```
%**************************************************************************
%* Função: Aplicacao_Grafica
%*
%* Informação: Função que mostra a janela principal de todo o sistema.
%**************************************************************************
function varargout = Aplicacao_Grafica(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Aplicacao_Grafica_OpeningFcn, ...
                   'gui_OutputFcn',  @Aplicacao_Grafica_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);

    if nargin & isstr(varargin{1})
        gui_State.gui_Callback = str2func(varargin{1});
    end

    if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
    else
        gui_mainfcn(gui_State, varargin{:});
    end
end
%%
%**************************************************************************
%* Função: Aplicacao_Grafica_OpeningFcn
%*
%* Informação: Esta função inicializa a janela gráfica da Aplicacao
%              Grafica.
%**************************************************************************
function Aplicacao_Grafica_OpeningFcn(hObject, eventdata,handles,varargin)

global Painel_Central Painel_Canais Contagem Factor_Tempo

%Limpa o texto do ecrãn.
clc
%Variável que indica o número de vezes que foi executado o cálculo.
Contagem = 0;
%Define variável que permite ajustar o valor do eixo do tempo (xx) para
% que o tempo apareça na unidade de "us".
Factor_Tempo = 1000;

handles.output = hObject;
guidata(hObject, handles);

%Coloca a Janela no Centro do Ecran.
FigPos=get(0,'DefaultFigurePosition');
OldUnits = get(hObject, 'Units');
set(hObject, 'Units', 'pixels');
OldPos = get(hObject,'Position');
FigWidth = OldPos(3);
FigHeight = OldPos(4);

if isempty(gcbf)
    ScreenUnits=get(0,'Units');
```

```
    set(0,'Units','pixels');
    ScreenSize=get(0,'ScreenSize');
    set(0,'Units',ScreenUnits);

    FigPos(1)=1/2*(ScreenSize(3)-FigWidth);%1/2 é metade do ecran.
    FigPos(2)=1/2*(ScreenSize(4)-FigHeight);%1/2 é metade do ecran.
else
    GCBFOldUnits = get(gcbf,'Units');
    set(gcbf,'Units','pixels');
    GCBFPos = get(gcbf,'Position');
    set(gcbf,'Units',GCBFOldUnits);
    FigPos(1:2) = [(GCBFPos(1) + GCBFPos(3) / 2) - FigWidth / 2, ...
                   (GCBFPos(2) + GCBFPos(4) / 2) - FigHeight / 2];
end
FigPos(3:4)=[FigWidth FigHeight];
set(hObject, 'Position', FigPos);
set(hObject, 'Units', OldUnits);

%Define os paineis e respectivas posições no ecran.
Pos_Painel_Canais = [50,520,800,90];
Painel_Canais=uipanel('Units','pixels','Position',Pos_Painel_Canais, ...
    'Title','Sinal de entrada dos 3 canais','FontSize',12,...
    'BackgroundColor','white');

Pos_Painel_Central = [50,70,800,450];
Painel_Central=uipanel('Units','pixels','Position',Pos_Painel_Central, ...
    'Title','Sinal Medido','FontSize',12,'BackgroundColor','white');

Pos_Painel_Ficheiro = [150,20,600,50];
Painel_Ficheiro=uipanel('Units','pixels','Position', ...
    Pos_Painel_Ficheiro,'Title','Ficheiro','FontSize',12, ...
    'BackgroundColor','white');

Texto_Ficheiro = uicontrol('Parent',Painel_Ficheiro,'Style', ...
    'Text','Position',[10,8,580,16],'FontWeight','bold',...
    'HorizontalAlignment','left',...
    'FontSize',10,'ForegroundColor',...
    'blue','BackgroundColor','white',...
    'Tag','Texto_Ficheiro');

%Efectua o carregamento da imagem do ficheiro na aplicação gráfica.
[Imagem_Open, Imagem_Cores] = imread('Open.gif');
Imagem_RGB = ind2rgb(Imagem_Open,Imagem_Cores);
button = uicontrol('style','pushbutton','position',...
    [68,10,80,60],'BackgroundColor','white','Callback',@Ficheiro);
set(button,'cdata',Imagem_RGB);

%Efectua o carregamento da imagem do "play" na aplicação gráfica.
[Imagem_Play, Imagem_Cores] = imread('Play.gif');
Imagem_RGB = ind2rgb(Imagem_Play,Imagem_Cores);
button = uicontrol('style','pushbutton','position',...
    [750,10,80,60],'BackgroundColor','white','userdata',0,...
    'Callback',@Play);
set(button,'cdata',Imagem_RGB);

end
%%
```

```
%**************************************************************************
%* Função: Aplicacao_Grafica_OutputFcn
%*
%* Informação: Esta função mostra a janela gráfica da Aplicacao Grafica.
%**************************************************************************
function varargout = Aplicacao_Grafica_OutputFcn(hObject,eventdata,handles)

varargout{1} = handles.output;

Menu_Opcoes();

end
%%
%**************************************************************************
%* Função: Menu_Opcoes
%*
%* Informação: Esta função Mostra o Menu de Opções.
%**************************************************************************
function Menu_Opcoes()

a = uimenu('Label','Sistema Geral');
    uimenu(a,'Label','Efectuar Cálculo','Accelerator','E','userdata',...
        0,'Callback',@Play);
    uimenu(a,'Label','Sair','separator','on','Accelerator',...
        'S','userdata',-1,'Callback',@Play);

b = uimenu('Label','Ficheiro');
    uimenu(b,'Label','Carregar Ficheiro','Accelerator','F','userdata',...
        0,'Callback',@Ficheiro);

c = uimenu('Label','Sobre');
    uimenu(c,'Label','Info do Sistema','Accelerator','I','Callback',@Info);

end
%%
%**************************************************************************
%* Função: Ficheiro
%*
%* Informação: Esta função permite carregar o ficheiro do tipo ".xls" e
%              ".xlsx" (Excel) com as medidas.
%**************************************************************************
function Ficheiro(gcbo,evnt)

global Filename Pathname

%Guarda o nome e caminho do Ficheiro Excel carregado previamente.
Filename_Old=Filename;
Pathname_Old=Pathname;

%Inibe a apresentação do Warning no ecran quando não existe nenhum ficheiro
% carregado.
warning('off', 'MATLAB:deblank:NonStringInput');

%Função que efectua o carregamento de ficheiros do tipo ".xls" e ".xlsx" e
% permite guardar a última localização de abertura do ficheiro.
[Filename, Pathname] = uigetfile( ...
        {  '*.xls','Excel 97-2003 (*.xls)'; ...
```

```
            '*.xlsx','Excel (*.xlsx)'; ...
            '*.*','Todos os Ficheiros (*.*)'}, ...
            'Escolha o ficheiro de medidas', ...
            strcat('C:\',Pathname_Old,Filename_Old));

%Apresenta o caminho completo do ficheiro carregado na janela de texto
% inferior do ecrãn de forma a saber qual o ficheiro que está a ser
% utilizado.
set(findobj('Tag','Texto_Ficheiro'),'String', ...
            strcat(Pathname,Filename));

  %Verifica se foi carregado um novo ficheiro excel, caso não tenha sido,
  % utiliza o mesmo ficheiro já existente.
  if (Filename==0)
      set(findobj('Tag','Texto_Ficheiro'),'String', ...
            strcat(Pathname_Old,Filename_Old));
       Filename=Filename_Old;
       Pathname=Pathname_Old;
  end
end
%%
%**************************************************************************
%* Função: Play
%*
%* Informação: Esta função executa o cálculo sobre o ficheiro de medidas.
%**************************************************************************
function Play(gcbo,evnt)

global Filename Pathname

%Obtém o tipo de evento do botão.
Play=get(gcbo,'userdata');

   %Caso o evento seja para a execução dos cálculos chama o respectivo
   % método. Caso contrário, termina a aplicação gráfica.
   if(Play==0)
       Descodificador_RGB_3Canais(Filename,Pathname);
   else
       clear all
       clc
       close all;
   end
end
%%
%**************************************************************************
%* Função: Info
%*
%* Informação: Esta função executa uma janela gráfica de informação do
%*             sistema.
%**************************************************************************
function Info(gcbo,evnt)

Sobre();

end
```

```
%*************************************************************************
%* Função: Descodificador_RGB_3Canais
%*
%* Informação: Função que faz o carregamento o ficheiro Excel dos dados a
%*             processar e que apresenta no ecrãn o resultado da
%*             descodificação dos 3 canais RBG.
%*             Esta função efectua o carregamento do ficheiro de calibração
%*             dos dados dos 3 canais e também dos dados a processar. É
%*             solicitada à função "Calcula_Valores" que efectue o cálculo
%*             dos valores de fotocorrente gerada com base nos tempos de
%*             bit e no tempo do início do bit e onde são retornados os
%*             valores que servirão de base para o cálculo dos respectivos
%*             patamares de sinais tendo em conta o número de combinações
%*             possíveis para os sinais em causa. Com os valores dos
%*             cálculos do ficheiro de calibração e com os valores dos
%*             dados a processar é possível efectuar cálculos de forma a
%*             obter o sinal de dados emitido pelos 3 canais RGB. Os
%*             resultados obtidos pela descodificação dos sinais são
%*             apresentados no mesmo subplot já existente de forma a ser
%*             perceptível o sinal que os gera.
%*************************************************************************
function Descodificador_RGB_3Canais(Filename,Pathname)

global Painel_Canais Eixo_Temporal Contagem Grafico_Canal_R ...
    Grafico_Canal_G Grafico_Canal_B Factor_Tempo

%Variável que indica o número de vezes que foi executado o cálculo.
Contagem = Contagem+1;

%Ficheiro_Calib - Nome do ficheiro Excel de calibração. A sequência de
% bits pode ser qualquer desde que estejam presentes as 8 combinações. O
% ficheiro deve ter três colunas: Tempo, Fotocorrente gerada pela
% influência da luz de fundo pelo lado frontal (Vpin1), Fotocorrente gerada
% pela influência da luz de fundo pelo lado posterior (Vpin2).
Ficheiro_Calib='Calibragem.xls';

%Ficheiro_Dados - Nome do ficheiro Excel que se pretende descodificar. Este
% ficheiro deve ter três colunas: Tempo, Fotocorrente gerada pela
% influência da luz de fundo pelo lado frontal (Vpin1), Fotocorrente gerada
% pela influência da luz de fundo pelo lado posterior (Vpin2).
Ficheiro_Dados=strcat(Pathname,Filename);

%Tempo de duração de cada bit.
Tempo_bit=0.1667E-3;

%Instante de tempo para o início do primeiro bit do ficheiro de
% calibração.
Inicio_bit=7.5E-5;

%Inicia o tratamento de excepções que poderão ocorrer no processamento do
% ficheiro de dados.
try

    %Efectua o carregamento dos dados do ficheiro de calibração.
    Dados=xlsread(Ficheiro_Calib);
    B=[0 1 0 1 0 1 0 1]; %Define o sinal B de dados do ficheiro de calib.
    R=[0 0 0 0 1 1 1 1]; %Define o sinal R de dados do ficheiro de calib.
```

```
G=[0 0 1 1 0 0 1 1]; %Define o sinal G de dados do ficheiro de calib.

%Normalização dos dados no tempo e em amplitude do ficheiro de
% calibração.
Dados(:,2)=Dados(:,2)-min(Dados(:,2));
Dados(:,3)=Dados(:,3)-min(Dados(:,3));
Corrente_Maxima=max(Dados(:,2));
Dados(:,2)=Dados(:,2)/Corrente_Maxima;
Dados(:,3)=Dados(:,3)/Corrente_Maxima;
Tempo=Dados(:,1)-Dados(1,1);

%Executa a função de cálculo dos valores de dados de calibração.
[Ipin1 Ipin2]=Calcula_Valores(Tempo,Inicio_bit,Tempo_bit, ...
    Dados(:,2),Dados(:,3),1);

%Calcula duas coordenadas por cada bit, uma correspondente à soma
% (Vpin1+Vpin2) e outra à diferença (Vpin1-Vpin2).
Calculo.bits=[B' R' G'];
Calculo.Ipin1=Ipin1(1:8);
Calculo.Ipin2=Ipin2(1:8);
Calculo.sum=Ipin1(1:8)+Ipin2(1:8);
Calculo.diff=Ipin1(1:8)-Ipin2(1:8);
Calculo.Ipin1;

%% Efectua o carregamento dos dados do ficheiro a descodificar.
Dados = xlsread(Ficheiro_Dados);

%Normalização dos dados no tempo e em amplitude do ficheiro de
% de dados.
Dados(:,2)=Dados(:,2)-min(Dados(:,2));
Dados(:,3)=Dados(:,3)-min(Dados(:,3));
Corrente_Maxima=max(Dados(:,2));
Dados(:,2)=Dados(:,2)/Corrente_Maxima;
Dados(:,3)=Dados(:,3)/Corrente_Maxima;
Tempo=Dados(:,1)-Dados(1,1);

%Executa a função de cálculo dos valores de dados a descodificar.
[Ipin1 Ipin2]=Calcula_Valores(Tempo,Inicio_bit,Tempo_bit, ...
    Dados(:,2),Dados(:,3),0);

%Calcula duas coordenadas por cada bit, uma correspondente à soma
% (Vpin1+Vpin2) e outra à diferença (Vpin1-Vpin2).
Data.Ipin1=Ipin1;
Data.Ipin2=Ipin2;
Data.sum=Ipin1+Ipin2;
Data.diff=Ipin1-Ipin2;

%% Define o número de combinações dos dados a descodificar que
% corresponde ao comprimento dos dados correspondentes ao valor da
% corrente do PIN1.
Num_Combinacoes=length(Ipin1);

%Define os sinais de dados de cada canal a descodificar.
B=[];R=[];G=[];

%Efectua a medição das distâncias no espaço vectorial bidimensional
% correspondente às coordenadas da soma e diferença de Vpin1 com Vpin2,
```

```
    % sendo que a medição é feita entre o símbolo que se pretende
    % descodificar e o os símbolos da sequência standard de calibração.
    % Esta medição é efectuada para cada um dos bits existentes.
    for i=1:Num_Combinacoes
        Valor=repmat([Data.sum(i) Data.diff(i)],8,1)- ...
            [Calculo.sum Calculo.diff];
        Valor=Valor*Valor';
        Valor=diag(Valor);
        Data.distances{i}=Valor;
    end

    %Efectua a verificação da distância mínima entre as coordenadas de
    % Vpin1 e Vpin2 no espaço vectorial e faz a respectiva atribuição do
    % código RGB respectivo.
    for i=1:Num_Combinacoes
        [auxmin auxpos]=min(Data.distances{i});
        Calculo.bits(auxpos,:);
        Data.bits(i,1:3)=Calculo.bits(auxpos,:);
    end

    %Faz a atribuição do bit do respectivo código RGB calculado
    % anteriormente.
    B=Data.bits(:,1);
    R=Data.bits(:,2);
    G=Data.bits(:,3);

    Valor_Maximo=1; %Define o valor máximo a aplicar ao gráfico do subplot.

    %Define as variáveis do sinal dos canais RGB para apresentar no
    % gráfico do subplot.
    Bplot=[];Rplot=[];Gplot=[];

    %Define a variável que contém o valor máximo do eixo temporal (xx)
    % para a apresentação do gráfico no subplot.
    Eixo_Temporal=[];

    %Faz a atribuição do sinal dos canais RGB para apresentar no gráfico
    % do subplot.
    for i=1:Num_Combinacoes
        Bplot=[Bplot B(i) B(i)];
        Rplot=[Rplot R(i) R(i)];
        Gplot=[Gplot G(i) G(i)];
        %Define o valor máximo do eixo temporal (xx).
        Eixo_Temporal=[Eixo_Temporal (i-1)*Tempo_bit i*Tempo_bit];
    end

    %Ajusta o sinal dos canais RGB no subplot.
    Bplot=Valor_Maximo*1.25+Valor_Maximo*0.05*Bplot;
    Rplot=Valor_Maximo*1.15+Valor_Maximo*0.05*Rplot;
    Gplot=Valor_Maximo*1.05+Valor_Maximo*0.05*Gplot;

    hold on %Retém o gráfico actual na figura já existente no subplot.

    %Verifica se é o primeiro gráfico a aparecer no subplot.
    if (Contagem==1)
        Grafico_Canal_B = plot(Eixo_Temporal*Factor_Tempo,Bplot, ...
            'Color','blue');%Mostra o gráfico do canal azul no subplot.
```

```
        hold on
        Grafico_Canal_R = plot(Eixo_Temporal*Factor_Tempo,Rplot, ...
            'Color','red');%Mostra o gráfico do canal vermelho no subplot.
        hold on
        Grafico_Canal_G = plot(Eixo_Temporal*Factor_Tempo,Gplot, ...
            'Color','green');%Mostra o gráfico do canal verde no subplot.
    else
        delete (Grafico_Canal_B);%Apaga os canal B existente no subplot.
        delete (Grafico_Canal_R);%Apaga os canal R existente no subplot.
        delete (Grafico_Canal_G);%Apaga os canal G existente no subplot.
        Grafico_Canal_B = plot(Eixo_Temporal*Factor_Tempo,Bplot, ...
            'Color','blue');
        hold on
        Grafico_Canal_R = plot(Eixo_Temporal*Factor_Tempo,Rplot, ...
            'Color','red');
        hold on
        Grafico_Canal_G = plot(Eixo_Temporal*Factor_Tempo,Gplot, ...
            'Color','green');
    end

    %Define o sistema de eixos do gráfico no subplot.
    axis([0 (max(Eixo_Temporal*Factor_Tempo)) 0 (max(Bplot*1.05))]);
    xlabel('Tempo (us)')
    ylabel('Fotocorrente (uA)')

    %Mostra o sinal de dados do canal R descodificado.
    Sinal_B = uicontrol('Parent',Painel_Canais,'Style', ...
        'Text','Position',[20,44,580,20],'FontWeight','bold',...
        'HorizontalAlignment','left',...
        'FontSize',12,'ForegroundColor',...
        'blue','BackgroundColor','white',...
        'Tag','Sinal_B','String',strcat('Canal B: ' ,num2str(B')));

    %Mostra o sinal de dados do canal R descodificado.
    Sinal_R = uicontrol('Parent',Painel_Canais,'Style', ...
        'Text','Position',[20,26,580,20],'FontWeight','bold',...
        'HorizontalAlignment','left',...
        'FontSize',12,'ForegroundColor',...
        'red','BackgroundColor','white',...
        'Tag','Sinal_R','String',strcat('Canal R: ' ,num2str(R')));

    %Mostra o sinal de dados do canal G descodificado.
    Sinal_G = uicontrol('Parent',Painel_Canais,'Style', ...
        'Text','Position',[20,4,580,20],'FontWeight','bold',...
        'HorizontalAlignment','left',...
        'FontSize',12,'ForegroundColor',...
        'green','BackgroundColor','white',...
        'Tag','Sinal_G','String',strcat('Canal G: ' ,num2str(G')));

    %Devolve uma mensagem de erro caso aconteça algum erro no processamento
    % do ficheiro.
    catch
        ['Não foi possível encontrar o ficheiro de dados. ' ...
          'Deve carregar o ficheiro de dados antes de efectuar o cálculo.']
    end
end
```

```
%*************************************************************************
%* Função: Calcula_Valores
%*
%* Informação: Função que efectua os cálculos relativos aos valores de
%*             fotocorrente carregados pelo ficheiro excel. Esta função faz
%*             o ajuste do bit inicial do sinal a descodificar e vai
%*             calcular o número de combinações que tem a descodificar.
%*             Com base nos valores de fotocorrente recebida pela função
%*             são feitos os cálculos necessários de forma a serem obtidos
%*             os valores gerados pela influência da luz de fundo pelo lado
%*             frontal e pelo lado posterior do dispositivo fotodetector.
%*             Os respectivos gráficos são apresentados no subplot já
%*             existente no ecrãn de forma a serem perceptíveis os sinais
%*             de fotocorrente gerada pelo dispositivo semicondutor e
%*             medidos quando é aplicada uma luz de fundo pelo lado frontal
%*             (Vpin1) e os valores de fotocorrente gerada  pela influência
%*             da luz de fundo pelo lado posterior (Vpin2).
%*************************************************************************
function [Iinv Idir]=Calcula_Valores(Tempo, Inicio_bit, Tempo_bit, ...
    Idatainv, Idatadir, Calibragem)

global Painel_Central Contagem Grafico_Idatainv Grafico_Idatadir ...
    Factor_Tempo Filename

    %Variáveis dos cálculos temporais para o ajuste do bit inicial.
    Tempo=Tempo-Inicio_bit;
    Tempo_Seguinte=find(Tempo>0);
    Tempo=Tempo(Tempo_Seguinte);
    Idatainv=Idatainv(Tempo_Seguinte);
    Idatadir=Idatadir(Tempo_Seguinte);

    %Encontra o número de combinações (bits) dos dados a descodificar.
    Num_Combinacoes=round(Tempo(end)/Tempo_bit);

    %Define as variáveis para os cálculos da corrente directa e corrente
    % inversa.
    Idir=[];
    Iinv=[];

    %Verifica se existe um ficheiro de dados carregado para processar.
    if (Filename~=0)
        %Cria um subplot no gráfico central para a apresentação dos
        % gráficos a processar.
        Grafico_Central = subplot(1,1,1,'Parent',Painel_Central);
        %Verifica se é a primeira execução do gráfico e se o gráfico
        % a apresentar é diferente do de calibragem.
        if (Contagem==1 && Calibragem==0)
            Grafico_Idatainv = plot(Tempo*Factor_Tempo, ...
                Idatainv, 'k--','Color','magenta');
        end
        if (Contagem~=1 && Calibragem==0)
            delete (Grafico_Idatainv)
            Grafico_Idatainv = plot(Tempo*Factor_Tempo, ...
                Idatainv, 'k--','Color','magenta');
        end

        hold on %Retém o gráfico actual na figura já existente no subplot.
```

```
        if (Contagem==1 && Calibragem==0)
            Grafico_Idatadir = plot(Tempo*Factor_Tempo, ...
                Idatadir,'k-');
        end
        if (Contagem~=0 && Calibragem==0)
            delete (Grafico_Idatadir)
            Grafico_Idatadir = plot(Tempo*Factor_Tempo, ...
                Idatadir,'k-');
        end
    end

    %Efectua o varrimento tendo em conta o número de combinações (bits).
    for i=1:Num_Combinacoes
        %Efectua a intersecção entre os valores iniciais e finais de cada
        % bit de forma a contabilizar apenas as amostras desse bit.
        index=intersect(find(Tempo<Tempo_bit*i-Tempo_bit*0.3),...
            find(Tempo>(i-1)*Tempo_bit+Tempo_bit*0.3));
        %Calcula a média de valores para cada bit da medição.
        Idir(end+1,1)=mean(Idatadir(index));
        Iinv(end+1,1)=mean(Idatainv(index));
    end
end
```

```
%**************************************************************************
%* Função: Sobre
%*
%* Informação: Função que mostra uma janela de descrição do autor do
%*             projecto, com uma pequena imagem gráfica de fundo.
%**************************************************************************
function varargout = Sobre(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Sobre_OpeningFcn, ...
                   'gui_OutputFcn',  @Sobre_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
    if nargin && ischar(varargin{1})
        gui_State.gui_Callback = str2func(varargin{1});
    end

    if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
    else
        gui_mainfcn(gui_State, varargin{:});
    end
end
%%
%**************************************************************************
%* Função: Sobre_OpeningFcn
%*
%* Informação: Esta função inicializa a janela gráfica da descrição Sobre.
%**************************************************************************
function Sobre_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;
guidata(hObject, handles);

%Coloca a Janela no Centro do Ecran
FigPos=get(0,'DefaultFigurePosition');
OldUnits = get(hObject, 'Units');
set(hObject, 'Units', 'pixels');
OldPos = get(hObject,'Position');
FigWidth = OldPos(3);
FigHeight = OldPos(4);

if isempty(gcbf)
    ScreenUnits=get(0,'Units');
    set(0,'Units','pixels');
    ScreenSize=get(0,'ScreenSize');
    set(0,'Units',ScreenUnits);

    FigPos(1)=1/2*(ScreenSize(3)-FigWidth);%%1/2 é metade do ecran
    FigPos(2)=1/2*(ScreenSize(4)-FigHeight);%%1/2 é metade do ecran
else
    GCBFOldUnits = get(gcbf,'Units');
    set(gcbf,'Units','pixels');
    GCBFPos = get(gcbf,'Position');
    set(gcbf,'Units',GCBFOldUnits);
```

```
    FigPos(1:2) = [(GCBFPos(1) + GCBFPos(3) / 2) - FigWidth / 2, ...
                   (GCBFPos(2) + GCBFPos(4) / 2) - FigHeight / 2];
end
FigPos(3:4)=[FigWidth FigHeight];
set(hObject, 'Position', FigPos);
set(hObject, 'Units', OldUnits);

[data, map]=imread('Fundo.gif');
pic=ind2rgb(data,map);
Img=image(pic, 'Parent', handles.Imagem);

set(handles.Imagem, ...
    'Visible', 'off', ...
    'YDir'   , 'reverse'       , ...
    'XLim'   , get(Img,'XData'), ...
    'YLim'   , get(Img,'YData')  ...
    );

% Coloca a Janela como Fixa
set(handles.output,'WindowStyle','modal')
end
%%
%**************************************************************************
%* Função: Sobre_OutputFcn
%*
%* Informação: Esta função mostra a janela gráfica da descrição Sobre.
%**************************************************************************
function varargout = Sobre_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

end
```