",grau,minuto,segundo);

     }

  else

    {
       grau=(int)(latitude_numerica);
       latitude_numerica=latitude_numerica - grau;
       minuto=(int) (latitude_numerica*60.00);
```

```
        latitude_numerica=latitude_numerica*60.00 - minuto;
        segundo=(int)(latitude_numerica * 60.00);

          if (grau>360) grau=0;
          if (minuto>59) minuto=0;
          if (segundo>59) segundo=0;

  sprintf(str," N %02d*%02d'%02d'' ",grau,minuto,segundo);

        }

      return &str[0];
}


////////////////////////////////////////////////////////////////////////////
//                                                                        //
// Longitude: Calcula a longitude enviada pelo GPS atraves de dados do frame.//
// Retornos:  longitude.                                                  //
////////////////////////////////////////////////////////////////////////////

char* longitude(int nframe)
{
  char str[10];
  double mantissa,exponent,sign,utct;
  double longitude_numerica;
  int grau,minuto,segundo;
  int Longitude=Latitude+3;

  WORD utctime[7];     // carrega vetor com os valores lidos do arquivo
  utctime[1]=dados[nframe][conv(Longitude)];
  utctime[2]=dados[nframe][conv(Longitude)+1];
  utctime[3]=dados[nframe][conv(Longitude+1)];
  utctime[4]=dados[nframe][conv(Longitude+1)+1];
  utctime[5]=dados[nframe][conv(Longitude+2)];
  utctime[6]=dados[nframe][conv(Longitude+2)+1];


  if (utctime[6] & 0x80) sign=-1;
  else sign=1;

  exponent=(unsigned char) utctime[1]-128;
  mantissa=(unsigned char) utctime[2];
  mantissa/=256.0;
  mantissa+=(unsigned char) utctime[3];
  mantissa/=256.0;
  mantissa+=(unsigned char) utctime[4];
  mantissa/=256.0;
  mantissa+=(unsigned char) utctime[5];
  mantissa/=256.0;
  mantissa+=(unsigned char) utctime[6] & 0x7f;
  mantissa/=256.0;
  mantissa+=0.5;

  utct=mantissa*pow(2.0,exponent)*sign;

  longitude_numerica=utct*(180/PI);

  if (longitude_numerica<0)

     {
       longitude_numerica=longitude_numerica*-1;
       grau=(int)(longitude_numerica);
       longitude_numerica=longitude_numerica - grau;
       minuto=(int) (longitude_numerica*60.00);
```

```
      longitude_numerica=longitude_numerica*60.00 - minuto;
      segundo=(int)(longitude_numerica * 60.00);

        if (grau>360) grau=0;
        if (minuto>59) minuto=0;
        if (segundo>59) segundo=0;

 sprintf(str," W %02d*%02d'%02d'' ",grau,minuto,segundo);

   }

 else

      {

      grau=(int)(longitude_numerica);
      longitude_numerica=longitude_numerica - grau;
      minuto=(int) (longitude_numerica*60.00);
      longitude_numerica=longitude_numerica*60.00 - minuto;
      segundo=(int)(longitude_numerica * 60.00);

        if (grau>360) grau=0;
        if (minuto>59) minuto=0;
        if (segundo>59) segundo=0;

 sprintf(str," E %02d*%02d'%02d'' ",grau,minuto,segundo);

      }

    return &str[0];
}


////////////////////////////////////////////////////////////////////////////
//                                                                        //
// Altitude: Calcula a altitude do Frame.                                 //
// Retornos:  altitude.                                                   //
////////////////////////////////////////////////////////////////////////////

char* altitude (int nframe)
{
  unsigned char alt_temp[5];
  char str[10];
  double var3,var4,altitude;
  unsigned char var1,var2;
  int Altitude=Latitude+6;

   // carrega vetor com os valores lidos do arquivo
  alt_temp[1]=(unsigned char)dados[nframe][conv(Altitude)];
  alt_temp[2]=(unsigned char)dados[nframe][conv(Altitude)+1];
  alt_temp[3]=(unsigned char)dados[nframe][conv(Altitude+1)];
  alt_temp[4]=(unsigned char)dados[nframe][conv(Altitude+1)+1];

  var1=alt_temp[4] & 127;
  var2=alt_temp[4] & 128;
  var3=(((double)alt_temp[2]/256.0)+(double)alt_temp[3])/256;
  var4=((var3+(double)var2)/256.0)+0.5;

  altitude=(pow(2.0,(double)alt_temp[1]-128.0))*var4;

  if(var1==1) altitude=altitude*-1;

  sprintf(str," %0.0f ",altitude);

  return &str[0];
```

```
}



////////////////////////////////////////////////////////////////////////
//                                                                    //
// Conv: Converte valor da longword para word no vetor.               //
//                                                                    //
// Retornos: retorna valor da word no vetor.                          //
//                                                                    //
////////////////////////////////////////////////////////////////////////

int conv(int val)
{
  if (val < NumWords + 1)
    return val;
  else
    return ((val-NumWords)*2 + NumWords-1);    // converte valor da longword para word no vetor

}


long filesize(FILE* arq)
{
  long tam;
  fseek(arq,0,SEEK_END);
  fgetpos(arq,&tam);
  fseek(arq,0,SEEK_SET);
  switch (tam)
  {
    case 200000:
      {
        Datasize = 100;
        Framenum = 1000;
        NumWords = 49;
        FrameCont = 73;
        Latitude = 60;
        break;
      }

    case 93000:
      {
        Datasize = 93;
        Framenum = 500;
        NumWords = 26;
        FrameCont = 55;
        Latitude = 35;
        break;
      }

    default:
      trataerro(4);
  }
  return (tam);
}


////////////////////////////////////////////////////////////////////////
//                                                                    //
// trataerro: Tabela de erros. Imprime na tela mensagens de erro.     //
//                                                                    //
////////////////////////////////////////////////////////////////////////

void trataerro(int err)
```

```
{
  char mens[255];
  switch (err)
  {
    case 0: { strcpy(mens,"Nao ocorreu erro."); break;}
    case 2: { strcpy(mens,"Arquivo nao existe."); break;}
    case 3: { strcpy(mens,"Faltam palavras no frame."); break;}
    case 4: { strcpy(mens,"Faltam frames no arquivo."); break;}
    case 5: { strcpy(mens,"Erro na gravacao do arquivo."); break;}
    case 255: { strcpy(mens,"Erro desconhecido."); break;}
    default: { strcpy(mens,"Erro nao tratado."); break;}
  }
//  printf("\n%s\n",mens);
  AfxMessageBox(mens);
}
```

```
// masco.h : main header file for the MASCO application
//

#ifndef __AFXWIN_H__
    #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols



/////////////////////////////////////////////////////////////////////////////
// Definicao de tipos

typedef int BOOL;
//#define FALSE 1
//#define TRUE 0

typedef unsigned char BYTE;
//typedef short int WORD;


#define MAXDATASIZE 100   // quantidade maxima de palavras por frame
#define MAXFRAMENUM 1000      // quantidade maxima de frames por arquivo
#define MAXFILES 20
#define PI 3.1416



/////////////////////////////////////////////////////////////////////////////
// Variaveis globais

int filenum;
CString filelist[MAXFILES];
//char filelist[MAXFILES][50];  // nomes dos arquivos
int Datasize;               // quantidade de palavras por frame
int Framenum;                 // quantidade de frames por arquivo
int NumWords;
int FrameCont;
int Latitude;
                                        ← VISUAL C++ 2.0
// WORD dados[MAXFRAMENUM + 1][MAXDATASIZE + 1];   // estrutura que armazena os frames
WORD huge dados[MAXFRAMENUM + 1][MAXDATASIZE + 1];   // estrutura que armazena os frames
                                        ↖ 1.0

/////////////////////////////////////////////////////////////////////////////
// Funcoes globais

int leitura(char *filename);
int gravacao(char* filename);                   // gravacao do arquivo ASC
char* printtime(int nframe,float* dechour);             // converte o tempo para hh:mm:ss
char* latitude(int nframe);                          // calculo da latitude
char* longitude(int nframe);                          // calculo da longitude
char* altitude(int nframe);

int conv(int val);      // obtem o indice efetivo no vetor dados
long filesize(FILE* arq);
void trataerro(int err); // tabela  de erros



/////////////////////////////////////////////////////////////////////////////
// CMascoApp:
// See masco.cpp for the implementation of this class
```

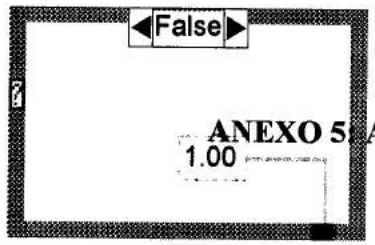
```
//

class CMascoApp : public CWinApp
{
public:
    CMascoApp();

// Overrides
    virtual BOOL InitInstance();
    afx_msg void OnAppAbout();
    afx_msg void OnHelp();
    afx_msg void OnFileOpen();
    afx_msg void OnFileSave();

    DECLARE_MESSAGE_MAP()
};


/////////////////////////////////////////////////////////////////////////////
```



ANEXO 5 ALTITUDE

```
}


int CCancelDlg::converte(CString filelist[MAXFILES],int filenum)
{
  int err,ind;
  char filename[80];

  Create(IDD_DIALOG1,NULL);
  ShowWindow(SW_SHOW);

  ind=1;
  while ((ind<=filenum)&&(!cancel))
  {
    strcpy(filename,filelist[0]);
    strcat(filename,filelist[ind]);

    m_mens="Arquivo: ";
    m_mens+=filename;
    UpdateData(FALSE);

    if (!cancel)
      err=leitura(filename);

    if ((!cancel)&&(err==0))
    {
m_mens="Gravando..";
UpdateData(FALSE);
ShowWindow(SW_SHOW);

      err=gravacao(filename);     // realiza gravacao se leitura bem sucedida
    }
    ind++;
  }

  DestroyWindow();
  return err;
}
```

```
// MascDlg.h


#define MAXFILES 20


/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
   CAboutDlg();

// Dialog Data
   //{{AFX_DATA(CAboutDlg)
   enum { IDD = IDD_ABOUTBOX };
   //}}AFX_DATA

// Implementation
protected:
   virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
   //{{AFX_MSG(CAboutDlg)
      // No message handlers
   //}}AFX_MSG
   DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
// CCancelDlg dialog

class CCancelDlg : public CDialog
{
// Construction
public:
   CCancelDlg(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
   //{{AFX_DATA(CCancelDlg)
   enum { IDD = IDD_DIALOG1 };
   CString m_mens;
   BOOL cancel;
   char filename[80];          // nome do arquivo sem extensao
   //}}AFX_DATA

   afx_msg void OnCancel();
   int converte(CString filelist[MAXFILES],int filenum);

// Overrides
   // ClassWizard generated virtual function overrides
   //{{AFX_VIRTUAL(CCancelDlg)
   protected:
   virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
   //}}AFX_VIRTUAL

// Implementation
protected:

   // Generated message map functions
   //{{AFX_MSG(CCancelDlg)
      // NOTE: the ClassWizard will add member functions here
   //}}AFX_MSG
   DECLARE_MESSAGE_MAP()
```

```
};
```

```
// mainfrm.cpp : implementation of the CMainFrame class
//

#include "stdafx.h"
#include "resource.h"

#include "mainfrm.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMainFrame

IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
   //{{AFX_MSG_MAP(CMainFrame)
   ON_WM_CREATE()
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// arrays of IDs used to initialize control bars

// toolbar buttons - IDs are command buttons
static UINT BASED_CODE buttons[] =
{
   // same order as in the bitmap 'toolbar.bmp'
   ID_FILE_OPEN,
   ID_FILE_SAVE,
      ID_SEPARATOR,
   ID_APP_ABOUT,
};

static UINT BASED_CODE indicators[] =
{
   ID_SEPARATOR,         // status line indicator
   ID_INDICATOR_CAPS,
   ID_INDICATOR_NUM,
   ID_INDICATOR_SCRL,
};

/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction

CMainFrame::CMainFrame()
{
}

CMainFrame::~CMainFrame()
{
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
   if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
      return -1;
```

```
    if (!m_wndToolBar.Create(this) ||
        !m_wndToolBar.LoadBitmap(IDR_MAINFRAME) ||
        !m_wndToolBar.SetButtons(buttons,
          sizeof(buttons)/sizeof(UINT)))
    {
        TRACE("Failed to create toolbar\n");
        return -1;      // fail to create
    }

    if (!m_wndStatusBar.Create(this) ||
        !m_wndStatusBar.SetIndicators(indicators,
          sizeof(indicators)/sizeof(UINT)))
    {
        TRACE("Failed to create status bar\n");
        return -1;      // fail to create
    }
    SetWindowText("Windows Application");

    return 0;
}


/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
    CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
    CFrameWnd::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers
```

```
// mainfrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////

class CMainFrame : public CFrameWnd
{
protected: // create from serialization only
    CMainFrame();
    DECLARE_DYNCREATE(CMainFrame)

// Attributes
public:

// Operations
public:

// Implementation
public:
    virtual ~CMainFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif


protected:  // control bar embedded members
    CStatusBar  m_wndStatusBar;
    CToolBar    m_wndToolBar;

// Generated message map functions
protected:
    //{{AFX_MSG(CMainFrame)
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
```

```
// mascovw.cpp : implementation of the CMascoView class
//

#include "stdafx.h"
#include "resource.h"

#include "mascodoc.h"
#include "mascovw.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMascoView

IMPLEMENT_DYNCREATE(CMascoView, CView)

BEGIN_MESSAGE_MAP(CMascoView, CView)
    //{{AFX_MSG_MAP(CMascoView)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMascoView construction/destruction

CMascoView::CMascoView()
{
}

CMascoView::~CMascoView()
{
}

/////////////////////////////////////////////////////////////////////////////
// CMascoView drawing

void CMascoView::OnDraw(CDC* pDC)
{
    CMascoDoc* pDoc = GetDocument();

}



/////////////////////////////////////////////////////////////////////////////
// CMascoView diagnostics

#ifdef _DEBUG
void CMascoView::AssertValid() const
{
    CView::AssertValid();
}

void CMascoView::Dump(CDumpContext& dc) const
{
    CView::Dump(dc);
}

CMascoDoc* CMascoView::GetDocument() // non-debug version is inline
```

```
{
   ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CMascoDoc)));
   return (CMascoDoc*) m_pDocument;
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CMascoView message handlers
```

```
// mascovw.h : interface of the CMascoView class
//
/////////////////////////////////////////////////////////////////////////////

class CMascoView : public CView
{
protected: // create from serialization only
    CMascoView();
    DECLARE_DYNCREATE(CMascoView)

// Attributes
public:
    CMascoDoc* GetDocument();

// Operations
public:

// Implementation
public:
    virtual ~CMascoView();
    virtual void OnDraw(CDC* pDC);  // overridden to draw this view
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif


// Generated message map functions
protected:
    //{{AFX_MSG(CMascoView)
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // debug version in mascovw.cpp
inline CMascoDoc* CMascoView::GetDocument()
   { return (CMascoDoc*) m_pDocument; }
#endif

/////////////////////////////////////////////////////////////////////////////
```

```
// mascodoc.cpp : implementation of the CMascoDoc class
//

#include "stdafx.h"
#include "resource.h"

#include "mascodoc.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMascoDoc

IMPLEMENT_DYNCREATE(CMascoDoc, CDocument)

BEGIN_MESSAGE_MAP(CMascoDoc, CDocument)
   //{{AFX_MSG_MAP(CMascoDoc)
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMascoDoc construction/destruction

CMascoDoc::CMascoDoc()
{
}

CMascoDoc::~CMascoDoc()
{
}

BOOL CMascoDoc::OnNewDocument()
{
   if (!CDocument::OnNewDocument())
      return FALSE;
   return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CMascoDoc serialization

void CMascoDoc::Serialize(CArchive& ar)
{
   if (ar.IsStoring())
   {
   }
   else
   {
   }
}


/////////////////////////////////////////////////////////////////////////////
// CMascoDoc diagnostics

#ifdef _DEBUG
void CMascoDoc::AssertValid() const
{
```

```
    CDocument::AssertValid();
}

void CMascoDoc::Dump(CDumpContext& dc) const
{
    CDocument::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CMascoDoc commands
```

```
// mascodoc.h : interface of the CMascoDoc class
//
/////////////////////////////////////////////////////////////////////////////

class CMascoDoc : public CDocument
{
protected: // create from serialization only
   CMascoDoc();
   DECLARE_DYNCREATE(CMascoDoc)

// Attributes
public:

// Operations
public:

// Implementation
public:
   virtual ~CMascoDoc();
   virtual void Serialize(CArchive& ar);   // overridden for document i/o
#ifdef _DEBUG
   virtual void AssertValid() const;
   virtual void Dump(CDumpContext& dc) const;
#endif
protected:
   virtual BOOL    OnNewDocument();

// Generated message map functions
protected:
   //{{AFX_MSG(CMascoDoc)
   //}}AFX_MSG
   DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
```

```
// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//      are changed infrequently
//

#include < afxwin.h >         // MFC core and standard components
#include < afxext.h >         // MFC extensions
```

```
//Microsoft Visual C++ generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS


#ifdef APSTUDIO_INVOKED
////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
   "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
   "#include ""afxres.h""\r\n"
   "\0"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
   "#include ""res\\masco.rc2""  // non-App Studio edited resources\r\n"
   "\r\n"
   "#include ""afxres.rc""  // Standard components\r\n"
   "#include ""afxprint.rc""  // printing/print preview resources\r\n"
   "\0"
END

////////////////////////////////////////////////////////////////////////
#endif    // APSTUDIO_INVOKED


////////////////////////////////////////////////////////////////////////
//
// Icon
//

IDR_MASCOTYPE        ICON    DISCARDABLE    "RES\\MASCO.ICO"

////////////////////////////////////////////////////////////////////////
//
// Bitmap
//

IDR_MAINFRAME        BITMAP  MOVEABLE PURE   "RES\\TOOLBAR.BMP"

////////////////////////////////////////////////////////////////////////
//
```

```
// Menu
//

IDR_MASCOTYPE MENU PRELOAD DISCARDABLE
BEGIN
  POPUP "&File"
  BEGIN
    MENUITEM "&Open...\tCtrl+O",          ID_FILE_OPEN
    MENUITEM SEPARATOR
    MENUITEM "Save As...",                ID_FILE_SAVE_
    MENUITEM SEPARATOR
    MENUITEM "E&xit",                     ID_APP_EXIT
  END
  POPUP "&View"
  BEGIN
    MENUITEM "&Toolbar",                  ID_VIEW_TOOLBAR
    MENUITEM "&Status Bar",               ID_VIEW_STATUS_BAR
  END
  POPUP "&Help"
  BEGIN
    MENUITEM "Help...",                   ID_HELP_HELP
    MENUITEM "&About MASCO...",           ID_APP_ABOUT
  END
END


////////////////////////////////////////////////////////////////////////
//
// Accelerator
//

IDR_MAINFRAME ACCELERATORS PRELOAD MOVEABLE PURE
BEGIN
  "N",        ID_FILE_NEW,        VIRTKEY, CONTROL
  "O",        ID_FILE_OPEN,       VIRTKEY, CONTROL
  "S",        ID_FILE_SAVE,       VIRTKEY, CONTROL
  "P",        ID_FILE_PRINT,      VIRTKEY, CONTROL
  "Z",        ID_EDIT_UNDO,       VIRTKEY, CONTROL
  "X",        ID_EDIT_CUT,        VIRTKEY, CONTROL
  "C",        ID_EDIT_COPY,       VIRTKEY, CONTROL
  "V",        ID_EDIT_PASTE,      VIRTKEY, CONTROL
  VK_BACK,    ID_EDIT_UNDO,       VIRTKEY, ALT
  VK_DELETE,  ID_EDIT_CUT,        VIRTKEY, SHIFT
  VK_INSERT,  ID_EDIT_COPY,       VIRTKEY, CONTROL
  VK_INSERT,  ID_EDIT_PASTE,      VIRTKEY, SHIFT
  VK_F6,      ID_NEXT_PANE,       VIRTKEY
  VK_F6,      ID_PREV_PANE,       VIRTKEY, SHIFT
END


////////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_ABOUTBOX DIALOG DISCARDABLE  34, 22, 197, 55
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "About MASCO"
FONT 8, "MS Sans Serif"
BEGIN
  ICON        IDR_MASCOTYPE,IDC_STATIC,11,17,18,20
```

```
    LTEXT           "MASCO Application Version 3.1",IDC_STATIC,40,10,109,8
    LTEXT           "Copyright \251 1997",IDC_STATIC,40,38,65,8
    DEFPUSHBUTTON   "OK",IDOK,154,36,35,14,WS_GROUP
    LTEXT           "Erika Trench Sestari",IDC_STATIC,40,26,80,7
END

IDD_DIALOG1 DIALOG DISCARDABLE  50, 40, 190, 58
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Convertendo..."
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "Cancel",IDCANCEL,130,34,50,14
    EDITTEXT        IDC_EDIT1,9,12,170,12,ES_AUTOHSCROLL | ES_READONLY
END


/////////////////////////////////////////////////////////////////////////////
//
// String Table
//

STRINGTABLE PRELOAD DISCARDABLE
BEGIN
    IDR_MAINFRAME           "MASCO Windows Application"
    IDR_MASCOTYPE           "\nMasco\nMASCO Document"
END

STRINGTABLE PRELOAD DISCARDABLE
BEGIN
    AFX_IDS_APP_TITLE       "MASCO Windows Application"
    AFX_IDS_IDLEMESSAGE     "Ready"
END

STRINGTABLE DISCARDABLE
BEGIN
    ID_INDICATOR_EXT        "EXT"
    ID_INDICATOR_CAPS       "CAP"
    ID_INDICATOR_NUM        "NUM"
    ID_INDICATOR_SCRL       "SCRL"
    ID_INDICATOR_OVR        "OVR"
    ID_INDICATOR_REC        "REC"
END

STRINGTABLE DISCARDABLE
BEGIN
    ID_FILE_NEW             "Create a new document"
    ID_FILE_OPEN            "Open an existing document"
    ID_FILE_CLOSE           "Close the active document"
    ID_FILE_SAVE            "Save the active document"
    ID_FILE_SAVE_           "Save the active document with a new name"
    ID_FILE_PAGE_SETUP      "Change the printing options"
    ID_FILE_PRINT_SETUP     "Change the printer and printing options"
    ID_FILE_PRINT           "Print the active document"
    ID_FILE_PRINT_PREVIEW   "Display full pages"
END

STRINGTABLE DISCARDABLE
BEGIN
    ID_APP_ABOUT            "Display program information, version number and copyright"
    ID_HELP_HELP            "Display program help"
    ID_APP_EXIT             "Quit the application"
```

```
END

STRINGTABLE DISCARDABLE
BEGIN
    ID_VIEW_TOOLBAR         "Show or hide the toolbar"
    ID_VIEW_STATUS_BAR      "Show or hide the status bar"
END

STRINGTABLE DISCARDABLE
BEGIN
    AFX_IDS_SCSIZE          "Change the window size"
    AFX_IDS_SCMOVE          "Change the window position"
    AFX_IDS_SCMINIMIZE      "Reduce the window to an icon"
    AFX_IDS_SCMAXIMIZE      "Enlarge the window to full size"
    AFX_IDS_SCNEXTWINDOW    "Switch to the next document window"
    AFX_IDS_SCPREVWINDOW    "Switch to the previous document window"
    AFX_IDS_SCCLOSE         "Close the active window"
END

STRINGTABLE DISCARDABLE
BEGIN
    AFX_IDS_SCRESTORE       "Restore the window to normal size"
    AFX_IDS_SCTASKLIST      "Activate Task List"
    AFX_IDS_MDICHILD        "Activate this window"
END


#ifndef APSTUDIO_INVOKED
////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//
#include "res\masco.rc2"  // non-App Studio edited resources

#include "afxres.rc"  // Standard components
#include "afxprint.rc"  // printing/print preview resources

////////////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```

```
END

STRINGTABLE DISCARDABLE
BEGIN
    ID_VIEW_TOOLBAR         "Show or hide the toolbar"
    ID_VIEW_STATUS_BAR      "Show or hide the status bar"
END

STRINGTABLE DISCARDABLE
BEGIN
    AFX_IDS_SCSIZE          "Change the window size"
    AFX_IDS_SCMOVE          "Change the window position"
    AFX_IDS_SCMINIMIZE      "Reduce the window to an icon"
    AFX_IDS_SCMAXIMIZE      "Enlarge the window to full size"
    AFX_IDS_SCNEXTWINDOW    "Switch to the next document window"
    AFX_IDS_SCPREVWINDOW    "Switch to the previous document window"
    AFX_IDS_SCCLOSE         "Close the active window"
END

STRINGTABLE DISCARDABLE
BEGIN
    AFX_IDS_SCRESTORE       "Restore the window to normal size"
    AFX_IDS_SCTASKLIST      "Activate Task List"
    AFX_IDS_MDICHILD        "Activate this window"
END


#ifndef APSTUDIO_INVOKED
////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//
#include "res\masco.rc2"  // non-App Studio edited resources

#include "afxres.rc"  // Standard components
#include "afxprint.rc"  // printing/print preview resources

////////////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```