```
%*************************************************************************************************************
%                                                                                                            %
%                                                Demultiplexor                                               %
%                                                                                                            %
%*************************************************************************************************************

%Parámetros de entrada
%   No recibe parámetros de entrada

%Parámetros de salida
%   No hay parámetros de salida. Crea un fichero .mat con la información demultiplexada

function demux_video()

%Abre fichero
nombre_archivo=input('Teclee el nombre del archivo a demultiplexar\n','s');
fid=fopen(nombre_archivo);
a=fread(fid,inf,'uint8=>uint8');

tam_archivo=size(a);

%Inicialización de índices
indice_actual=1;
indice_siguiente=1;
indice_video=1;

%Tabla con valores de código de comienzo
table_stream_id

while(indice_actual<tam_archivo(1)-3)    %Hasta el final del archivo
    if((a(indice_actual+3)>=id_13818_2(1))&(a(indice_actual+3)<=id_13818_2(2)))
        num_bytes=4;
        %packet_start_code
        indice_actual=indice_actual+num_bytes;

        num_bytes=2;
        PES_packet_length_aux(1)=a(indice_actual);
        PES_packet_length_aux(2)=a(indice_actual+1);

        PES_packet_length=double(PES_packet_length_aux(1))*(2^8)+double(PES_packet_length_aux(2));

        indice_actual=indice_actual+num_bytes;

        num_bytes=PES_packet_length;

        indice_siguiente=next_start_code(a,indice_actual);

        if((a(indice_siguiente+3)>=0)&(a(indice_siguiente+3)<=184))
            num_bytes=num_bytes-(indice_siguiente-indice_actual);
            indice_actual=indice_siguiente;
            info_video(indice_video:indice_video+num_bytes-1,1)=a(indice_actual:...
                indice_actual+num_bytes-1);
            indice_video=indice_video+num_bytes;
            indice_actual=indice_actual+num_bytes;
        end
    else
        indice_siguiente=next_start_code(a,indice_actual+1);
        indice_actual=indice_siguiente;
    end
end
save video.mat info_video
```

```
%*************************************************************************************************
%                                          Video sequence                                        %
%                                                                          Nº of bits  Mnemonic   %
%   video_sequence() {                                                                            %
%       next_start_code()                                                                         %
%       sequence_header()                                                                         %
%       if(nextbits()==extension_start_code) {                                                    %
%           sequence_extension()                                                                  %
%           do {                                                                                  %
%               extension_and_user_data(0)                                                        %
%               do {                                                                              %
%                   if(nextbits()==group_start_code) {                                            %
%                       group_of_pictures_header()                                                %
%                       extension_and_user_data(1)                                                %
%                   }                                                                             %
%                   picture_header()                                                              %
%                   picture_coding_extension()                                                    %
%                   extension_and_user_data(2)                                                    %
%                   picture_data()                                                                %
%               } while ((nextbits()==picture_start_code)||(nextbits()==group_start_code))        %
%               if(nextbits()!=sequence_end_code) {                                               %
%                   sequence_header()                                                             %
%                   sequence_extension()                                                          %
%               }                                                                                 %
%           } while(nextbits()!=sequence_end_code)                                                %
%       }else {                                                                                   %
%           /* ISO/IEC 11172-2 */                                                                %
%       }                                                                                         %
%       sequene_end_code                                                    32      bslbf         %
%   }                                                                                             %
%*************************************************************************************************

clear all
close all

%**************************************************************************
%ABRIR ARCHIVO E INICIALIZAR INDICES
function video_sequence()

archivo=input('Teclee 1 si el archivo contiene audio y video\nTeclee 0 si el archivo solo contiene el video\n','s');

if(archivo=='0')                        %si es 0 abre el arcivo de video
    nombre_archivo=input('Teclee el nombre del archivo con el video a decodificar\n','s');
    fid=fopen(nombre_archivo);
    a=fread(fid,inf,'uint8=>uint8');    %vector columna
else
    load ..\demux\video.mat             %si es 1 carga el archivo demultiplexado
    a=info_video;                       %vector columna
end

tam_archivo=size(a);                    %Nx1
table_stream_id;

%Inicialización índices
indice_actual=1;
indice_siguiente=1;
%Inicialización imagenes de referencia y banderas
im_referencia=0;
flag_gop=0;

%**************************************************************************
%SITUARSE AL COMIENZO DE LA INFO DE VIDEO

start_code_value=sequence_header_code;
%next_start_code()
indice_buscado=next_start_code_value(a,indice_actual,start_code_value);
%Posición del código de comienzo buscado

if(indice_buscado<tam_archivo(1)-3)
    indice_actual=indice_buscado;
end

indice_siguiente=next_start_code(a,indice_actual+4);
num_bytes=indice_siguiente-indice_actual;

%**************************************************************************
%sequence_header
aux=a(indice_actual:indice_actual+num_bytes-1);
[horizontal_size_value,vertical_size_value,intra_quantiser_matrix,...
    non_intra_quantiser_matrix]=sequence_header(aux);

indice_actual=indice_siguiente;

%if(nextbits()==extension_start_code)
```

```
if(a(indice_actual+3)==extension_start_code)

    indice_siguiente=next_start_code(a,indice_actual+4);
    num_bytes=indice_siguiente-indice_actual;

    %sequence_extension()
    aux=a(indice_actual:indice_actual+num_bytes-1);
    [progressive_sequence,chroma_format,horizontal_size_extension,...
        vertical_size_extension]=sequence_extension(aux);

    %Calculo de variables necesarias
    horizontal_size_bin=[dec2bin(horizontal_size_extension,2),dec2bin(horizontal_size_value,12)];
    horizontal_size=uint16(bin2dec(horizontal_size_bin));
    mb_width=floor((horizontal_size+15)/16);        %numero de macrobloques en sentido horizontal

    vertical_size_bin=[dec2bin(vertical_size_extension,2),dec2bin(vertical_size_value,12)];
    vertical_size=uint16(bin2dec(vertical_size_bin));
    if(progressive_sequence==1)
        mb_height=floor((vertical_size+15)/16);     %numero de macrobloques en sentido vertical
    else
        mb_height=2*floor((vertical_size+31)/32);
    end

    indice_actual=indice_siguiente;

    %do{
    %extension_and_user_data(0)
    [indice_actual]=extension_and_user_data(0,a,indice_actual);

    %do{
    %if(nextbits()==group_start_code)
    if(a(indice_actual+3)==group_start_code)

       indice_siguiente=next_start_code(a,indice_actual+4);
       num_bytes=indice_siguiente-indice_actual;

       %group_of_pictures_header
       aux=a(indice_actual:indice_actual+num_bytes-1);
       group_of_pictures_header(aux);

       num_GOP=1;       %Contador de GOPs para reordenar tramas

       indice_actual=indice_siguiente;

       %extension_and_user_data(1)
       [indice_actual]=extension_and_user_data(1,a,indice_actual);
    end

    indice_siguiente=next_start_code(a,indice_actual+4);
    num_bytes=indice_siguiente-indice_actual;

    %picture_header()
    aux=a(indice_actual:indice_actual+num_bytes-1);
    [temporal_reference,picture_coding_type]=picture_header(aux);

    num_frame_in_GOP=temporal_reference+1; %Contador de tramas en GOPs para reordenar tramas

    indice_actual=indice_siguiente;

    indice_siguiente=next_start_code(a,indice_actual+4);
    num_bytes=indice_siguiente-indice_actual;

    %picture_coding_extension()
    aux=a(indice_actual:indice_actual+num_bytes-1);
    [picture_structure,frame_pred_frame_dct,concealment_motion_vectors,...
        f_code,precision,alternate_scan,q_scale_type,intra_vlc_format]=picture_coding_extension(aux);

    indice_actual=indice_siguiente;

    %extension_and_user_data(2)
    [indice_actual]=extension_and_user_data(2,a,indice_actual);

    indice_siguiente=next_start_code(a,indice_actual+4);
    num_bytes=indice_siguiente-indice_actual;

    %picture_data()
    %*************************************************************************************************
    %                                              Picure_data                                       %
    %                                                                    Nº of bits  Mnemonic        %
    %   picture_data() {                                                                             %
    %       do {                                                                                     %
    %           slice()                                                                              %
    %       } while (nextbits()==slice_start_code                                                    %
    %       next_start_code()                                                                        %
    %   }                                                                                            %
```

```
%*************************************************************************************************************

%do{
aux=a(indice_actual:indice_actual+num_bytes-1+3);   %Se pasan 3 bytes pertenecientes a las siguiente
                                                    %estructura para determinar fin de slice
    %slice()
[slice_luma,slice_chroma_b,slice_chroma_r,mb_row,mb_column]=slice(aux,horizontal_size, ...
    vertical_size,mb_width,picture_coding_type,picture_structure,frame_pred_frame_dct, ...
    concealment_motion_vectors,chroma_format,f_code,precision,alternate_scan, ...
    intra_quantiser_matrix,non_intra_quantiser_matrix,q_scale_type,intra_vlc_format,im_referencia);

pos_y=(mb_row-1)*16+1;      %Posición del primer pixel de cada slice

%Formación de imágenes
pict_luma(pos_y:pos_y+15,1:mb_column*16)=slice_luma;
pict_chroma_b(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_b;
pict_chroma_r(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_r;

indice_actual=indice_siguiente;
%}while(nextbits()==slice_start_code
while((a(indice_actual+3)>=slice_start_code(1)) & (a(indice_actual+3)<=slice_start_code(2)))

    indice_siguiente=next_start_code(a,indice_actual+4);
    num_bytes=indice_siguiente-indice_actual;

    aux=a(indice_actual:indice_actual+num_bytes-1+3);
    [slice_luma,slice_chroma_b,slice_chroma_r,mb_row,mb_column]=slice(aux,horizontal_size, ...
        vertical_size,mb_width,picture_coding_type,picture_structure,frame_pred_frame_dct, ...
        concealment_motion_vectors,chroma_format,f_code,precision,alternate_scan, ...
        intra_quantiser_matrix,non_intra_quantiser_matrix,q_scale_type,intra_vlc_format,im_referencia);
    indice_actual=indice_siguiente;

    pos_y=(mb_row-1)*16+1;  %Posición del primer pixel de cada slice

    %Formación de imagenes
    pict_luma(pos_y:pos_y+15,1:mb_column*16)=slice_luma;
    pict_chroma_b(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_b;
    pict_chroma_r(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_r;
end

%Matriz con las 3 componentes (luma, croma azul y croma roja)
YCbCr(:,:,1)=(pict_luma);
YCbCr(:,:,2)=(pict_chroma_b);
YCbCr(:,:,3)=(pict_chroma_r);

YCbCr=double(YCbCr);    %Necesario para la siguiente función

%Matriz con las 2 componentes de color
RGB=colorspace('YCbCr->RGB',double(YCbCr));

num_frame=num_frame_in_GOP; %Es el primer GOP, no hay que calcular nada más

figure(num_frame);
image(RGB)

M(num_frame)=getframe;  %Formación de movie frame

%Almacenamiento de predicciones
if(picture_coding_type==1) %Imagen tipo I
    frame_pred_I=YCbCr;
end

primera_P=0;

%}while((nextbits()==picture_start_code)||(nextbits()==group_start_code
while(((a(indice_actual+3)==picture_start_code)|(a(indice_actual+3)==group_start_code)))
    %if(nextbits()==group_start_code)
    if(a(indice_actual+3)==group_start_code)

        indice_siguiente=next_start_code(a,indice_actual+4);
        num_bytes=indice_siguiente-indice_actual;

        %group_of_pictures_header
        aux=a(indice_actual:indice_actual+num_bytes-1);
        group_of_pictures_header(aux);

        flag_gop=1;         %Indica que no es el primer GOP
        num_GOP=num_GOP+1;  %Contador de GOPs para reordenar tramas
        numero_frames_in_GOP=num_frame_P+2; %Numero total de frames en un GOP

        indice_actual=indice_siguiente;

        %extension_and_user_data(1)
        [indice_actual]=extension_and_user_data(1,a,indice_actual);
    end
```

```
indice_siguiente=next_start_code(a,indice_actual+4);
num_bytes=indice_siguiente-indice_actual;

%picture_header()
aux=a(indice_actual:indice_actual+num_bytes-1);
[temporal_reference,picture_coding_type]=picture_header(aux);
num_frame_in_GOP=temporal_reference+1;

%Matrices para predicciones
if(picture_coding_type==1)  %Imagen tipo I
    im_referencia=0;        %No hay predicción
end

if(picture_coding_type==2)  %Imagen tipo P
    if(primera_P==0)
        im_referencia(1:240,1:320,1:3,1)=frame_pred_I;  %Predicción forward de tipo I
        im_referencia=int16(im_referencia);
    else
        im_referencia(1:240,1:320,1:3,1)=frame_pred_P;  %Predicción forward de tipo P
        im_referencia=int16(im_referencia);
    end
    num_frame_P=num_frame_in_GOP;
end

if(picture_coding_type==3)  %Imagen tipo P
    if(primera_P==0)
        im_referencia(1:240,1:320,1:3,1)=frame_pred_I;  %Prediccion forward de tipo I
        im_referencia=int16(im_referencia);
    else
        im_referencia(1:240,1:320,1:3,1)=frame_pred_P;  %Prediccion forward de tipo P
        im_referencia=int16(im_referencia);
    end
    im_referencia(1:240,1:320,1:3,2)=frame_pred_P;      %Prediccion backward de tipo P
    im_referencia=int16(im_referencia);
end

indice_actual=indice_siguiente;

indice_siguiente=next_start_code(a,indice_actual+4);
num_bytes=indice_siguiente-indice_actual;

%picture_coding_extension()
aux=a(indice_actual:indice_actual+num_bytes-1);
[picture_structure,frame_pred_frame_dct,concealment_motion_vectors,f_code,precision,...
    alternate_scan,q_scale_type,intra_vlc_format]=picture_coding_extension(aux);

indice_actual=indice_siguiente;

%extension_and_user_data(2)
[indice_actual]=extension_and_user_data(2,a,indice_actual);

indice_siguiente=next_start_code(a,indice_actual+4);
num_bytes=indice_siguiente-indice_actual;

%picture_data()
%*************************************************************************************************
%                                              Picure_data                                       %
%                                                                  Nº of bits  Mnemonic          %
%   picture_data() {                                                                             %
%       do {                                                                                     %
%           slice()                                                                              %
%       } while (nextbits()==slice_start_code                                                    %
%       next_start_code()                                                                        %
%   }                                                                                            %
%*************************************************************************************************

%do{
    %slice()

aux=a(indice_actual:indice_actual+num_bytes-1+3);   %Se pasan 3 bytes pertenecientes a las siguiente
                                                    %estructura para determinar fin de slice
[slice_luma,slice_chroma_b,slice_chroma_r,mb_row,mb_column]=slice(aux,horizontal_size,...
    vertical_size,mb_width,picture_coding_type,picture_structure,frame_pred_frame_dct,...
    concealment_motion_vectors,chroma_format,f_code,precision,alternate_scan,...
    intra_quantiser_matrix,non_intra_quantiser_matrix,q_scale_type,intra_vlc_format,im_referencia);

pos_y=(mb_row-1)*16+1;      %Posición del primer pixel de cada slice

%Formación de imagenes
pict_luma(pos_y:pos_y+15,1:mb_column*16)=slice_luma;
pict_chroma_b(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_b;
pict_chroma_r(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_r;

indice_actual=indice_siguiente;
```

```
    %}while(nextbits()==slice_start_code
    while((a(indice_actual+3)>=1) & (a(indice_actual+3)<=175))

        indice_siguiente=next_start_code(a,indice_actual+4);
        num_bytes=indice_siguiente-indice_actual;

        aux=a(indice_actual:indice_actual+num_bytes-1+3);
        [slice_luma,slice_chroma_b,slice_chroma_r,mb_row,mb_column]=slice(aux,horizontal_size, ...
            vertical_size,mb_width,picture_coding_type,picture_structure,frame_pred_frame_dct, ...
            concealment_motion_vectors,chroma_format,f_code,precision,alternate_scan, ...
            intra_quantiser_matrix,non_intra_quantiser_matrix,q_scale_type,intra_vlc_format, ...
            im_referencia);

        pos_y=(mb_row-1)*16+1   %Posición del primer pixel de cada slice

        %Formacion de imagenes
        pict_luma(pos_y:pos_y+15,1:mb_column*16)=slice_luma;
        pict_chroma_b(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_b;
        pict_chroma_r(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_r;

        indice_actual=indice_siguiente;
    end


    %Matriz con las 3 componentes (luma, croma azul y croma roja)
    YCbCr(:,:,1)=pict_luma;
    YCbCr(:,:,2)=pict_chroma_b;
    YCbCr(:,:,3)=pict_chroma_r;

    YCbCr=double(YCbCr);    %Necesario para la siguiente función

    %Matriz con las 2 componentes de color
    RGB=colorspace('YCbCr->RGB',YCbCr);
    if(flag_gop==1)
        num_frame=num_frame_in_GOP+num_frame_P+numero_frames_in_GOP*(num_GOP-1);
    else
        num_frame=num_frame_in_GOP;
    end

    figure(num_frame)
    image(RGB)

    M(num_frame)=getframe;  %Formación de movie frame

    %Almacenamiento de predicciones
    if(picture_coding_type==1)  %Imagen tipo I
        frame_pred_I=YCbCr;
    end
    if(picture_coding_type==2)  %Imagen tipo P
        primera_P=1;
        frame_pred_P=YCbCr;
    end

end
a(indice_actual+3)=183;
%if(next_bits()!=sequence_end_code)
if(a(indice_actual+3)~=sequence_end_code)

    indice_siguiente=next_start_code(a,indice_actual+4);
    num_bytes=indice_siguiente-indice_actual;

    %sequence_header()
    aux=a(indice_actual:indice_actual+num_bytes-1);
    [horizontal_size_value,vertical_size_value]=sequence_header(aux);

    indice_actual=indice_siguiente;

    %sequence_extension()
    [indice_actual]=extension_and_user_data(0,nombre_archivo,indice_actual);
end



%extension_and_user_data(0)
[indice_actual]=extension_and_user_data(0,a,indice_actual);

%do{
%if(nextbits()==group_start_code)
if(a(indice_actual+3)==group_start_code)

   indice_siguiente=next_start_code(a,indice_actual+4);
   num_bytes=indice_siguiente-indice_actual;

   %group_of_pictures_header
```

```
   aux=a(indice_actual:indice_actual+num_bytes-1);
   group_of_pictures_header(aux);

   num_GOP=1;       %Contador de GOPs para reordenar tramas

   indice_actual=indice_siguiente;

   %extension_and_user_data(1)
   [indice_actual]=extension_and_user_data(1,a,indice_actual);
end

indice_siguiente=next_start_code(a,indice_actual+4);
num_bytes=indice_siguiente-indice_actual;

%picture_header()
aux=a(indice_actual:indice_actual+num_bytes-1);
[temporal_reference,picture_coding_type]=picture_header(aux);

num_frame_in_GOP=temporal_reference+1; %Contador de tramas en GOPs para reordenar tramas

indice_actual=indice_siguiente;

indice_siguiente=next_start_code(a,indice_actual+4);
num_bytes=indice_siguiente-indice_actual;

%picture_coding_extension()
aux=a(indice_actual:indice_actual+num_bytes-1);
[picture_structure,frame_pred_frame_dct,concealment_motion_vectors,...
    f_code,precision,alternate_scan,q_scale_type,intra_vlc_format]=picture_coding_extension(aux);

indice_actual=indice_siguiente;

%extension_and_user_data(2)
[indice_actual]=extension_and_user_data(2,a,indice_actual);

indice_siguiente=next_start_code(a,indice_actual+4);
num_bytes=indice_siguiente-indice_actual;

%picture_data()
%*************************************************************************************************
%                                          Picure_data                                           %
%                                                                  Nº of bits  Mnemonic    %
%   picture_data() {                                                                             %
%       do {                                                                                     %
%           slice()                                                                              %
%       } while (nextbits()==slice_start_code                                                    %
%       next_start_code()                                                                        %
%   }                                                                                            %
%*************************************************************************************************

%do{
aux=a(indice_actual:indice_actual+num_bytes-1+3);   %Se pasan 3 bytes pertenecientes a las siguiente
                                                    %estructura para determinar fin de slice
    %slice()
[slice_luma,slice_chroma_b,slice_chroma_r,mb_row,mb_column]=slice(aux,horizontal_size,...
    vertical_size,mb_width,picture_coding_type,picture_structure,frame_pred_frame_dct,...
    concealment_motion_vectors,chroma_format,f_code,precision,alternate_scan,...
    intra_quantiser_matrix,non_intra_quantiser_matrix,q_scale_type,intra_vlc_format,im_referencia);

pos_y=(mb_row-1)*16+1;      %Posición del primer pixel de cada slice

%Formación de imágenes
pict_luma(pos_y:pos_y+15,1:mb_column*16)=slice_luma;
pict_chroma_b(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_b;
pict_chroma_r(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_r;

indice_actual=indice_siguiente;
%}while(nextbits()==slice_start_code
while((a(indice_actual+3)>=slice_start_code(1)) & (a(indice_actual+3)<=slice_start_code(2)))

    indice_siguiente=next_start_code(a,indice_actual+4);
    num_bytes=indice_siguiente-indice_actual;

    aux=a(indice_actual:indice_actual+num_bytes-1+3);
    [slice_luma,slice_chroma_b,slice_chroma_r,mb_row,mb_column]=slice(aux,horizontal_size,...
        vertical_size,mb_width,picture_coding_type,picture_structure,frame_pred_frame_dct,...
        concealment_motion_vectors,chroma_format,f_code,precision,alternate_scan,...
        intra_quantiser_matrix,non_intra_quantiser_matrix,q_scale_type,intra_vlc_format,...
        im_referencia);
    indice_actual=indice_siguiente;

    pos_y=(mb_row-1)*16+1;  %Posición del primer pixel de cada slice

    %Formación de imagenes
    pict_luma(pos_y:pos_y+15,1:mb_column*16)=slice_luma;
```

```
    pict_chroma_b(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_b;
    pict_chroma_r(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_r;
end

%Matriz con las 3 componentes (luma, croma azul y croma roja)
YCbCr(:,:,1)=(pict_luma);
YCbCr(:,:,2)=(pict_chroma_b);
YCbCr(:,:,3)=(pict_chroma_r);

YCbCr=double(YCbCr);    %Necesario para la siguiente función

%Matriz con las 2 componentes de color
RGB=colorspace('YCbCr->RGB',double(YCbCr));

num_frame=num_frame_in_GOP; %Es el primer GOP, no hay que calcular nada más

figure(num_frame);
image(RGB)

M(num_frame)=getframe;  %Formación de movie frame

%Almacenamiento de predicciones
if(picture_coding_type==1) %Imagen tipo I
    frame_pred_I=YCbCr;
end

primera_P=0;

%}while((nextbits()==picture_start_code)||(nextbits()==group_start_code
while(((a(indice_actual+3)==picture_start_code)|(a(indice_actual+3)==group_start_code)))
    %if(nextbits()==group_start_code)
    if(a(indice_actual+3)==group_start_code)

        indice_siguiente=next_start_code(a,indice_actual+4);
        num_bytes=indice_siguiente-indice_actual;

        %group_of_pictures_header
        aux=a(indice_actual:indice_actual+num_bytes-1);
        group_of_pictures_header(aux);

        flag_gop=1;         %Indica que no es el primer GOP
        num_GOP=num_GOP+1;  %Contador de GOPs para reordenar tramas
        numero_frames_in_GOP=num_frame_P+2; %Numero total de frames en un GOP

        indice_actual=indice_siguiente;

        %extension_and_user_data(1)
        [indice_actual]=extension_and_user_data(1,a,indice_actual);
    end

    indice_siguiente=next_start_code(a,indice_actual+4);
    num_bytes=indice_siguiente-indice_actual;

    %picture_header()
    aux=a(indice_actual:indice_actual+num_bytes-1);
    [temporal_reference,picture_coding_type]=picture_header(aux);
    num_frame_in_GOP=temporal_reference+1;

    %Matrices para predicciones
    if(picture_coding_type==1)  %Imagen tipo I
        im_referencia=0;        %No hay predicción
    end

    if(picture_coding_type==2)  %Imagen tipo P
        if(primera_P==0)
            im_referencia(1:240,1:320,1:3,1)=frame_pred_I;  %Predicción forward de tipo I
            im_referencia=int16(im_referencia);
        else
            im_referencia(1:240,1:320,1:3,1)=frame_pred_P;  %Predicción forward de tipo P
            im_referencia=int16(im_referencia);
        end
        num_frame_P=num_frame_in_GOP;
    end

    if(picture_coding_type==3)  %Imagen tipo P
        if(primera_P==0)
            im_referencia(1:240,1:320,1:3,1)=frame_pred_I;  %Prediccion forward de tipo I
            im_referencia=int16(im_referencia);
        else
            im_referencia(1:240,1:320,1:3,1)=frame_pred_P;  %Prediccion forward de tipo P
            im_referencia=int16(im_referencia);
        end
        im_referencia(1:240,1:320,1:3,2)=frame_pred_P;      %Prediccion backward de tipo P
        im_referencia=int16(im_referencia);
    end
```

```
indice_actual=indice_siguiente;

indice_siguiente=next_start_code(a,indice_actual+4);
num_bytes=indice_siguiente-indice_actual;

%picture_coding_extension()
aux=a(indice_actual:indice_actual+num_bytes-1);
[picture_structure,frame_pred_frame_dct,concealment_motion_vectors,f_code,precision, ...
    alternate_scan,q_scale_type,intra_vlc_format]=picture_coding_extension(aux);

indice_actual=indice_siguiente;

%extension_and_user_data(2)
[indice_actual]=extension_and_user_data(2,a,indice_actual);

indice_siguiente=next_start_code(a,indice_actual+4);
num_bytes=indice_siguiente-indice_actual;

%picture_data()
%*************************************************************************************************
%                                          Picure_data                                           %
%                                                                    Nº of bits  Mnemonic        %
%   picture_data() {                                                                             %
%       do {                                                                                     %
%           slice()                                                                              %
%       } while (nextbits()==slice_start_code                                                    %
%       next_start_code()                                                                        %
%   }                                                                                            %
%*************************************************************************************************

%do{
    %slice()

aux=a(indice_actual:indice_actual+num_bytes-1+3);   %Se pasan 3 bytes pertenecientes a las siguiente
                                                    %estructura para determinar fin de slice
[slice_luma,slice_chroma_b,slice_chroma_r,mb_row,mb_column]=slice(aux,horizontal_size, ...
    vertical_size,mb_width,picture_coding_type,picture_structure,frame_pred_frame_dct, ...
    concealment_motion_vectors,chroma_format,f_code,precision,alternate_scan,...
    intra_quantiser_matrix,non_intra_quantiser_matrix,q_scale_type,intra_vlc_format, ...
    im_referencia);

pos_y=(mb_row-1)*16+1;      %Posición del primer pixel de cada slice

%Formación de imagenes
pict_luma(pos_y:pos_y+15,1:mb_column*16)=slice_luma;
pict_chroma_b(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_b;
pict_chroma_r(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_r;

indice_actual=indice_siguiente;

%}while(nextbits()==slice_start_code
while((a(indice_actual+3)>=1) & (a(indice_actual+3)<=175))

    indice_siguiente=next_start_code(a,indice_actual+4);
    num_bytes=indice_siguiente-indice_actual;

    aux=a(indice_actual:indice_actual+num_bytes-1+3);
    [slice_luma,slice_chroma_b,slice_chroma_r,mb_row,mb_column]=slice(aux,horizontal_size, ...
        vertical_size,mb_width,picture_coding_type,picture_structure,frame_pred_frame_dct, ...
        concealment_motion_vectors,chroma_format,f_code,precision,alternate_scan, ...
        intra_quantiser_matrix,non_intra_quantiser_matrix,q_scale_type,intra_vlc_format, ...
        im_referencia);

    pos_y=(mb_row-1)*16+1   %Posición del primer pixel de cada slice

    %Formacio de imagenes
    pict_luma(pos_y:pos_y+15,1:mb_column*16)=slice_luma;
    pict_chroma_b(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_b;
    pict_chroma_r(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_r;

    indice_actual=indice_siguiente;
end


%Matriz con las 3 componentes (luma, croma azul y croma roja)
YCbCr(:,:,1)=pict_luma;
YCbCr(:,:,2)=pict_chroma_b;
YCbCr(:,:,3)=pict_chroma_r;

YCbCr=double(YCbCr);    %Necesario para la siguiente función

%Matriz con las 2 componentes de color
RGB=colorspace('YCbCr->RGB',YCbCr);
if(flag_gop==1)
```

```
        num_frame=num_frame_in_GOP+num_frame_P+numero_frames_in_GOP*(num_GOP-1);
    else
        num_frame=num_frame_in_GOP;
    end

    figure(num_frame)
    image(RGB)

    M(num_frame)=getframe;  %Formación de movie frame

    %Almacenamiento de predicciones
    if(picture_coding_type==1)  %Imagen tipo I
        frame_pred_I=YCbCr;
    end
    if(picture_coding_type==2)  %Imagen tipo P
        primera_P=1;
        frame_pred_P=YCbCr;
    end

end

%if(next_bits()!=sequence_end_code)
if(a(indice_actual+3)~=sequence_end_code)

    indice_siguiente=next_start_code(a,indice_actual+4);
    num_bytes=indice_siguiente-indice_actual;

    %sequence_header()
    aux=a(indice_actual:indice_actual+num_bytes-1);
    [horizontal_size_value,vertical_size_value]=sequence_header(aux);

    indice_actual=indice_siguiente;

    %sequence_extension()
    [indice_actual]=extension_and_user_data(0,nombre_archivo,indice_actual);
end

%while(nextbits()!=sequence_end_code)
while(a(indice_actual+3)~=183)
    %extension_and_user_data(0)
[indice_actual]=extension_and_user_data(0,a,indice_actual);

%do{
%if(nextbits()==group_start_code)
if(a(indice_actual+3)==group_start_code)

   indice_siguiente=next_start_code(a,indice_actual+4);
   num_bytes=indice_siguiente-indice_actual;

   %group_of_pictures_header
   aux=a(indice_actual:indice_actual+num_bytes-1);
   group_of_pictures_header(aux);

   num_GOP=1;       %Contador de GOPs para reordenar tramas

   indice_actual=indice_siguiente;

   %extension_and_user_data(1)
   [indice_actual]=extension_and_user_data(1,a,indice_actual);
end

indice_siguiente=next_start_code(a,indice_actual+4);
num_bytes=indice_siguiente-indice_actual;

%picture_header()
aux=a(indice_actual:indice_actual+num_bytes-1);
[temporal_reference,picture_coding_type]=picture_header(aux);

num_frame_in_GOP=temporal_reference+1; %Contador de tramas en GOPs para reordenar tramas

indice_actual=indice_siguiente;

indice_siguiente=next_start_code(a,indice_actual+4);
num_bytes=indice_siguiente-indice_actual;

%picture_coding_extension()
aux=a(indice_actual:indice_actual+num_bytes-1);
[picture_structure,frame_pred_frame_dct,concealment_motion_vectors,...
    f_code,precision,alternate_scan,q_scale_type,intra_vlc_format]=picture_coding_extension(aux);

indice_actual=indice_siguiente;

%extension_and_user_data(2)
[indice_actual]=extension_and_user_data(2,a,indice_actual);
```

```
indice_siguiente=next_start_code(a,indice_actual+4);
num_bytes=indice_siguiente-indice_actual;

%picture_data()
%*************************************************************************************************
%                                          Picure_data                                           %
%                                                                    Nº of bits  Mnemonic        %
%   picture_data() {                                                                             %
%       do {                                                                                     %
%           slice()                                                                              %
%       } while (nextbits()==slice_start_code                                                    %
%       next_start_code()                                                                        %
%   }                                                                                            %
%*************************************************************************************************

%do{
aux=a(indice_actual:indice_actual+num_bytes-1+3);   %Se pasan 3 bytes pertenecientes a las siguiente
                                                    %estructura para determinar fin de slice
    %slice()
[slice_luma,slice_chroma_b,slice_chroma_r,mb_row,mb_column]=slice(aux,horizontal_size,...
    vertical_size,mb_width,picture_coding_type,picture_structure,frame_pred_frame_dct,...
    concealment_motion_vectors,chroma_format,f_code,precision,alternate_scan,...
    intra_quantiser_matrix,non_intra_quantiser_matrix,q_scale_type,intra_vlc_format,im_referencia);

pos_y=(mb_row-1)*16+1;      %Posición del primer pixel de cada slice

%Formación de imágenes
pict_luma(pos_y:pos_y+15,1:mb_column*16)=slice_luma;
pict_chroma_b(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_b;
pict_chroma_r(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_r;

indice_actual=indice_siguiente;
%}while(nextbits()==slice_start_code
while((a(indice_actual+3)>=slice_start_code(1)) & (a(indice_actual+3)<=slice_start_code(2)))

    indice_siguiente=next_start_code(a,indice_actual+4);
    num_bytes=indice_siguiente-indice_actual;

    aux=a(indice_actual:indice_actual+num_bytes-1+3);
    [slice_luma,slice_chroma_b,slice_chroma_r,mb_row,mb_column]=slice(aux,horizontal_size,...
        vertical_size,mb_width,picture_coding_type,picture_structure,frame_pred_frame_dct,...
        concealment_motion_vectors,chroma_format,f_code,precision,alternate_scan,...
        intra_quantiser_matrix,non_intra_quantiser_matrix,q_scale_type,intra_vlc_format,...
        im_referencia);
    indice_actual=indice_siguiente;

    pos_y=(mb_row-1)*16+1;  %Posición del primer pixel de cada slice

    %Formación de imagenes
    pict_luma(pos_y:pos_y+15,1:mb_column*16)=slice_luma;
    pict_chroma_b(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_b;
    pict_chroma_r(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_r;
end

%Matriz con las 3 componentes (luma, croma azul y croma roja)
YCbCr(:,:,1)=(pict_luma);
YCbCr(:,:,2)=(pict_chroma_b);
YCbCr(:,:,3)=(pict_chroma_r);

YCbCr=double(YCbCr);    %Necesario para la siguiente función

%Matriz con las 2 componentes de color
RGB=colorspace('YCbCr->RGB',double(YCbCr));

num_frame=num_frame_in_GOP; %Es el primer GOP, no hay que calcular nada más

figure(num_frame);
image(RGB)

M(num_frame)=getframe;  %Formación de movie frame

%Almacenamiento de predicciones
if(picture_coding_type==1) %Imagen tipo I
    frame_pred_I=YCbCr;
end

primera_P=0;

%}while((nextbits()==picture_start_code)||(nextbits()==group_start_code
while(((a(indice_actual+3)==picture_start_code)|(a(indice_actual+3)==group_start_code))&(flag_gop==0))
    %if(nextbits()==group_start_code)
    if(a(indice_actual+3)==group_start_code)

        indice_siguiente=next_start_code(a,indice_actual+4);
        num_bytes=indice_siguiente-indice_actual;
```

```
    %group_of_pictures_header
    aux=a(indice_actual:indice_actual+num_bytes-1);
    group_of_pictures_header(aux);

    flag_gop=1;         %Indica que no es el primer GOP
    num_GOP=num_GOP+1;  %Contador de GOPs para reordenar tramas
    numero_frames_in_GOP=num_frame_P+2; %Numero total de frames en un GOP

    indice_actual=indice_siguiente;

    %extension_and_user_data(1)
    [indice_actual]=extension_and_user_data(1,a,indice_actual);
end

indice_siguiente=next_start_code(a,indice_actual+4);
num_bytes=indice_siguiente-indice_actual;

%picture_header()
aux=a(indice_actual:indice_actual+num_bytes-1);
[temporal_reference,picture_coding_type]=picture_header(aux);
num_frame_in_GOP=temporal_reference+1;

%Matrices para predicciones
if(picture_coding_type==1)  %Imagen tipo I
    im_referencia=0;        %No hay predicción
end

if(picture_coding_type==2)  %Imagen tipo P
    if(primera_P==0)
        im_referencia(1:240,1:320,1:3,1)=frame_pred_I;  %Predicción forward de tipo I
        im_referencia=int16(im_referencia);
    else
        im_referencia(1:240,1:320,1:3,1)=frame_pred_P;  %Predicción forward de tipo P
        im_referencia=int16(im_referencia);
    end
    num_frame_P=num_frame_in_GOP;
end

if(picture_coding_type==3)  %Imagen tipo P
    if(primera_P==0)
        im_referencia(1:240,1:320,1:3,1)=frame_pred_I;  %Prediccion forward de tipo I
        im_referencia=int16(im_referencia);
    else
        im_referencia(1:240,1:320,1:3,1)=frame_pred_P;  %Prediccion forward de tipo P
        im_referencia=int16(im_referencia);
    end
    im_referencia(1:240,1:320,1:3,2)=frame_pred_P;      %Prediccion backward de tipo P
    im_referencia=int16(im_referencia);
end

indice_actual=indice_siguiente;

indice_siguiente=next_start_code(a,indice_actual+4);
num_bytes=indice_siguiente-indice_actual;

%picture_coding_extension()
aux=a(indice_actual:indice_actual+num_bytes-1);
[picture_structure,frame_pred_frame_dct,concealment_motion_vectors,f_code,precision,...
    alternate_scan,q_scale_type,intra_vlc_format]=picture_coding_extension(aux);

indice_actual=indice_siguiente;

%extension_and_user_data(2)
[indice_actual]=extension_and_user_data(2,a,indice_actual);

indice_siguiente=next_start_code(a,indice_actual+4);
num_bytes=indice_siguiente-indice_actual;

%picture_data()
%*************************************************************************************************
%                                              Picure_data                                       %
%                                                                  Nº of bits  Mnemonic          %
%   picture_data() {                                                                             %
%       do {                                                                                     %
%           slice()                                                                              %
%       } while (nextbits()==slice_start_code                                                    %
%       next_start_code()                                                                        %
%   }                                                                                            %
%*************************************************************************************************

%do{
    %slice()

aux=a(indice_actual:indice_actual+num_bytes-1+3);   %Se pasan 3 bytes pertenecientes a la
```

```
                                                    %sigiente estructura para determinar
                                                    %fin de slice
    [slice_luma,slice_chroma_b,slice_chroma_r,mb_row,mb_column]=slice(aux,horizontal_size, ...
        vertical_size,mb_width,picture_coding_type,picture_structure,frame_pred_frame_dct, ...
        concealment_motion_vectors,chroma_format,f_code,precision,alternate_scan, ...
        intra_quantiser_matrix,non_intra_quantiser_matrix,q_scale_type,intra_vlc_format, ...
        im_referencia);

    pos_y=(mb_row-1)*16+1;      %Posición del primer pixel de cada slice

    %Formación de imagenes
    pict_luma(pos_y:pos_y+15,1:mb_column*16)=slice_luma;
    pict_chroma_b(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_b;
    pict_chroma_r(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_r;

    indice_actual=indice_siguiente;

    %}while(nextbits()==slice_start_code
    while((a(indice_actual+3)>=1) & (a(indice_actual+3)<=175))

        indice_siguiente=next_start_code(a,indice_actual+4);
        num_bytes=indice_siguiente-indice_actual;

        aux=a(indice_actual:indice_actual+num_bytes-1+3);
        [slice_luma,slice_chroma_b,slice_chroma_r,mb_row,mb_column]=slice(aux,horizontal_size, ...
            vertical_size,mb_width,picture_coding_type,picture_structure,frame_pred_frame_dct, ...
            concealment_motion_vectors,chroma_format,f_code,precision,alternate_scan, ...
            intra_quantiser_matrix,non_intra_quantiser_matrix,q_scale_type,intra_vlc_format, ...
            im_referencia);

        pos_y=(mb_row-1)*16+1   %Posición del primer pixel de cada slice

        %Formacio de imagenes
        pict_luma(pos_y:pos_y+15,1:mb_column*16)=slice_luma;
        pict_chroma_b(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_b;
        pict_chroma_r(pos_y:pos_y+15,1:mb_column*16)=slice_chroma_r;

        indice_actual=indice_siguiente;
    end


    %Matriz con las 3 componentes (luma, croma azul y croma roja)
    YCbCr(:,:,1)=pict_luma;
    YCbCr(:,:,2)=pict_chroma_b;
    YCbCr(:,:,3)=pict_chroma_r;

    YCbCr=double(YCbCr);    %Necesario para la siguiente función

    %Matriz con las 2 componentes de color
    RGB=colorspace('YCbCr->RGB',YCbCr);
    if(flag_gop==1)
        num_frame=num_frame_in_GOP+num_frame_P+numero_frames_in_GOP*(num_GOP-1);
    else
        num_frame=num_frame_in_GOP;
    end

    figure(num_frame)
    image(RGB)

    M(num_frame)=getframe;  %Formación de movie frame

    %Almacenamiento de predicciones
    if(picture_coding_type==1)  %Imagen tipo I
        frame_pred_I=YCbCr;
    end
    if(picture_coding_type==2)  %Imagen tipo P
        primera_P=1;
        frame_pred_P=YCbCr;
    end

end
a(indice_actual+3)=183;
%if(next_bits()!=sequence_end_code)
if(a(indice_actual+3)~=sequence_end_code)

    indice_siguiente=next_start_code(a,indice_actual+4);
    num_bytes=indice_siguiente-indice_actual;

    %sequence_header()
    aux=a(indice_actual:indice_actual+num_bytes-1);
    [horizontal_size_value,vertical_size_value]=sequence_header(aux);

    indice_actual=indice_siguiente;

    %sequence_extension()
```

```
        [indice_actual]=extension_and_user_data(0,nombre_archivo,indice_actual);
    end
else

    %iso/iec 11172-2

end

sequence_end_code=a(indice_actual:indice_actual+3);
```

```
%*************************************************************************************************
%                                        Sequence header                                         %
%                                                          Nº of bits          Mnemonic        %
%   sequence_header() {                                                                          %
%       sequence_header_code                                  32               bslbf           %
%       horizontal_size_value                                 12               uimsbf          %
%       vertical_size_value                                   12               uimsbf          %
%       aspect_ratio_information                               4               uimsbf          %
%       frame_rate_code                                        4               uimsbf          %
%       bit_rate_value                                        18               uimsbf          %
%       marker_bit                                             1               bslbf           %
%       vbv_buffer_size_value                                 10               uimsbf          %
%       constrained_parameters_flag                            1               bslbf           %
%       load_intra_quantiser_matrix                            1               uimsbf          %
%       if(load_intra_quantiser_matrix)                                                         %
%           intra_quantiser_matrix[64]                      8*64               uimsbf          %
%       load_non_intra_quantiser_matrix                        1               uimsbf          %
%       if(load_non_itra_quantiser_matrix)                                                      %
%           non_intra_quantiser_matrix[64]                  8*64               uimsbf          %
%       next_start_code()                                                                       %
%   }                                                                                           %
%*************************************************************************************************

%Parametros de entrada
%   a:bytes que pertenecen a la cabecera

%Parametros de salida
%   horizontal_size_value
%   vertical_size_value
%   intra_quantiser_matrix
%   non_intra_quantiser_matrix

function [horizontal_size_value,vertical_size_value,intra_quantiser_matrix, ...
    non_intra_quantiser_matrix]=sequence_header(a)

%Convertir de decimal a binario

a_bin=dec2bin(a);
[num_bytes,num_bits]=size(a_bin);

for(i=1:num_bytes)
    for(j=1:num_bits)
    bitstream(1,(i-1)*num_bits+j)=a_bin(i,j);
    j=j+1;
    end
    i=i+1;
end

k=1;

%Extraer parametros de la cabecera

bits=32;
sequence_header_code=bitstream(k:k+bits-1);
k=k+bits;

bits=12;
aux=bitstream(k:k+bits-1);
horizontal_size_value=uint16(bin2dec(aux));
k=k+bits;

bits=12;
aux=bitstream(k:k+bits-1);
vertical_size_value=uint16(bin2dec(aux));
k=k+bits;

bits=4;
aux=bitstream(k:k+bits-1);
aspect_ratio_information=uint8(bin2dec(aux));
%aspect_ratio_information       Sample Aspect Ratio         DAR
%
%           0                   forbidden                   forbidden
%           1                   1,0 (Square Sample)         -
%           2                   -                           3:4
%           3                   -                           9:16
%           4                   -                           1:2,21
%           5                   -                           reserved
%           ...                 ...                         ...
%           15                  -                           reserved
k=k+bits;

bits=4;
aux=bitstream(k:k+bits-1);
frame_rate_code=uint8(bin2dec(aux));
%frame_rate_code        frame_rate_value
```

```
%   0                   forbidden
%   1                   24000:1001 (23,976...)
%   2                   24
%   3                   25
%   4                   3000:1001 (29.97...)
%   5                   30
%   6                   50
%   7                   60000:1001 (59.94...)
%   8                   60
%   ...                 reserved
%   15                  reserved

k=k+bits;

bits=18;
aux=bitstream(k:k+bits-1);
bit_rate_value=uint32(bin2dec(aux));
k=k+bits;

bits=1;
marker_bit=bitstream(k:k+bits-1);
k=k+bits;

bits=10;
aux=bitstream(k:k+bits-1);
vbv_buffer_size_value=uint16(bin2dec(aux));
k=k+bits;

bits=1;
constrained_parameters_flag=bitstream(k:k+bits-1);
k=k+bits;

bits=1;
aux=bitstream(k:k+bits-1);
load_intra_quantiser_matrix=uint8(bin2dec(aux));
k=k+bits;

if(load_intra_quantiser_matrix==1)
        i=1;
        j=1;
        bits=8*64;
        aux=bitstream(k:k+bits-1);
        for(i=1:8)
            for(j=1:8)
                l=((j-1)*8+(i-1)*8)+1;
                intra_quantiser_matrix(i,j)=uint8(bin2dec(aux(l:l+8)));
                j=j+1;
            end
            i=i+1;
        end
        k=k+bits;
else        %matriz por defecto
    intra_quantiser_matrix=uint8([8 16 19 22 26 27 29 34
                                  16 16 22 24 27 29 34 37
                                  19 22 26 27 29 34 34 38
                                  22 22 26 27 29 34 37 40
                                  22 26 27 29 32 35 40 48
                                  26 27 29 32 35 40 48 58
                                  26 27 29 34 38 46 56 69
                                  27 29 35 38 46 56 69 83]);
end

bits=1;
aux=bitstream(k:k+bits-1);
load_non_intra_quantiser_matrix=uint8(bin2dec(aux));
k=k+bits;

if(load_non_intra_quantiser_matrix==1)
    i=1;
    j=1;
    bits=8*64;
    aux=bitstream(k:k+bits-1);
    for(i=1:8)
        for(j=1:8)
            l=((j-1)*8+(i-1)*8)+1;
            non_intra_quantiser_matrix(i,j)=uint8(bin2dec(aux(l:l+8)));
            j=j+1;
        end
            i=i+1;
    end
    k=k+bits;
else        %matriz por defecto
    non_intra_quantiser_matrix=uint8([16 16 16 16 16 16 16 16
                                      16 16 16 16 16 16 16 16
                                      16 16 16 16 16 16 16 16
```

```
                                16 16 16 16 16 16 16 16
                                16 16 16 16 16 16 16 16
                                16 16 16 16 16 16 16 16
                                16 16 16 16 16 16 16 16
                                16 16 16 16 16 16 16 16]);
end
```

```
%*************************************************************************************************
%                                          Sequence extension                                     %
%                                                                  Nº of bits          Mnemonic   %
%   sequence_extension() {                                                                        %
%       extension_start_code                                          32              bslbf      %
%       extension_start_code_identifier                                4              uimsbf     %
%       profile_and_level_indication                                   8              uimsbf     %
%       progressive_sequence                                           1              uimsbf     %
%       chroma_format                                                  2              uimsbf     %
%       horizontal_size_extension                                      2              uimsbf     %
%       vertical_size_extension                                        2              uimsbf     %
%       bit_rate_extension                                            12              uimsbf     %
%       marker_bit                                                     1              bslbf      %
%       vbv_buffer_size_extension                                      8              uimsbf     %
%       low_delay                                                      1              uimsbf     %
%       frame_rate_extension_n                                         2              uimsbf     %
%       frame_rate_extension_d                                         5              uimsbf     %
%       next_start_code()                                                                         %
%   }                                                                                             %
%*************************************************************************************************

%Parametros de entrada
%   a:bytes que pertenecen a la cabecera

%Parametros de salida
%   progressive_sequence
%   chroma_format
%   horizontal_size_extension
%   vertical_size_extension

function [progressive_sequence,chroma_format,horizontal_size_extension,...
    vertical_size_extension]=sequence_extension(a)

%Convertir de decimal a binario

a_bin=dec2bin(a);
[num_bytes,num_bits]=size(a_bin);

for(i=1:num_bytes)
    for(j=1:num_bits)
    bitstream(1,(i-1)*num_bits+j)=a_bin(i,j);
    j=j+1;
    end
    i=i+1;
end

k=1;

%Extraer parametros de la cabecera

bits=32;
extension_start_code=bitstream(k:k+bits-1);
k=k+bits;

bits=4;
aux=bitstream(k:k+bits-1);
extension_start_code_identifier=uint8(bin2dec(aux));
k=k+bits;

bits=8;
aux=bitstream(k:k+bits-1);
profile_and_level_indication=uint8(bin2dec(aux));
k=k+bits;

bits=1;
aux=bitstream(k:k+bits-1);
progressive_sequence=uint8(bin2dec(aux));
k=k+bits;

bits=2;
aux=bitstream(k:k+bits-1);
chroma_format=uint8(bin2dec(aux));
%chroma_format      Meaning
%   0               reserved
%   1               4:2:0
%   2               4:2:2
%   3               4:4:4
k=k+bits;

bits=2;
aux=bitstream(k:k+bits-1);
horizontal_size_extension=uint8(bin2dec(aux));
k=k+bits;

bits=2;
```

```
aux=bitstream(k:k+bits-1);
vertical_size_extension=uint8(bin2dec(aux));
k=k+bits;

bits=12;
aux=bitstream(k:k+bits-1);
bit_rate_extension=uint16(bin2dec(aux));
k=k+bits;

bits=1;
marker_bit=bitstream(k:k+bits-1);
k=k+bits;

bits=8;
aux=bitstream(k:k+bits-1);
vbv_buffer_size_extension=uint8(bin2dec(aux));
k=k+bits;

bits=1;
aux=bitstream(k:k+bits-1);
low_delay=uint8(bin2dec(aux));
k=k+bits;

bits=2;
aux=bitstream(k:k+bits-1);
frame_rate_extension_n=uint8(bin2dec(aux));
k=k+bits;

bits=5;
aux=bitstream(k:k+bits-1);
frame_rate_extension_d=uint8(bin2dec(aux));
k=k+bits;
```

```
%*************************************************************************************************
%                                          Extension and user data                                %
%                                                          Nº of bits          Mnemonic          %
%   extension_and_user_data() {                                                                   %
%       while((nextbits()==extension_start_code)||                                                %
%               (nextbits()==user_data_start_code)) {                                             %
%           if((i!=1)&&(nextbits()==extension_start_code))                                        %
%               extension_data(i)                                                                 %
%           if(nextbits()==user_data_start_code)                                                  %
%               user_data()                                                                       %
%       }                                                                                         %
%   }                                                                                             %
%*************************************************************************************************

%Parametros de entrada
%   i
%   a:cadena completa, se desconoce el numero de bytes que empleara la extension
%   indice_actual

%Parametros de salida
%   indice_siguiente: para saber el numero de bytes leidos

function [indice_siguiente]=extension_and_user_data(i,a,indice_actual)

table_stream_id;

while((a(indice_actual+3)==extension_start_code) | (a(indice_actual+3)==user_data_start_code))
    if((i~=1) & (a(indice_actual+3)==extension_start_code))
        indice_siguiente=next_start_code(a,indice_actual+4);
        num_bytes=indice_siguiente-indice_actual;
        aux=a(indice_actual:indice_actual+num_bytes-1);
        extension_data(i,aux);
        indice_actual=indice_siguiente;
    end
    if(a(indice_actual+3)==user_data_start_code)
        indice_siguiente=next_start_code(a,indice_actual+4);
        num_bytes=indice_siguiente-indice_actual;
        aux=a(indice_actual:indice_actual+num_bytes-1);
        user_data(aux);
        indice_actual=indice_siguiente;
    end
end
indice_siguiente=indice_actual;
```

```
%*************************************************************************************************************
%                                            Extension data                                                   %
%                                                                          Nº of bits          Mnemonic     %
%   extension_data(i) {                                                                                      %
%       while(nextbits()==extension_start_code) {                                                            %
%           extension_start_code                                              32              bslbf        %
%           if(i==0) {/*follows sequence_extension()*/                                                       %
%               if(nextbits()=="Sequence Display Extension ID")                                              %
%                   sequene_display_extension()                                                              %
%               else                                                                                         %
%                   sequence_scalable_extension()                                                            %
%           }                                                                                                %
%       /* NOTE - i never takes the value 1 because extension_data()                                         %
%                         never follows a group_of_pictures_header() */                                      %
%           if(i==2) { /*follows picture_coding_extension() */                                               %
%               if(nextbits()=="Quant Matrix Extension ID")                                                  %
%                   quant_matrix_extension()                                                                 %
%               else if(nextbits()=="Copyright Extension ID")                                                %
%                   copyright_extension()                                                                    %
%               else if(nextbits()=="Picture Display Extension ID")                                          %
%                   picture_display_extension()                                                              %
%               else if(nextbits()=="Picture Spatial Scalable Extension ID")                                 %
%                   picture_spatial_scalable_extension                                                       %
%               else                                                                                         %
%                   picture_temporal_scalable_extension()                                                    %
%           }                                                                                                %
%       }                                                                                                    %
%   }                                                                                                        %
%*************************************************************************************************************

%Parametros de entrada
%   l
%   a=bytes que pertenecen a la extension

%Parametros de salida
%

function extension_data(l,a)

table_stream_id

%Convertir de decimal a binario

a_bin=dec2bin(a);
[num_bytes,num_bits]=size(a_bin);

for(i=1:num_bytes)
    for(j=1:num_bits)
    bitstream(1,(i-1)*num_bits+j)=a_bin(i,j);
    j=j+1;
    end
    i=i+1;
end

k=1;

bits=32;
extension_start_code=bitstream(k:k+bits-1);
k=k+bits;

if(l==0)
    bits=4;
    if(bitstream(k:k+bits-1)==Sequence_Display_Extension_ID)
        sequence_display_extension(bitstream(k:num_bytes*num_bits));
    else
        sequence_scalable_extension(bitstream(k:num_bytes*num_bits));
    end
end

if(l==2)
    bits=4;
    if(bitstream(k:k+bits-1)==Quant_Matrix_Extension_ID)
        quant_matrix_extension(bitstream(k:num_bytes*num_bits));

    else if (bitstream(k:k+bits-1)==Copyright_Extension_ID)
            copyright_extension(bitstream(k:num_bytes*num_bits));

        else if(bitstream(k:k+bits-1)==Picture_Display_Extension_ID)
                picture_display_extension(bitstream(k:num_bytes*num_bits));

            else if(bitstream(k:k+bits-1)==Picture_Spatial_Scalable_Extension_ID)
                    picture_spatial_scalable_extension(bitstream(k:num_bytes*num_bits));

                else
                    picure_temporal_scalable_extension(bitstream(k:num_bytes*num_bits));
```

```
                end
            end
        end
    end
end
```

```
%*************************************************************************************************************
%                                                User data                                                    %
%                                                                                 Nº of bits          Mnemonic   %
%   user_data() {                                                                                              %
%       user_data_start_code                                                         32                bslbf      %
%       while(nextbits()!='0000 0000 0000 0000 0000 0001') {                                                   %
%           user_data                                                                 8                uimsbf     %
%       }                                                                                                      %
%       next_start_code                                                                                        %
%   }                                                                                                          %
%*************************************************************************************************************

%Parametros de entrada
%   a=bytes que pertenecen a la extension

%Parametros de salida
%

function user_datas=user_data(a)

k=1;

bytes=4;
user_data_start_code=a(k:k+bytes-1);
k=k+bytes;

num_bytes=size(a);

user_datas=a(k:num_bytes(1)-4);
```

```
%*************************************************************************************************
%                                        Sequence display extension                               %
%                                                          Nº of bits          Mnemonic          %
%   sequence_display_extension() {                                                                %
%       extension_start_code_identifier                        4                uimsbf           %
%       video_format                                           3                uimsbf           %
%       colour_description                                     1                uimsbf           %
%       if(colour_description) {                                                                  %
%           colour_primaries                                   8                uimsbf           %
%           transfer_characteritics                            8                uimsbf           %
%           matrix_coefficients                                8                uimsbf           %
%       }                                                                                         %
%       display_horizontal_size                               14                uimsbf           %
%       marker_bit                                             1                bslbf            %
%       display_vertical_size                                 14                uimsbf           %
%       next_start_code()                                                                         %
%   }                                                                                             %
%*************************************************************************************************

%Parametros de entrada
%   bitstream: bits pertenecientes a la extension

%Parametros de salida
%

function sequence_display_extension(bitstream)

k=1;

%Extraccion de parametros

bits=4;
extension_start_code_identifier=bitstream(k:k+bits-1);
k=k+bits;

bits=3;
aux=bitstream(k:k+bits-1);
video_format=uint8(bin2dec(aux));
%video_format   Meaning
%   0           component
%   1           PAL
%   2           NTSC
%   3           SECAM
%   4           MAC
%   5           Unspecified video format
%   6           reserved
%   7           reserved
k=k+bits;

bits=1;
aux=bitstream(k:k+bits-1);
colour_description=uint8(bin2dec(aux));
k=k+bits;

if(colour_description==1)
    bits=8;
    aux=bitstream(k:k+bits-1);
    colour_primaries=uint8(bin2dec(aux));
    k=k+bits;

    bits=8;
    aux=bitstream(k:k+bits-1);
    transfer_characteristics=uint8(bin2dec(aux));
    k=k+bits;

    bits=8;
    aux=bitstream(k:k+bits-1);
    matrix_coefficients=uint8(bin2dec(aux));
    k=k+bits;
end

bits=14;
aux=bitstream(k:k+bits-1);
display_horizontal_size=uint16(bin2dec(aux));
k=k+bits;

bits=1;
marker_bit=bitstream(k:k+bits-1);
k=k+bits;

bits=14;
aux=bitstream(k:k+bits-1);
display_vertical_size=uint16(bin2dec(aux));
k=k+bits;
```

```
%*************************************************************************************************************
%                                          Sequence scalable extension                                      %
%                                                          Nº of bits          Mnemonic                    %
%   sequence_scalable_extension() {                                                                         %
%       extension_start_code_identifier                        4               uimsbf                      %
%       scalable_mode                                          2               uimsbf                      %
%       layer_id                                               4               uimsbf                      %
%       if(scalable_mode=="spatial_scalability) {                                                          %
%           lower_layer_prediction_horizontal_size            14               uimsbf                      %
%           marker_bit                                         1               bslbf                       %
%           lower_layer_prediction_vertical_size              14               uimsbf                      %
%           horizontal_subsampling_factor_m                    5               uimsbf                      %
%           horizontal_subsampling_factor_n                    5               uimsbf                      %
%           vertical_subsampling_factor_m                      5               uimsbf                      %
%           vertical_subsampling_factor_n                      5               uimsbf                      %
%       }                                                                                                  %
%       if(scalable_mode=="temporal scalability") {                                                        %
%           picture_mux_enable                                 1               uimsbf                      %
%           if(picture_mux_enable)                                                                         %
%               mux_to_progressive_sequence                    1               uimsbf                      %
%           picture_mux_order                                  3               uimsbf                      %
%           picture_mux_factor                                 3               uimsbf                      %
%       }                                                                                                  %
%       next_start_code()                                                                                  %
%   }                                                                                                      %
%*************************************************************************************************************

%Parametros de entrada
%   bitstream: bits pertenecientes a la extension

%Parametros de salida
%

function sequence_scalable_extension(bitstream)

k=1;

bits=4;
extension_start_code_identifier=bitstream(k:k+bits-1);
k=k+bits;

bits=2;
aux=bitstream(k:k+bits-1);
scalable_mode=uint8(bin2dec(aux));
%scalable_mode    Meaning                 picture_spatial_scalable_extension()    macroblock_type tables
%   sequence_scalable_extension() not present                                       B-2,B-3 and B-4
%   0             data partitioning                                                 B-2,B-3 and B-4
%   1             spatial scalability     present                                   B-5,B-6 and B-4
%                                         not present                               B-2,B-3 and B-4
%   2             SNR scalability                                                   B-8
%   3             temporal scalability                                              B-2,B-3 and B-4
k=k+bits;

bits=4;
aux=bitstream(k:k+bits-1);
layer_id=uint8(bin2dec(aux));
k=k+bits;

if(scalable_mode==1)
    bits=14;
    aux=bitstream(k:k+bits-1);
    lower_layer_prediction_horizontal_size=uint16(bin2dec(aux));
    k=k+bits;

    bits=1;
    marker_bit=bitstream(k:k+bits-1);
    k=k+bits;

    bits=14;
    aux=bitstream(k:k+bits-1);
    lower_layer_prediction_vertical_size=uint16(bin2dec(aux));
    k=k+bits;

    bits=5;
    aux=bitstream(k:k+bits-1);
    horizontal_subsampling_factor_m=uint8(bin2dec(aux));
    k=k+bits;

    bits=5;
    aux=bitstream(k:k+bits-1);
    horizontal_subsampling_factor_n=uint8(bin2dec(aux));
    k=k+bits;

    bits=5;
    aux=bitstream(k:k+bits-1);
```

```
    vertical_subsampling_factor_m=uint8(bin2dec(aux));
    k=k+bits;

    bits=5;
    aux=bitstream(k:k+bits-1);
    vertical_subsampling_factor_m=uint8(bitstream(k:k+bits-1);
    k=k+bits;
end

if(scalable_mode==3)
    bits=1;
    aux=bitstream(k:k+bits-1);
    picture_mux_enable=uint8(bin2dec(aux));
    k=k+bits;

    if(picture_mux_enable==1)
        bits=1;
        aux=bitstream(k:k+bits-1);
        mux_to_progressive_sequence=uint8(bin2dec(aux));
        k=k+bits;
    end

    bits=3;
    aux=bitstream(k:k+bits-1);
    picture_mux_order=uint8(bin2dec(aux));
    k=k+bits;

    bits=3;
    aux=bitstream(k:k+bits-1);
    picture_mux_factor=uint8(bin2dec(aux));
    k=k+bits;
end
```

```
%*************************************************************************************************************
%                                          Quant matrix extension                                             %
%                                                                 Nº of bits          Mnemonic   %
%   quant_matrix_extension() {                                                                   %
%       extension_start_code_identifier                              4              uimsbf     %
%       load_intra_quantiser_matrix                                  1              uimsbf     %
%       if(load_intra_quantiser_matrix)                                                         %
%           intra_quantiser_matrix[64]                             8*64             uimsbf     %
%       load_non_intra_quantiser_matrix                              1              uimsbf     %
%       if(load_non_intra_quantiser_matrix)                                                     %
%           non_intra_quantiser_matrix[64]                         8*64             uimsbf     %
%       load_chroma_intra_quantiser_matrix                           1              uimsbf     %
%       if(load_chroma_intra_quantiser_matrix)                                                  %
%           chroma_intra_quantiser_matrix[64]                      8*64             uimsbf     %
%       load_chroma_non_intra_quantiser_matrix                       1              uimsbf     %
%       if(load_chroma_non_intra_quantiser_matrix)                                              %
%           chroma_non_intra_quantiser_matrix[64]                  8*64             uimsbf     %
%       next_start_code()                                                                       %
%   }                                                                                           %
%*************************************************************************************************************

%Parametros de entrada
%   bitstream: bits pertenecientes a la extension

%Parametros de salida
%

function quant_matrix_extension(bitstream)

k=1;

bits=4;
extension_start_code_identifier=bitstream(k:k+bits-1);
k=k+bits;

bits=1;
aux=bitstream(k:k+bits-1);
load_intra_quantiser_matrix=uint8(bin2dec(aux));
k=k+bits;

if(load_intra_quantiser_matrix==1)
        i=1;
        j=1;
        bits=8*64;
        aux=bitstream(k:k+bits-1);
        for(i=1:8)
            for(j=1:8)
                l=((j-1)*8+(i-1)*8)+1;
                intra_quantiser_matrix(i,j)=uint8(bin2dec(aux(l:l+8));
                j=j+1;
            end
            i=i+1;
        end
        k=k+bits;
end

bits=1;
aux=bitstream(k:k+bits-1);
load_non_intra_quantiser_matrix=uint8(bin2dec(aux));
k=k+bits;

if(load_non_intra_quantiser_matrix==1)
    i=1;
    j=1;
    bits=8*64;
    aux=bitstream(k:k+bits-1);
    for(i=1:8)
        for(j=1:8)
            l=((j-1)*8+(i-1)*8)+1;
            non_intra_quantiser_matrix(i,j)=uint8(bin2dec(aux(l:l+8)));
            j=j+1;
        end
            i=i+1;
    end
    k=k+bits;
end

bits=1;
aux=bitstream(k:k+bits-1);
load_chroma_intra_quantiser_matrix=uint8(bin2dec(aux));
k=k+bits;

if(load_chroma_intra_quantiser_matrix==1)
        i=1;
        j=1;
```

```
        bits=8*64;
        aux=bitstream(k:k+bits-1);
        for(i=1:8)
            for(j=1:8)
                l=((j-1)*8+(i-1)*8)+1;
                chroma_intra_quantiser_matrix(i,j)=uint8(bin2dec(aux(l:l+8)));
                j=j+1;
            end
            i=i+1;
        end
        k=k+bits;
end

bits=1;
aux=bitstream(k:k+bits-1);
load_chroma_non_intra_quantiser_matrix=uint8(bin2dec(aux));
k=k+bits;

if(load_chroma_non_intra_quantiser_matrix==1)
        i=1;
        j=1;
        bits=8*64;
        aux=bitstream(k:k+bits-1);
        for(i=1:8)
            for(j=1:8)
                l=((j-1)*8+(i-1)*8)+1;
                chroma_non_intra_quantiser_matrix(i,j)=uint8(bin2dec(aux(l:l+8)));
                j=j+1;
            end
            i=i+1;
        end
        k=k+bits;
end
```

```
%*************************************************************************************************
%                                        Copyright extension                                     %
%                                                          Nº of bits          Mnemonic        %
%   copyright_extension() {                                                                      %
%       extension_start_code_identifier                        4                uimsbf          %
%       copyright_flag                                         1                bslbf           %
%       copyright_idendifier                                   8                uimsbf          %
%       original_or_copy                                       1                bslbf           %
%       reserved                                               7                uimsbf          %
%       marker_bit                                             1                bslbf           %
%       copyright_number_1                                    20                uimsbf          %
%       marker_bit                                             1                bslbf           %
%       copyright_number_2                                    22                uimsbf          %
%       marker_bit                                             1                bslbf           %
%       copyright_number_3                                    22                uimsbf          %
%       next_start_code()                                                                        %
%   }                                                                                            %
%*************************************************************************************************

%Parametros de entrada
%   bitstream:bits que pertenecen a la extension

%Parametros de salida
%

function copyright_extension(bitstream)

k=1;

%Extraccion de parametros

bits=4;
extension_start_code=bitstream(k:k+bits-1);
k=k+bits;

bits=1;
copyright_flag=bitstream(k:k+bits-1);
k=k+bits;

bits=8;
aux=bitstream(k:k+bits-1);
copyright_identifier=uint8(bin2dec(aux));
k=k+bits;

bits=1;
original_or_copy=bitstream(k:k+bits-1);
k=k+bits;

bits=7;
aux=bitstream(k:k+bits-1);
reserved=uint8(bin2dec(aux));
k=k+bits;

bits=1;
marker_bit=bitstream(k:k+bits-1);
k=k+bits;

bits=20;
aux=bitstream(k:k+bits-1);
copyright_number_1=uint32(bin2dec(aux));
k=k+bits;

bits=1;
marker_bit=bitstream(k:k+bits-1);
k=k+bits;

bits=22;
aux=bitstream(k:k+bits-1);
copyright_number_2=uint32(bin2dec(aux));
k=k+bits;

bits=1;
marker_bit=bitstream(k:k+bits-1);
k=k+bits;

bits=22;
aux=bitstream(k:k+bits-1);
copyright_number_3=uint32(bin2dec(aux));
k=k+bits;
```

```
%*************************************************************************************************
%                                        Group of pictures header                                 %
%                                                          Nº of bits          Mnemonic          %
%   group_of_pictures_header() {                                                                  %
%       group_start_code                                       32               bslbf           %
%       time_code                                              25               bslbf           %
%       closed_gop                                              1               uimsbf          %
%       broken_link                                             1               uimsbf          %
%       next_start_code()                                                                         %
%   }                                                                                             %
%*************************************************************************************************

%Parametros de entrada
%   a:bytes que pertenecen a la cabecera

%Parametros de salida
%

function group_of_pictures_header(a)

%Pasar de decimal a binario
a_bin=dec2bin(a);

[num_bytes,num_bits]=size(a_bin);
for(i=1:num_bytes)
    for(j=1:num_bits)
    bitstream(1,(i-1)*num_bits+j)=a_bin(i,j);
    j=j+1;
    end
    i=i+1;
end

k=1;

%Extraccion de parametros

bits=32;
group_start_code=bitstream(k:k+bits-1);
k=k+bits;

bits=25;
time_code=bitstream(k:k+bits-1);
k=k+bits;

bits=1;
aux=bitstream(k:k+bits-1);
closed_gop=uint8(bin2dec(aux));
k=k+bits;

bits=1;
broken_link=uint8(bin2dec(aux));
k=k+bits;
```

```
%*************************************************************************************************
%                                          Picture header                                        %
%                                                          Nº of bits          Mnemonic        %
%   picture_header() {                                                                           %
%       picture_start_code                                      32              bslbf           %
%       temporal_reference                                      10              uimsbf          %
%       picture_coding_type                                      3              uimsbf          %
%       vbv_delay                                               16              uimsbf          %
%       if(picture_coding_type==2||picture_coding_type==3) {                                     %
%           full_pel_forward_vector                              1              bslbf           %
%           forward_f_code                                       3              bslbf           %
%       }                                                                                        %
%       if(picture_coding_type==3) {                                                             %
%           full_pel_backward_vector                             1              bslbf           %
%           backward_f_code                                      3              bslbf           %
%       }                                                                                        %
%       while(nextbits()=='1') {                                                                 %
%           extra_bit_picture   /* with the value '1' */         1              uimsbf          %
%           extra_infromation_picture                            8              uimsbf          %
%       }                                                                                        %
%       extra_bit_picture   /* with the value '0' */             1              uimsbf          %
%       next_start_code()                                                                        %
%   }                                                                                            %
%*************************************************************************************************

%Parametros de entrada
%   a:bytes pertenecientes a la cabecera

%Parametros de salida
%   temporal_reference
%   picture_coding_type

function [temporal_reference,picture_coding_type]=picture_header(a)

%Pasar de decimal a binario

a_bin=dec2bin(a);

[num_bytes,num_bits]=size(a_bin);
for(i=1:num_bytes)
    for(j=1:num_bits)
    bitstream(1,(i-1)*num_bits+j)=a_bin(i,j);
    j=j+1;
    end
    i=i+1;
end

k=1;

%Extraer parametros

bits=32;
picture_start_code=bitstream(k:k+bits-1);
k=k+bits;

bits=10;
aux=bitstream(k:k+bits-1);
temporal_reference=uint16(bin2dec(aux));
k=k+bits;

bits=3;
aux=bitstream(k:k+bits-1);
picture_coding_type=uint8(bin2dec(aux));
%picture_coding_type    coding method
%   0                   forbidden
%   1                   intra_coded(I)
%   2                   predictive_coded(P)
%   3                   bidirectionally-predictive-coded(B)
%   4                   shall not be used (dc intra-coded (D) in ISO/IEC 11172-2)
%   5                   reserved
%   6                   reserved
%   7                   reserved
k=k+bits;

bits=16;
aux=bitstream(k:k+bits-1);
vbv_delay=uint16(bin2dec(aux));
k=k+bits;

if((picture_coding_type==2)|(picture_coding_type==3))
    bits=1;
    full_pel_forward_vector=bitstream(k:k+bits-1);
    k=k+bits;

    bits=3;
```

```
    forward_f_code=bitstream(k:k+bits-1);
    k=k+bits;
end

if(picture_coding_type==3)
    bits=1;
    full_pel_backward_vector=bitstream(k:k+bits-1);
    k=k+bits;

    bits=3;
    backward_f_code=bitstream(k:k+bits-1);
    k=k+bits;
end

while(bitstream(k)=='1')
    bits=1;
    aux=bitstream(k:k+bits-1);
    extra_bit_picture=uint8(bin2dec(aux));
    k=k+1;

    bits=8;
    aux=bitstream(k:k+bits-1);
    extra_information_picture=uint8(bin2dec(aux));
    k=k+bits;
end

bits=1;
aux=bitstream(k:k+bits-1);
extra_bit_picture=uint8(bin2dec(aux));
k=k+1;
```

```
%*************************************************************************************************************
%                                          Picture coding extension                                           %
%                                                          Nº of bits          Mnemonic                       %
%   picture_coding_extension() {                                                                              %
%       extension_start_code                                   32               bslbf                      %
%       extension_start_code_identifier                         4               uimsbf                     %
%       f_code[0][0] /* forward horizontal */                   4               uimsbf                     %
%       f_code[0][1] /* forward vertical */                     4               uimsbf                     %
%       f_code[1][0] /* backward horizontal */                  4               uimsbf                     %
%       f_code[1][1] /* backward vertical */                    4               uimsbf                     %
%       intra_dc_precision                                      2               uimsbf                     %
%       picture_structure                                       2               uimsbf                     %
%       top_field_first                                         1               uimsbf                     %
%       frame_pred_frame_dct                                    1               uimsbf                     %
%       concealment_motion_vectors                              1               uimsbf                     %
%       q_scale_type                                            1               uimsbf                     %
%       intra_vlc_format                                        1               uimsbf                     %
%       alternate_scan                                          1               uimsbf                     %
%       repeat_first_field                                      1               uimsbf                     %
%       chroma_420_type                                         1               uimsbf                     %
%       progressive_frame                                       1               uimsbf                     %
%       composite_display_flag                                  1               uimsbf                     %
%       if(composite_display_flag) {                                                                       %
%           v_axis                                              1               uimsbf                     %
%           field_sequence                                      3               uimsbf                     %
%           sub_carrier                                         1               uimsbf                     %
%           burst_amplitude                                     7               uimsbf                     %
%           sub_carrier_phase                                   8               uimsbf                     %
%       }                                                                                                  %
%       next_start_code()                                                                                  %
%   }                                                                                                      %
%*************************************************************************************************************

%Parametros de entrada
%   a:bytes pertenecientes a la cabecera

%Parametros de salida
%   picture_structurs
%   frame_pred_frame_dct
%   concealment_motion_vectors
%   f_code
%   precision
%   alternate_scan
%   q_scale_type
%   intra_vlc_format

function [picture_structure,frame_pred_frame_dct,concealment_motion_vectors,f_code,precision, ...
    alternate_scan,q_scale_type,intra_vlc_format]=picture_coding_extension(a)

%Pasar de decimal a binario

a_bin=dec2bin(a);

[num_bytes,num_bits]=size(a_bin);
for(i=1:num_bytes)
    for(j=1:num_bits)
    bitstream(1,(i-1)*num_bits+j)=a_bin(i,j);
    j=j+1;
    end
    i=i+1;
end

k=1;
f_code=uint8([0 0;0 0]);

%Extraer parametros de la extension

bits=32;
extension_start_code=bitstream(k:k+bits-1);
k=k+bits;

bits=4;
extension_start_code_identifier=bitstream(k:k+bits-1);
k=k+bits;

bits=4;
aux=bitstream(k:k+bits-1);
f_code(1,1)=uint8(bin2dec(aux));    %forward horizontal
k=k+bits;

bits=4;
aux=bitstream(k:k+bits-1);
f_code(1,2)=uint8(bin2dec(aux));    %forward vertical
k=k+bits;
```

```
bits=4;
aux=bitstream(k:k+bits-1);
f_code(2,1)=uint8(bin2dec(aux));    %backward horizontal
k=k+bits;

bits=4;
aux=bitstream(k:k+bits-1);
f_code(2,2)=uint8(bin2dec(aux));    %forward vertical
k=k+bits;

bits=2;
aux=bitstream(k:k+bits-1);
intra_dc_precision=uint8(bin2dec(aux));
%intra_dc_precison  Precision(bits)
%   0               8
%   1               9
%   2               10
%   3               11
k=k+bits;

if(intra_dc_precision==0)
    precision=uint8(8);
else if(intra_dc_precision==1)
        precision=uint8(9);
    else if(intra_dc_precision==2)
            precision=uint8(10);
        else
            precision=uint8(11);
        end
    end
end

bits=2;
aux=bitstream(k:k+bits-1);
picture_structure=uint8(bin2dec(aux));
%picture_structure  Meaning
%   0               reserved
%   1               Top Field
%   2               Bottom Field
%   3               Frame Picture
k=k+bits;

bits=1;
aux=bitstream(k:k+bits-1);
top_field_first=uint8(bin2dec(aux));
k=k+bits;

bits=1;
aux=bitstream(k:k+bits-1);
frame_pred_frame_dct=uint8(bin2dec(aux));
k=k+bits;

bits=1;
aux=bitstream(k:k+bits-1);
concealment_motion_vectors=uint8(bin2dec(aux));
k=k+bits;

bits=1;
aux=bitstream(k:k+bits-1);
q_scale_type=uint8(bin2dec(aux));
k=k+bits;

bits=1;
aux=bitstream(k:k+bits-1);
intra_vlc_format=uint8(bin2dec(aux));
k=k+bits;

bits=1;
aux=bitstream(k:k+bits-1);
alternate_scan=uint8(bin2dec(aux));
k=k+bits;

bits=1;
aux=bitstream(k:k+bits-1);
repeat_first_field=uint8(bin2dec(aux));
k=k+bits;

bits=1;
aux=bitstream(k:k+bits-1);
chroma_420_type=uint8(bin2dec(aux));
k=k+bits;

bits=1;
aux=bitstream(k:k+bits-1);
progressive_frame=uint8(bin2dec(aux));
```

```
k=k+bits;

bits=1;
aux=bitstream(k:k+bits-1);
composite_display_flag=uint8(bin2dec(aux));
k=k+bits;

if(composite_display_flag==1)
    bits=1;
    aux=bitstream(k:k+bits-1);
    v_axis=uint8(bin2dec(aux));
    k=k+bits;

    bits=3;
    aux=bitstream(k:k+bits-1);
    field_sequence=uint8(bin2dec(aux));
    %field_sequence     frame   field
    %   0               1       1
    %   1               1       2
    %   2               2       3
    %   3               2       4
    %   4               3       5
    %   5               3       6
    %   6               4       7
    %   7               4       8
    k=k+bits;

    bits=1;
    aux=bitstream(k:k+bits-1);
    sub_carrier=uint8(bin2dec(aux));
    k=k+bits;

    bits=7;
    aux=bitstream(k:k+bits-1);
    burst_amplitude=uint8(bin2dec(aux));
    k=k+bits;

    bits=8;
    aux=bitstream(k:k+bits-1);
    sub_carrier_phase=uint8(bin2dec(aux));
    k=k+bits;
end
```

```
%*************************************************************************************************
%                                          Picture display extension                              %
%                                                          Nº of bits          Mnemonic          %
%   picture_display_extension() {                                                                 %
%       extension_start_code_identifier                          4              uimsbf          %
%       for(i=0;i<number_of_frame_centre_offsets;i++) {                                          %
%           frame_centre_horizontal_offset                      16              simsbf          %
%           marker_bit                                           1              bslbf           %
%           frame_centre_vertical_offset                        16              simsbf          %
%           marker_bit                                           1              bslbf           %
%       }                                                                                       %
%       next_start_code()                                                                       %
%   }                                                                                           %
%*************************************************************************************************

%Parametros de entrada
%   bitstream:bits que pertenecen a la extension

%Parametros de salida
%

function picture_display_extension(bitstream)

k=1;

bits=4;
extension_start_code_identifier=bitstream(k:k+bits-1);
k=k+bits;

for i=1:number_of_frame_centre_offsets
    bits=16;
    aux=bitstream(k:k+bits-1);
    frame_centre_horizontal_offset=int15(bin2dec(aux));
    k=k+bits;

    bits=1;
    marker_bit=bitstream(k:k+bits-1);
    k=k+bits;

    bits=16;
    aux=bitstream(k:k+bits-1);
    frame_centre_vertical_offset=int16(bin2dec(aux));
    k=k+bits;

    bits=1;
    marker_bit=bitstream(k:k+bits-1);
    k=k+bits;
end
```

```
%*************************************************************************************************************
%                                          Picture spatial scalable extension                                 %
%                                                          Nº of bits          Mnemonic                       %
%   picture_spatial_scalabe_extensionn() {                                                                    %
%       extension_start_code_identifier                          4              uimsbf                        %
%       lower_layer_temporal_reference                          10              uimsbf                        %
%       marker_bit                                               1              bslbf                         %
%       lower_layer_horizontal_offset                           15              simsbf                        %
%       maker_bit                                                1              bslbf                         %
%       lower_layer_vertical_offset                             15              simsbf                        %
%       spatial_temporal_weight_code_table_index                 2              uimsbf                        %
%       lower_layer_progressive_frame                            1              uimsbf                        %
%       lower_layer_deinterlaced_field_select                    1              uimsbf                        %
%       next_start_code()                                                                                     %
%   }                                                                                                         %
%*************************************************************************************************************

%Parametros de entrada
%   bitstream:bits que pertenecen a la extension

%Parametros de salida
%

function picture_spatial_scalable_extension(bitstream)

k=1;

bits=4;
extension_start_code_identifier=bitstream(k:k+bits-1);
k=k+bits;

bits=10;
aux=bitstream(k:k+bits-1);
lower_leyer_temporal_reference=uint16(bin2dec(aux));
k=k+bits;

bits=1;
marker_bit
k=k+bits;

bits=15;
aux=bitstream(k:k+bits-1);
lower_layer_hrizontal_offset=int16(bin2dec(aux));
k=k+bits;

bits=1;
marker_bit
k=k+bits;

bits=15;
aux=bitstream(k:k+bits-1);
lower_layer_vertical_offset=int16(bin2dec(aux));
k=k+bits;

bits=2;
aux=bitstream(k:k+bits-1);
spatial_temporal_weight_code_table_index=uint8(bin2dec(aux));
k=k+bits;

bits=1;
aux=bitstream(k:k+bits-1);
lower_layer_progressive_frame=uint8(bin2dec(aux));
k=k+bits;

bits=1;
aux=bitstream(k:k+bits-1);
lower_layer_deinterlaced_fiel_select=uint8(bin2dec(aux));
k=k+bits;
```

```
%*************************************************************************************************************
%                                          Picture temporal scalable extension                                %
%                                                          Nº of bits          Mnemonic                       %
%   picture_temporal_scalabe_extensionn() {                                                                   %
%       extension_start_code_identifier                          4              uimsbf                        %
%       reference_select_code                                    2              uimsbf                        %
%       forward_temporal_reference                              10              uimsbf                        %
%       marker_bit                                               1              bslbf                         %
%       backward_temporal_reference                             10              uimsbf                        %
%       next_start_code()                                                                                     %
%   }                                                                                                         %
%*************************************************************************************************************

%Parametros de entrada
%   bitstream:bits que pertenecen a la extension

%Parametros de salida
%

function picture_temporal_scalable_extension(bitstream)
k=1;

bits=4;
extension_start_code_idetifier=bitstream(k:k+bits-1);
k=k+bits;

bits=2;
aux=bitstream(k:k+bits-1);
reference_select_code=uint8(bin2dec(aux));
k=k+bits;

bits=10;
aux=bitstream(k:k+bits-1);
forward_temporal_reference=uint16(bin2dec(aux));
k=k+bits;

bits=1;
marker_bit
k=k+bits;

bits=10;
aux=bitstream(k:k+bits-1);
backward_temporal_reference=uint16(bin2dec(aux));
k=k+bits;
```

```
%*************************************************************************************************************
%                                              Slice                                                        %
%                                                                  Nº of bits          Mnemonic   %
%   slice() {                                                                                               %
%       slice_start_code                                               32              bslbf      %
%       if(vertical_size>2800)                                                                          %
%           slice_vertical_pisition_extension                           3              uimsbf     %
%       if(<sequence_scalable_extension() is present in the bitstream>) {                               %
%           if(scalable_mode=="data partitioning")                                                      %
%               priority_breakpoint                                     7              uimsbf     %
%       }                                                                                               %
%       quantiser_scale_code                                            5              uimsbf     %
%       if(nextbits()=='1') {                                                                           %
%           intra_slice_flag                                            1              bslbf      %
%           intra_slice                                                 1              uimsbf     %
%           reserved_bits                                               7              uimsbf     %
%           while(nextbits()=='1')                                                                      %
%               extra_bit_slice     /* with the value '1' */            1              uimsbf     %
%               extra_infomation_slice                                  8              uimsbf     %
%           }                                                                                           %
%       }                                                                                               %
%       extra_bit_slice             /* with the value '0' */            1              uimsbf     %
%       do {                                                                                            %
%           macroblock()                                                                                %
%       } while(nexbits()!='000 0000 0000 0000 0000 0000                                                %
%       next_start_code()                                                                               %
%   }                                                                                                   %
%*************************************************************************************************************

%Parametros de entrada
%   a: bytes pertenecientes a la slice
%   horizontal_size
%   vertical_size
%   mb_width
%   picture_coding_type
%   picture_structure
%   frame_pred_frame_dct
%   concealment_motion_vectors
%   chroma_format
%   f_code
%   precision
%   alternate_scan
%   intra_quantiser_matrix
%   non_intra_quantiser_matrix
%   q_scale_type
%   intra_vlc_format
%   im_referencia

%Parametros de salida
%   slice_luma
%   slice_chroma_b
%   slice_chroma_r
%   mb_row
%   mb_column

function [slice_luma,slice_chroma_b,slice_chroma_r,mb_row,mb_column]=slice(a,horizontal_size, ...
    vertical_size,mb_width,picture_coding_type,picture_structure,frame_pred_frame_dct, ...
    concealment_motion_vectors,chroma_format,f_code,precision,alternate_scan,intra_quantiser_matrix, ...
    non_intra_quantiser_matrix,q_scale_type,intra_vlc_format,im_referencia)

%INICIALIZACION PREDICTOR COEFICIENTES DC
if(precision==8)
    dc_dct_pred=int16([128 128 128]);
else if(precision==9)
        dc_dct_pred=int16([256 256 256]);
    else if(precision==10)
            dc_dct_pred=int16([512 512 512]);
        else
            dc_dct_pred=int16([1024 1024 1024]);
        end
    end
end

%RESET PREDICTOR motion vector
%four motion vector predictors with horizontal and vertical component
%               PMV[r][s][t], vector[r][s][t] and vector'[r][s][t]
%                   0                           1
%   r       First motion vector in MC   Second motion vector in MC
%   s       Forward motion vector       Backward motion vector
%   t       Horizontal Component
r=uint8([1 2]);
s=uint8([1 2]);
t=uint8([1 2]);
PMV(r,s,t)=0;
```

```
%INICIALIZACION DIRECCIONES MACROBLOCK
previous_macroblock_address=int16(-1);

% Pasar de decimal a binario

a_bin=dec2bin(a);
[num_bytes,num_bits]=size(a_bin);
for(i=1:num_bytes)
    for(j=1:num_bits)
    bitstream(1,(i-1)*num_bits+j)=a_bin(i,j);
    j=j+1;
    end
    i=i+1;
end

k=1;

%Extraer parametros y resto funcion

bits=32;
slice_start_code=bitstream(k:k+bits-1);
bits=8;
aux=bitstream(k+24:k+24+bits-1);
slice_vertical_position=uint8(bin2dec(aux));
k=k+bits+24;

if (vertical_size>2800)
    bits=3;
    aux=bitstream(k:k+bits-1);
    slice_vertical_position_extension=uint8(bin2dec(aux));
    mb_row=uint16(uint8(slice_vertical_position_extension*(2^8))+uint8(slice_vertical_position));
    k=k+bits;

else
    mb_row=uint16(slice_vertical_position);
end

sequence_scalable_extension=0;
if(sequence_scalable_extension==1)
    bits=7;
    aux=bitstream(k:k+bits-1);
    priority_breakpoint=uint8(bin2dec(aux));
    k=k+bits;
end

bits=5;
aux=bitstream(k:k+bits-1);
quantiser_scale_code=uint8(bin2dec(aux));
k=k+bits;

if(bitstream(k)=='1')
    bits=1;
    intra_slice_flag=bitstream(k:k+bits-1);
    k=k+bits;

    bits=1;
    aux=bitstream(k:k+bits-1);
    intra_slice=uint8(bin2dec(aux));
    k=k+bits;

    bits=7;
    aux=bitstream(k:k+bits-1);
    reserved_bits=uint8(bin2dec(aux));
    k=k+bits;

    while(bitstream(k)=='1')
        bits=1;
        aux=bitstream(k:k+bits-1);
        extra_bit_slice=uint8(bin2dec(aux));
        k=k+bits;

        bits=8;
        aux=bitstream(k:k+bits-1);
        extra_information_slice=uint8(bin2dec(aux));
        k=k+bits;
    end
end

bits=1;
aux=bitstream(k:k+bits-1);
extra_bit_slice=uint8(bin2dec(aux));
k=k+bits;

%do {
```

```
%Parametros empleados para el caso de macroblock skipped. Se inicializan a 0 ya que el primer macrobloque
%siempre está codificado
half_flag=0;
int_vec=0;
macroblock_motion_forward=uint8(0);
macroblock_motion_backward=uint8(0);

%macroblock()
[macro_luma,macro_chroma_b,macro_chroma_r,bits,dc_dct_pred,PMV,mb_column,macroblock_address,half_flag, ...
    int_vec,macroblock_motion_forward, ...
    macroblock_motion_backward]=macroblock(bitstream(k:num_bytes*num_bits),picture_coding_type, ...
    picture_structure,frame_pred_frame_dct,concealment_motion_vectors,chroma_format,f_code,precision, ...
    dc_dct_pred,alternate_scan,intra_quantiser_matrix,non_intra_quantiser_matrix,q_scale_type, ...
    quantiser_scale_code,intra_vlc_format,PMV,im_referencia,mb_width,mb_row,previous_macroblock_address, ...
    half_flag,int_vec,macroblock_motion_forward,macroblock_motion_backward);

k=k+bits;

%Actualización direcciones
previous_macroblock_address=macroblock_address;

pos_x=(mb_column-1)*16+1;   %Posición del primer pixel de cada macro

%Formacion de slices
slice_luma(1:16,pos_x:pos_x+15)=macro_luma;
slice_chroma_b(1:16,pos_x:pos_x+15)=macro_chroma_b;
slice_chroma_r(1:16,pos_x:pos_x+15)=macro_chroma_r;

fin_slice=0;

%} while (nextbits!= '000 0000 0000 0000 0000 0000')
while(fin_slice==0)
    bits=23;
    if(bitstream(k:k+bits-1)=='00000000000000000000000')
        fin_slice=1;
    else
        [macro_luma,macro_chroma_b,macro_chroma_r,bits,dc_dct_pred,PMV,mb_column,macroblock_address, ...
            half_flag,int_vec,macroblock_motion_forward, ...
            macroblock_motion_backward]=macroblock(bitstream(k:num_bytes*num_bits),picture_coding_type, ...
            picture_structure,frame_pred_frame_dct,concealment_motion_vectors,chroma_format, ...
            f_code,precision,dc_dct_pred,alternate_scan,intra_quantiser_matrix, ...
            non_intra_quantiser_matrix,q_scale_type,quantiser_scale_code,intra_vlc_format,PMV, ...
            im_referencia,mb_width,mb_row,previous_macroblock_address,half_flag,int_vec, ...
            macroblock_motion_forward,macroblock_motion_backward);
            %si hay skipped macroblock devuelve todos hasta el siguiente no
            %skipped

        k=k+bits;

        %Actualización direcciones
        previous_macroblock_address=macroblock_address;

        pos_x=(previous_macroblock_address+1)*16+1; %Posición del primer pixel de cada macro

        tam=size(macro_luma);   %Si ha habido macrobloques skipped la funcion macroblock() devuelve varios
                                %macrobloques

        %Formacion de slices
        slice_luma(1:16,pos_x:pos_x+tam(2)-1)=macro_luma;
        slice_chroma_b(1:16,pos_x:pos_x+tam(2)-1)=macro_chroma_b;
        slice_chroma_r(1:16,pos_x:pos_x+tam(2)-1)=macro_chroma_r;
    end
end
```

```
%*************************************************************************************************************
%                                          Macroblock                                                       %
%                                                                  Nº of bits          Mnemonic              %
%   macroblock() {                                                                                          %
%       while(nextbits()=='0000 0001 000')                                                                  %
%           macroblock_escape                                        11                bslbf               %
%       macroblock_address_increment                                1-11               vlclbf              %
%       macroblock_modes                                                                                    %
%       if(macroblock_quant)                                                                                %
%           quantiser_scale_code                                     5                 uimsbf              %
%       if(macroblock_motion_forwardd||                                                                     %
%           (macroblock_intra && concealment_motion_vectors))                                               %
%           motion_vectors(0)                                                                               %
%       if(macroblock_motion_backward)                                                                      %
%           motion_vectors(1)                                                                               %
%       if(macroblock_intra && concealment_motion_vectors)                                                  %
%           marker_bit                                               1                 bslbf               %
%       if(macroblock_pattern)                                                                              %
%           coded_block_pattern()                                                                           %
%       for (i=0;i<block_count;i++) {                                                                       %
%           block(i)                                                                                        %
%       }                                                                                                   %
%   }                                                                                                       %
%*************************************************************************************************************

%Parametros de entrada
%   bitstream
%   picture_coding_type
%   picture_structure
%   frame_pred_frame_dct
%   concealment_motion_vectors
%   chroma_format
%   f_code
%   precision
%   dc_dct_pred
%   alternate_scan
%   intra_quantiser_matrix
%   non_intra_quantiser_matrix
%   q_scale_type
%   quantiser_scale_code
%   intra_vlc_format
%   PMV
%   im_referncia
%   mb_width
%   mb_row
%   previous_macroblock_address
%   half_flag
%   int_vec
%   macroblock_motion_forward
%   macroblock_motion_backward

%Parametros de salida
%   macro_luma
%   macro_chroma_b
%   macro_chroma_r
%   bits_leidos
%   dc_dct_pred
%   PMV
%   mb_column
%   macroblock_address
%   half_flag
%   int_vec
%   macroblock_motion_forward
%   macroblock_motion_backward

function [macro_luma,macro_chroma_b,macro_chroma_r,bits_leidos,dc_dct_pred,PMV,mb_column, ...
    macroblock_address,half_flag,int_vec,macroblock_motion_forward, ...
    macroblock_motion_backward]=macroblock(bitstream,picture_coding_type,picture_structure, ...
    frame_pred_frame_dct,concealment_motion_vectors,chroma_format,f_code,precision,dc_dct_pred, ...
    alternate_scan,intra_quantiser_matrix,non_intra_quantiser_matrix,q_scale_type,quantiser_scale_code, ...
    intra_vlc_format,PMV,im_referencia,mb_width,mb_row,previous_macroblock_address,half_flag,int_vec, ...
    macroblock_motion_forward,macroblock_motion_backward)

k=1;

long_bitstream=size(bitstream);

bits=11;
while(bitstream(k:k+bits-1)=='00000001000')
    macroblock_escape=bitstream(k:k+bits-1);
    k=k+bits;
end

table_b1; %Variable lenght codes for macroblock_address_increment
i=1;
```

```
flag=0;

while(flag==0)
    if(bitstream(k:k+num_bits(i)-1)==cod_long_var(i,1:num_bits(i)))
        macroblock_address_increment=increment_value(i);
        k=k+num_bits(i);
        flag=1;
    else
        i=i+1;
    end
end

%Calculo variables necesarias
macroblock_address=previous_macroblock_address+int16(macroblock_address_increment);
mb_column=mod((macroblock_address+1),int16(mb_width)); %numero de macroblock
num_macroblock_skipped=int16(macroblock_address-previous_macroblock_address-1);

if(mb_column==0)
    mb_column=mb_width;
end

if(chroma_format==1) %4:2:0
    block_count=uint8(6);
else if(chroma_format==2) %4:2:2
        block_count=uint8(8);
    else if(chroma_format==3) %4:4:4
            block_count=uint8(12);
        else
            block_count=uint8(0);
        end
    end
end

num_mb=int16(0);

%Compruebo si hay skipped macroblock y los decodifico en primer lugar con
%los parametros del macroblock anterior
if(macroblock_address_increment~=1)
    if(picture_coding_type==2) %Imagen P
        if(picture_structure==3)    %Imagen trama
            frame_motion_type='10';
            motion_vector_count=1;
            mv_format='frame';
            dmv=0;

            PMV(:,:,:)=0;

            vector_Y(1:2,1:2,1:2)=0;
            vector_Cb(1:2,1:2,1:2)=0;
            vector_Cr(1:2,1:2,1:2)=0;

            int_vec(1:2,1:2,1:2,1:3)=0;
            half_flag(1:2,1:2,1:2,1:3)=0;
        else
            field_motion_type='01';
            motion_vector_count=1;
            mv_format='field';
            dmv=0;

            PMV(:,:,:)=0;

            int_vec(1:2,1:2,1:2,1:3)=0;
            half_flag(1:2,1:2,1:2,1:3)=0;
        end
    else if(picture_coding_type==3) %Imagen B
            if(picture_structure==3) %Imagen trama
                frame_motion_type='10';
                motion_vector_count=1;
                mv_format='frame';
                dmv=0;

                PMV(:,:,:)=PMV;

                int_vec=int_vec;
                half_flag=half_flag;
            else
                field_motion_type='01';
                motion_vector_count=1;
                mv_format='field';
                dmv=0;

                PMV(:,:,:)=PMV;

                int_vec=int_vec;
                half_flag=half_flag;
```

```
            end
        end
    end
    for num_mb=1:num_macroblock_skipped
        mb_column_sk=mb_column-num_macroblock_skipped+num_mb-1;
        [macros_luma(:,:,num_mb),macros_chroma_b(:,:,num_mb),...
            macros_chroma_r(:,:,num_mb)]=macroblock_skipped(block_count,mb_row,mb_column_sk,...
            im_referencia,chroma_format,half_flag,int_vec,macroblock_motion_forward,...
            macroblock_motion_backward);
    end

    if(precision==8)
        dc_dct_pred=int16([128 128 128]);
    else if(precision==9)
            dc_dct_pred=int16([256 256 256]);
        else if(precision==10)
                dc_dct_pred=int16([512 512 512]);
            else
                dc_dct_pred=int16([1024 1024 1024]);
            end
        end
    end
end

num_mb=num_mb+1; %Contador de macrobloques

[bits,macroblock_quant,macroblock_motion_forward,macroblock_motion_backward,macroblock_pattern,...
    macroblock_intra,frame_motion_type,motion_vector_count,mv_format,dmv]=...
    macroblock_modes(bitstream(k:long_bitstream(2)),picture_coding_type,picture_structure,...
    frame_pred_frame_dct);
k=k+bits;

%RESETS

%reseteo predictor coeficientes dc
if(macroblock_intra==0)
    if(precision==8)
        dc_dct_pred=int16([128 128 128]);
    else if(precision==9)
            dc_dct_pred=int16([256 256 256]);
        else if(precision==10)
            dc_dct_pred=int16([512 512 512]);
            else
                dc_dct_pred=int16([1024 1024 1024]);
            end
        end
    end
end

%Resetting motion vector predictors
if((macroblock_intra==0)&(macroblock_motion_forward==0))
    %four motion vector predictors with horizontal and vertical component
    %               PMV[r][s][t], vector[r][s][t] and vector'[r][s][t]
    %                   0                               1
    %   r       First motion vector in MC   Second motion vector in MC
    %   s       Forward motion vector       Backward motion vector
    %   t       Horizontal Component
    r=uint8([1 2]);
    s=uint8([1 2]);
    t=uint8([1 2]);

    PMV(r,s,t)=0;

    vector_Y(1:2,1,1:2)=0;
    vector_Cb(1:2,1,1:2)=0;
    vector_Cr(1:2,1,1:2)=0;
end

if((macroblock_intra==1)&(concealment_motion_vectors==0))
    %four motion vector predictors with horizontal and vertical component
    %               PMV[r][s][t], vector[r][s][t] and vector'[r][s][t]
    %                   0                               1
    %   r       First motion vector in MC   Second motion vector in MC
    %   s       Forward motion vector       Backward motion vector
    %   t       Horizontal Component
    r=uint8([1 2]);
    s=uint8([1 2]);
    t=uint8([1 2]);

    PMV(r,s,t)=0;
end

vector_Y(1:2,1:2,1:2)=0;
vector_Cb(1:2,1:2,1:2)=0;
vector_Cr(1:2,1:2,1:2)=0;
```

```
if(macroblock_quant==1)
    bits=5;
    aux=bitstream(k:k+bits-1);
    quantiser_scale_code=uint8(bin2dec(aux));
    k=k+bits;
end

if((macroblock_motion_forward==1) | ((macroblock_intra==1) & (concealment_motion_vectors==1)))
    s=uint8(1); %Forward Motion Vector (en la norma es s=0)
    [bits,vector_Y(1:2,s,1:2),vector_Cb(1:2,s,1:2),vector_Cr(1:2,s,1:2),PMV]=motion_vectors(s, ...
        bitstream(k:long_bitstream(2)),motion_vector_count,mv_format,dmv,f_code,PMV,picture_structure, ...
        chroma_format);
    k=k+bits;
end

if(macroblock_motion_backward==1)
    s=uint8(2);    %Backward Motion Vector (en la norma es s=1)
    [bits,vector_Y(1:2,s,1:2),vector_Cb(1:2,s,1:2),vector_Cr(1:2,s,1:2),PMV]=motion_vectors(s, ...
        bitstream(k:long_bitstream(2)),motion_vector_count,mv_format,dmv,f_code,PMV,picture_structure, ...
        chroma_format);
    k=k+bits;
end

cero(1:2,1:2,1:2)=uint8(0);
if(vector_Y==cero)
    flag=0;
else
    flag=1;
end
if(flag==1)
    for r=1:2
        for s=1:2
            for t=1:2
                int_vec(r,s,t,1)=floor(vector_Y(r,s,t)/2);
                if((vector_Y(r,s,t)-(2*int_vec(r,s,t,1))==0))
                    half_flag(r,s,t,1)=0;
                else
                    half_flag(r,s,t,1)=1;
                end
                int_vec(r,s,t,2)=floor(vector_Cb(r,s,t)/2);
                if((vector_Cb(r,s,t)-(2*int_vec(r,s,t,2))==0))
                    half_flag(r,s,t,2)=0;
                else
                    half_flag(r,s,t,2)=1;
                end
                int_vec(r,s,t,3)=floor(vector_Cr(r,s,t)/2);
                if((vector_Cr(r,s,t)-(2*int_vec(r,s,t,3))==0))
                    half_flag(r,s,t,3)=0;
                else
                    half_flag(r,s,t,3)=1;
                end
            end
        end
    end
else
    int_vec(1:2,1:2,1:2,1:3)=0;
    half_flag(1:2,1:2,1:2,1:3)=0;
end

if((macroblock_intra==1) & (concealment_motion_vectors==1))
    bits=1;
    %marker_bit;
    k=k+bits;
end

if(macroblock_pattern==1)
    [bits,cbp]=coded_block_pattern(bitstream(k:long_bitstream(2)),chroma_format);
    k=k+bits;
end

%Calculo de pattern_code
for i=1:block_count
    if(macroblock_intra==1)
        pattern_code(i)=uint8(1);
    else
        pattern_code(i)=uint8(0);
    end
end

if(macroblock_pattern==1)
    for i=1:6
        comparo=dec2bin(bitshift(1,6-i));
        comparo=uint8(comparo)-48;
        switch i
```

```
            case 1
                comparo=comparo;
            case 2
                comparo=[0,comparo];
            case 3
                comparo=[0,0,comparo];
            case 4
                comparo=[0,0,0,comparo];
            case 5
                comparo=[0,0,0,0,comparo];
            case 6
                comparo=[0,0,0,0,0,comparo];
        end
        cbp_bin_char=dec2bin(cbp,6);
        cbp_bin=uint8(cbp_bin_char)-48;
        cero=[0,0,0,0,0,0];
        flag=0;
        if(bitand(cbp_bin,comparo)==cero)
            pattern_code(i)=0;
        else
            pattern_code(i)=1;
        end
    end
end

%Calculo de cada bloque
for i=1:block_count
    aux=bitstream(k:long_bitstream(2));
    [f,bits,dc_dct_pred]=block(i,aux,pattern_code(i),macroblock_intra,dc_dct_pred,alternate_scan, ...
        precision,intra_quantiser_matrix,non_intra_quantiser_matrix,q_scale_type,quantiser_scale_code, ...
        intra_vlc_format);
    k=k+bits;

    x=1:8;
    y=1:8;

    if(macroblock_intra==1)
        p(y,x)=int16(0);    %No hay prediccion
    %Definicion de pel_ref. Matriz formada por el rango de la matriz de referencia que corresponda y
    %rrellenada de ceros alrededor para posibilitar predicciones
    else if(macroblock_intra==0)
            if(i<=4)
                switch i
                    case 1
                        y_ref=double((mb_row-1)*16+1+16);
                        x_ref=double((mb_column-1)*16+1+16);
                    case 2
                        y_ref=double((mb_row-1)*16+1+16);
                        x_ref=double((mb_column-1)*16+8+1+16);
                    case 3
                        y_ref=double((mb_row-1)*16+8+1+16);
                        x_ref=double((mb_column-1)*16+1+16);
                    case 4
                        y_ref=double((mb_row-1)*16+8+1+16);
                        x_ref=double((mb_column-1)*16+8+1+16);
                end
                %Relleno con ceros alrededor
                pel_ref(1:16,1:16,:)=int16(0);
                pel_ref(257:272,337:352,:)=int16(0);
                pel_ref(17:256,17:336,:)=im_referencia(:,:,1,:);
            else if(chroma_format==1)   %4:2:0
                    switch i
                        case 5
                            y_ref=double((mb_row-1)*8+1+8);
                            x_ref=double((mb_column-1)*8+1+8);
                            for m=1:240
                                for n=1:320
                                    fila=floor(m/2)+1;
                                    columna=floor(n/2)+1;
                                    pel_ref(fila,columna,:)=im_referencia(m,n,2,:);

                                end
                            end
                            pel_ref(9:128,9:168,:)=pel_ref;
                            pel_ref(1:8,1:8,:)=int16(0);
                            pel_ref(129:136,169:176,:)=int16(0);
                        case 6
                            y_ref=double((mb_row-1)*8+1+8);
                            x_ref=double((mb_column-1)*8+1+8);
                            for m=1:240
                                for n=1:320
                                    fila=floor(m/2)+1;
                                    columna=floor(n/2)+1;
                                    pel_ref(fila,columna,:)=im_referencia(m,n,3,:);
                                end
```

```
                end
                pel_ref(9:128,9:168,:)=pel_ref;
                pel_ref(1:8,1:8,:)=int16(0);
                pel_ref(129:136,169:176,:)=int16(0);
        end
    else if(chroma_format==2)   %4:2:2
            switch i
                case 5
                    y_ref=double((mb_row-1)*8+1+8);
                    x_ref=double((mb_column-1)*8+1+8);
                    for m=1:240
                        for  n=1:320
                            fila=m;
                            columna=floor(n/2)+1;
                            pel_ref(fila,columna,:)=im_referencia(m,n,2,:);
                        end
                    end
                    pel_ref(9:128,9:168,:)=pel_ref;
                    pel_ref(1:8,1:8,:)=int16(0);
                    pel_ref(129:136,169:176,:)=int16(0);
                case 6
                    y_ref=double((mb_row-1)*8+1+8);
                    x_ref=double((mb_column-1)*8+1+8);
                    for m=1:240
                        for  n=1:320
                            fila=m;
                            columna=floor(n/2)+1;
                            pel_ref(fila,columna,:)=im_referencia(m,n,3,:);
                        end
                    end
                    pel_ref(9:128,9:168,:)=pel_ref;
                    pel_ref(1:8,1:8,:)=int16(0);
                    pel_ref(129:136,169:176,:)=int16(0);
                case 7
                    y_ref=double((mb_row-1)*8+8+1+8);
                    x_ref=double((mb_column-1)*8+1+8);
                    for m=1:240
                        for  n=1:320
                            fila=m;
                            columna=floor(n/2)+1;
                            pel_ref(fila,columna,:)=im_referencia(m,n,2,:);
                        end
                    end
                    pel_ref(9:128,9:168,:)=pel_ref;
                    pel_ref(1:8,1:8,:)=int16(0);
                    pel_ref(129:136,169:176,:)=int16(0);
                case 8
                    y_ref=double((mb_row-1)*8+8+1+8);
                    x_ref=double((mb_column-1)*8+1+8);
                    for m=1:240
                        for  n=1:320
                            fila=m;
                            columna=floor(n/2)+1;
                            pel_ref(fila,columna,:)=im_referencia(m,n,3,:);
                        end
                    end
                    pel_ref(9:128,9:168,:)=pel_ref;
                    pel_ref(1:8,1:8,:)=int16(0);
                    pel_ref(129:136,169:176,:)=int16(0);
            end
        else if(chroma_format==3)   %4:4:4
                switch i
                    case 5
                        y_ref=double((mb_row-1)*8+1+8);
                        x_ref=double((mb_column-1)*8+1+8);
                        pel_ref(:,:,:)=im_referencia(:,:,2,:);
                    case 6
                        y_ref=double((mb_row-1)*8+1+8);
                        x_ref=double((mb_column-1)*8+1+8);
                        pel_ref(:,:,:)=im_referencia(:,:,3,:);
                    case 7
                        y_ref=double((mb_row-1)*8+8+1+8);
                        x_ref=double((mb_column-1)*8+1+8);
                        pel_ref(:,:,:)=im_referencia(:,:,2,:);
                    case 8
                        y_ref=double((mb_row-1)*8+8+1+8);
                        x_ref=double((mb_column-1)*8+1+8);
                        pel_ref(:,:,:)=im_referencia(:,:,3,:);
                    case 9
                        y_ref=double((mb_row-1)*8+1+8);
                        x_ref=double((mb_column-1)*8+8+1+8);
                        pel_ref(:,:,:)=im_referencia(:,:,2,:);
                    case 10
                        y_ref=double((mb_row-1)*8+1+8);
                        x_ref=double((mb_column-1)*8+8+1+8);
```

```
                        pel_ref(:,:,:)=im_referencia(:,:,3,:);
                    case 11
                        y_ref=double((mb_row-1)*8+8+1+8);
                        x_ref=double((mb_column-1)*8+8+1+8);
                        pel_ref(:,:,:)=im_referencia(:,:,2,:);
                    case 12
                        y_ref=double((mb_row-1)*8+8+1+8);
                        x_ref=double((mb_column-1)*8+8+1+8);
                        pel_ref(:,:,:)=im_referencia(:,:,3,:);
                end
                pel_ref(9:128,9:168,:)=pel_ref;
                pel_ref(1:8,1:8,:)=int16(0);
                pel_ref(129:136,169:176,:)=int16(0);
            end
        end
    end
end

%Predicción forward
if(macroblock_motion_forward==1)
    if(i<=4)
        if((half_flag(1,1,1,1)==0)&(half_flag(1,1,2,1)==0))
            pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,1):...
                y_ref+int_vec(1,1,2,1)+7,x_ref+int_vec(1,1,1,1):...
                x_ref+int_vec(1,1,1,1)+7,1);
        end
        if((half_flag(1,1,1,1)==0)&(half_flag(1,1,2,1)==1))
            pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,1):...
                y_ref+int_vec(1,1,2,1)+7,x_ref+int_vec(1,1,1,1):...
                x_ref+int_vec(1,1,1,1)+7,1)+pel_ref(y_ref+int_vec(1,1,2,1)+1:...
                y_ref+int_vec(1,1,2,1)+1+7,x_ref+int_vec(1,1,1,1):...
                x_ref+int_vec(1,1,1,1)+7,1))/2);
        end
        if((half_flag(1,1,1,1)==1)&(half_flag(1,1,2,1)==0))
            pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,1):...
                y_ref+int_vec(1,1,2,1)+7,x_ref+int_vec(1,1,1,1):...
                x_ref+int_vec(1,1,1,1)+7,1)+pel_ref(y_ref+int_vec(1,1,2,1):...
                y_ref+int_vec(1,1,2,1)+7,x_ref+int_vec(1,1,1,1)+1:...
                x_ref+int_vec(1,1,1,1)+7+1,1))/2);
        end
        if((half_flag(1,1,1,1)==1)&(half_flag(1,1,2,1)==1))
            pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,1):...
                y_ref+int_vec(1,1,2,1)+7,x_ref+int_vec(1,1,1,1):...
                x_ref+int_vec(1,1,1,1)+7,1)+pel_ref(y_ref+int_vec(1,1,2,1):...
                y_ref+int_vec(1,1,2,1)+7,x_ref+int_vec(1,1,1,1)+1:...
                x_ref+int_vec(1,1,1,1)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,1)+1:...
                y_ref+int_vec(1,1,2,1)+1+7,x_ref+int_vec(1,1,1,1):...
                x_ref+int_vec(1,1,1,1)+7,1)+pel_ref(y_ref+int_vec(1,1,2,1)+1:...
                y_ref+int_vec(1,1,2,1)+1+7,x_ref+int_vec(1,1,1,1)+1:...
                x_ref+int_vec(1,1,1,1)+1+7,1))/4);
        end
    else if(chroma_format==1)
            switch i
                case 5
                    if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==0))
                        pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,2):...
                            y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                            x_ref+int_vec(1,1,1,2)+7,1);
                    end
                    if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==1))
                        pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                            y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                            x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1:...
                            y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2):...
                            x_ref+int_vec(1,1,1,2)+7,1))/2);
                    end
                    if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==0))
                        pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                            y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                            x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2):...
                            y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1:...
                            x_ref+int_vec(1,1,1,2)+7+1,1))/2);
                    end
                    if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==1))
                        pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                            y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                            x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2):...
                            y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1:...
                            x_ref+int_vec(1,1,1,2)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1:...
                            y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2):...
                            x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1:...
                            y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2)+1:...
                            x_ref+int_vec(1,1,1,2)+1+7,1))/4);
                    end
                case 6
```

```
                    if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==0))
                        pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,3): ...
                            y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                            x_ref+int_vec(1,1,1,3)+7,1);
                    end
                    if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==1))
                        pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                            y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                            x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                            y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3): ...
                            x_ref+int_vec(1,1,1,3)+7,1))/2);
                    end
                    if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==0))
                        pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                            y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                            x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3): ...
                            y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1: ...
                            x_ref+int_vec(1,1,1,3)+7+1,1))/2);
                    end
                    if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==1))
                        pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                            y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                            x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3): ...
                            y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1: ...
                            x_ref+int_vec(1,1,1,3)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                            y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3): ...
                            x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                            y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3)+1: ...
                            x_ref+int_vec(1,1,1,3)+1+7,1))/4);
                    end
                end
            else if(chroma_format==2)
                    switch i
                        case 5
                            if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==0))
                                pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,2): ...
                                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                                    x_ref+int_vec(1,1,1,2)+7,1);
                            end
                            if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==1))
                                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2): ...
                                    x_ref+int_vec(1,1,1,2)+7,1))/2);
                            end
                            if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==0))
                                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2): ...
                                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1: ...
                                    x_ref+int_vec(1,1,1,2)+7+1,1))/2);
                            end
                            if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==1))
                                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2): ...
                                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1: ...
                                    x_ref+int_vec(1,1,1,2)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2): ...
                                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2)+1: ...
                                    x_ref+int_vec(1,1,1,2)+1+7,1))/4);
                            end
                        case 6
                            if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==0))
                                pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,3): ...
                                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                                    x_ref+int_vec(1,1,1,3)+7,1);
                            end
                            if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==1))
                                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3): ...
                                    x_ref+int_vec(1,1,1,3)+7,1))/2);
                            end
                            if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==0))
                                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3): ...
                                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1: ...
                                    x_ref+int_vec(1,1,1,3)+7+1,1))/2);
                            end
                            if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==1))
```

```
                        pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                            y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                            x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3): ...
                            y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1: ...
                            x_ref+int_vec(1,1,1,3)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                            y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3): ...
                            x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                            y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3)+1: ...
                            x_ref+int_vec(1,1,1,3)+1+7,1))/4);
                    end
                case 7
                    if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==0))
                        pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,2): ...
                            y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                        x_ref+int_vec(1,1,1,2)+7,1);
                    end
                    if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==1))
                        pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                            y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                            x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                            y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2): ...
                            x_ref+int_vec(1,1,1,2)+7,1))/2);
                    end
                    if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==0))
                        pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                            y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                            x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2): ...
                            y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1: ...
                            x_ref+int_vec(1,1,1,2)+7+1,1))/2);
                    end
                    if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==1))
                        pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                            y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                            x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2): ...
                            y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1: ...
                            x_ref+int_vec(1,1,1,2)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                            y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2): ...
                            x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                            y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2)+1: ...
                            x_ref+int_vec(1,1,1,2)+1+7,1))/4);
                    end
                case 8
                    if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==0))
                        pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,3): ...
                            y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                            x_ref+int_vec(1,1,1,3)+7,1);
                    end
                    if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==1))
                        pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                            y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                            x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                            y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3): ...
                            x_ref+int_vec(1,1,1,3)+7,1))/2);
                    end
                    if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==0))
                        pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                            y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                            x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3): ...
                            y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1: ...
                            x_ref+int_vec(1,1,1,3)+7+1,1))/2);
                    end
                    if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==1))
                        pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                            y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                            x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3): ...
                            y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1: ...
                            x_ref+int_vec(1,1,1,3)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                            y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3): ...
                            x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                            y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3)+1: ...
                        x_ref+int_vec(1,1,1,3)+1+7,1))/4);
                    end
            end

        else if(chroma_format==3)
                switch i
                    case 5
                        if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==0))
                            pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,2): ...
                                y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                                x_ref+int_vec(1,1,1,2)+7,1);
                        end
                        if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==1))
                            pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                                y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
```

```
                x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2): ...
                x_ref+int_vec(1,1,1,2)+7,1))/2);
            end
            if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==0))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2): ...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1: ...
                    x_ref+int_vec(1,1,1,2)+7+1,1))/2);
            end
            if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2): ...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1: ...
                    x_ref+int_vec(1,1,1,2)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2): ...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2)+1: ...
                    x_ref+int_vec(1,1,1,2)+1+7,1))/4);
            end
        case 6
            if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==0))
                pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,3): ...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                    x_ref+int_vec(1,1,1,3)+7,1);
            end
            if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3): ...
                    x_ref+int_vec(1,1,1,3)+7,1))/2);
            end
            if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==0))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3): ...
                y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1: ...
                x_ref+int_vec(1,1,1,3)+7+1,1))/2);
            end
            if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3): ...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1: ...
                    x_ref+int_vec(1,1,1,3)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3): ...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3)+1: ...
                    x_ref+int_vec(1,1,1,3)+1+7,1))/4);
            end
        case 7
            if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==0))
                pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,2): ...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                    x_ref+int_vec(1,1,1,2)+7,1);
            end
            if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2): ...
                    x_ref+int_vec(1,1,1,2)+7,1))/2);
            end
            if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==0))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2): ...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1: ...
                    x_ref+int_vec(1,1,1,2)+7+1,1))/2);
            end
            if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2): ...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1: ...
                    x_ref+int_vec(1,1,1,2)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2): ...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2)+1: ...
                    x_ref+int_vec(1,1,1,2)+1+7,1))/4);
            end
        case 8
```

```
            if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==0))
                pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,3): ...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                    x_ref+int_vec(1,1,1,3)+7,1);
            end
            if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3): ...
                    x_ref+int_vec(1,1,1,3)+7,1))/2);
            end
            if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==0))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3): ...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1: ...
                    x_ref+int_vec(1,1,1,3)+7+1,1))/2);
            end
            if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3): ...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1: ...
                    x_ref+int_vec(1,1,1,3)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3): ...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3)+1: ...
                    x_ref+int_vec(1,1,1,3)+1+7,1))/4);
            end
        case 9
            if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==0))
                pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,2): ...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                    x_ref+int_vec(1,1,1,2)+7,1);
            end
            if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2): ...
                    x_ref+int_vec(1,1,1,2)+7,1))/2);
            end
            if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==0))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2): ...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1: ...
                    x_ref+int_vec(1,1,1,2)+7+1,1))/2);
            end
            if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2): ...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1: ...
                    x_ref+int_vec(1,1,1,2)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2): ...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2)+1: ...
                    x_ref+int_vec(1,1,1,2)+1+7,1))/4);
            end
        case 10
            if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==0))
                pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,3): ...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                    x_ref+int_vec(1,1,1,3)+7,1);
            end
            if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3): ...
                    x_ref+int_vec(1,1,1,3)+7,1))/2);
            end
            if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==0))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3): ...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1: ...
                    x_ref+int_vec(1,1,1,3)+7+1,1))/2);
            end
            if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3): ...
```

```
                                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1:...
                                    x_ref+int_vec(1,1,1,3)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1:...
                                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3):...
                                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1:...
                                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3)+1:...
                                    x_ref+int_vec(1,1,1,3)+1+7,1))/4);
                                end
                            case 11
                                if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==0))
                                    pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,2):...
                                        y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                                        x_ref+int_vec(1,1,1,2)+7,1);
                                end
                                if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==1))
                                    pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                                        y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                                        x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1:...
                                        y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2):...
                                        x_ref+int_vec(1,1,1,2)+7,1))/2);
                                end
                                if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==0))
                                    pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                                        y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                                        x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2):...
                                        y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1:...
                                        x_ref+int_vec(1,1,1,2)+7+1,1))/2);
                                end
                                if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==1))
                                    pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                                        y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                                        x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2):...
                                        y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1:...
                                        x_ref+int_vec(1,1,1,2)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1:...
                                        y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2):...
                                        x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1:...
                                        y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2)+1:...
                                        x_ref+int_vec(1,1,1,2)+1+7,1))/4);
                                end
                            case 12
                                if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==0))
                                    pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,3):...
                                        y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                                        x_ref+int_vec(1,1,1,3)+7,1);
                                end
                                if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==1))
                                    pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3):...
                                        y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                                        x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1:...
                                        y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3):...
                                        x_ref+int_vec(1,1,1,3)+7,1))/2);
                                end
                                if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==0))
                                    pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3):...
                                        y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                                        x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3):...
                                        y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1:...
                                        x_ref+int_vec(1,1,1,3)+7+1,1))/2);
                                end
                                if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==1))
                                    pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3):...
                                        y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                                        x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3):...
                                        y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1:...
                                        x_ref+int_vec(1,1,1,3)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1:...
                                        y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3):...
                                        x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1:...
                                        y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3)+1:...
                                        x_ref+int_vec(1,1,1,3)+1+7,1))/4);
                                end
                        end
                    end
                end
            end
        end
    else
        pel_pred_forward=pel_ref(y_ref:y_ref+7,x_ref:x_ref+7);
    end
    p=pel_pred_forward;
    %Prediccion backward
    if(macroblock_motion_backward==1)
        if(i<=4)
            if((half_flag(1,2,1,1)==0)&(half_flag(1,2,2,1)==0))
                pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,1):...
                    y_ref+int_vec(1,2,2,1)+7,x_ref+int_vec(1,2,1,1):...
                    x_ref+int_vec(1,2,1,1)+7,2);
```

```
        end
        if((half_flag(1,2,1,1)==0)&(half_flag(1,2,2,1)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,1):...
                y_ref+int_vec(1,2,2,1)+7,x_ref+int_vec(1,2,1,1):...
                x_ref+int_vec(1,2,1,1)+7,2)+pel_ref(y_ref+int_vec(1,2,2,1)+1:...
                y_ref+int_vec(1,2,2,1)+1+7,x_ref+int_vec(1,2,1,1):...
                x_ref+int_vec(1,2,1,1)+7,2))/2);
        end
        if((half_flag(1,2,1,1)==1)&(half_flag(1,2,2,1)==0))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,1):...
                y_ref+int_vec(1,2,2,1)+7,x_ref+int_vec(1,2,1,1):...
                x_ref+int_vec(1,2,1,1)+7,2)+pel_ref(y_ref+int_vec(1,2,2,1):...
                y_ref+int_vec(1,2,2,1)+7,x_ref+int_vec(1,2,1,1)+1:...
                x_ref+int_vec(1,2,1,1)+7+1,2))/2);
        end
        if((half_flag(1,2,1,1)==1)&(half_flag(1,2,2,1)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,1):...
                y_ref+int_vec(1,2,2,1)+7,x_ref+int_vec(1,2,1,1):...
                x_ref+int_vec(1,2,1,1)+7,2)+pel_ref(y_ref+int_vec(1,2,2,1):...
                y_ref+int_vec(1,2,2,1)+7,x_ref+int_vec(1,2,1,1)+1:...
                x_ref+int_vec(1,2,1,1)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,1)+1:...
                y_ref+int_vec(1,2,2,1)+1+7,x_ref+int_vec(1,2,1,1):...
                x_ref+int_vec(1,2,1,1)+7,2)+pel_ref(y_ref+int_vec(1,2,2,1)+1:...
                y_ref+int_vec(1,2,2,1)+1+7,x_ref+int_vec(1,2,1,1)+1:...
                x_ref+int_vec(1,2,1,1)+1+7,2))/4);
        end
    else if(chroma_format==1)
        switch i
            case 5
                if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==0))
                    pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,2):...
                        y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                        x_ref+int_vec(1,2,1,2)+7,2);
                end
                if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==1))
                    pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2):...
                        y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                        x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1:...
                        y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2):...
                        x_ref+int_vec(1,2,1,2)+7,2))/2);
                end
                if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==0))
                    pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2):...
                        y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                        x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2):...
                        y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1:...
                        x_ref+int_vec(1,2,1,2)+7+1,2))/2);
                end
                if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==1))
                    pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2):...
                        y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                        x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2):...
                        y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1:...
                        x_ref+int_vec(1,2,1,2)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1:...
                        y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2):...
                        x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1:...
                        y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2)+1:...
                        x_ref+int_vec(1,2,1,2)+1+7,2))/4);
                end
            case 6
                if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==0))
                    pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,3):...
                        y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                        x_ref+int_vec(1,2,1,3)+7,2);
                end
                if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==1))
                    pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                        y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                        x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                        y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3):...
                        x_ref+int_vec(1,2,1,3)+7,2))/2);
                end
                if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==0))
                    pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                        y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                        x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3):...
                        y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1:...
                        x_ref+int_vec(1,2,1,3)+7+1,2))/2);
                end
                if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==1))
                    pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                        y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                        x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3):...
                        y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1:...
                        x_ref+int_vec(1,2,1,3)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
```

```
                        y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3): ...
                        x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1: ...
                        y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3)+1: ...
                        x_ref+int_vec(1,2,1,3)+1+7,2))/4);
                end
            end
        else if(chroma_format==2)
                switch i
                    case 5
                        if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==0))
                            pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,2): ...
                                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                                x_ref+int_vec(1,2,1,2)+7,2);
                        end
                        if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==1))
                            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2): ...
                                x_ref+int_vec(1,2,1,2)+7,2))/2);
                        end
                        if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==0))
                            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2): ...
                                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1: ...
                                x_ref+int_vec(1,2,1,2)+7+1,2))/2);
                        end
                        if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==1))
                            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2): ...
                                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1: ...
                                x_ref+int_vec(1,2,1,2)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2): ...
                                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2)+1: ...
                                x_ref+int_vec(1,2,1,2)+1+7,2))/4);
                        end
                    case 6
                        if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==0))
                            pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,3): ...
                                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                                x_ref+int_vec(1,2,1,3)+7,2);
                        end
                        if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==1))
                            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3): ...
                            y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                            x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1: ...
                            y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3): ...
                            x_ref+int_vec(1,2,1,3)+7,2))/2);
                        end
                        if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==0))
                            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3): ...
                                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3): ...
                                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1: ...
                                x_ref+int_vec(1,2,1,3)+7+1,2))/2);
                        end
                        if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==1))
                            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3): ...
                                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3): ...
                                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1: ...
                                x_ref+int_vec(1,2,1,3)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1: ...
                                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3): ...
                                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1: ...
                                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3)+1: ...
                                x_ref+int_vec(1,2,1,3)+1+7,2))/4);
                        end
                    case 7
                        if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==0))
                            pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,2): ...
                                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                                x_ref+int_vec(1,2,1,2)+7,2);
                        end
                        if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==1))
                            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2): ...
                                x_ref+int_vec(1,2,1,2)+7,2))/2);
                        end
                        if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==0))
                            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
```

```
                                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2): ...
                                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1: ...
                                x_ref+int_vec(1,2,1,2)+7+1,2))/2);
                        end
                        if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==1))
                            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2): ...
                                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1: ...
                                x_ref+int_vec(1,2,1,2)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2): ...
                                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2)+1: ...
                                x_ref+int_vec(1,2,1,2)+1+7,2))/4);
                        end
                    case 8
                        if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==0))
                            pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,3): ...
                                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                                x_ref+int_vec(1,2,1,3)+7,2);
                        end
                        if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==1))
                            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3): ...
                                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1: ...
                                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3): ...
                                x_ref+int_vec(1,2,1,3)+7,2))/2);
                        end
                        if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==0))
                            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3): ...
                                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3): ...
                                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1: ...
                                x_ref+int_vec(1,2,1,3)+7+1,2))/2);
                        end
                        if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==1))
                            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3): ...
                                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3): ...
                                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1: ...
                                x_ref+int_vec(1,2,1,3)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1: ...
                                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3): ...
                                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1: ...
                                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3)+1: ...
                                x_ref+int_vec(1,2,1,3)+1+7,2))/4);
                        end
                end
            else if(chroma_format==3)
                    switch i
                        case 5
                            if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==0))
                                pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,2): ...
                                    y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                                    x_ref+int_vec(1,2,1,2)+7,2);
                            end
                            if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==1))
                                pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                                    y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                                    x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                                    y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2): ...
                                    x_ref+int_vec(1,2,1,2)+7,2))/2);
                            end
                            if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==0))
                                pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                                    y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                                    x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2): ...
                                    y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1: ...
                                    x_ref+int_vec(1,2,1,2)+7+1,2))/2);
                            end
                            if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==1))
                                pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                                    y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                                    x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2): ...
                                    y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1: ...
                                    x_ref+int_vec(1,2,1,2)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                                    y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2): ...
                                    x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                                    y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2)+1: ...
                                    x_ref+int_vec(1,2,1,2)+1+7,2))/4);
                            end
                        case 6
                            if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==0))
                                pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,3): ...
                                    y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
```

```
                x_ref+int_vec(1,2,1,3)+7,2);
        end
        if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2))/2);
        end
        if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==0))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1:...
                x_ref+int_vec(1,2,1,3)+7+1,2))/2);
        end
        if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1:...
                x_ref+int_vec(1,2,1,3)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3)+1:...
                x_ref+int_vec(1,2,1,3)+1+7,2))/4);
        end
    case 7
        if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==0))
            pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,2):...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                x_ref+int_vec(1,2,1,2)+7,2);
        end
        if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2):...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1:...
                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2):...
                x_ref+int_vec(1,2,1,2)+7,2))/2);
        end
        if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==0))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2):...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2):...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1:...
                x_ref+int_vec(1,2,1,2)+7+1,2))/2);
        end
        if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2):...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2):...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1:...
                x_ref+int_vec(1,2,1,2)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1:...
                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2):...
                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1:...
                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2)+1:...
                x_ref+int_vec(1,2,1,2)+1+7,2))/4);
        end
    case 8
        if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==0))
            pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2);
        end
        if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2))/2);
        end
        if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==0))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1:...
                x_ref+int_vec(1,2,1,3)+7+1,2))/2);
        end
        if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1:...
                x_ref+int_vec(1,2,1,3)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3):...
```

```
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1: ...
                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3)+1: ...
                x_ref+int_vec(1,2,1,3)+1+7,2))/4);
        end
    case 9
        if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==0))
            pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,2): ...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                x_ref+int_vec(1,2,1,2)+7,2);
        end
        if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2): ...
                x_ref+int_vec(1,2,1,2)+7,2))/2);
        end
        if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==0))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2): ...
            y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1: ...
            x_ref+int_vec(1,2,1,2)+7+1,2))/2);
        end
        if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2): ...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1: ...
                x_ref+int_vec(1,2,1,2)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2): ...
                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2)+1: ...
                x_ref+int_vec(1,2,1,2)+1+7,2))/4);
        end
    case 10
        if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==0))
            pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,3): ...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                x_ref+int_vec(1,2,1,3)+7,2);
        end
        if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3): ...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1: ...
                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3): ...
                x_ref+int_vec(1,2,1,3)+7,2))/2);
        end
        if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==0))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3): ...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3): ...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1: ...
                x_ref+int_vec(1,2,1,3)+7+1,2))/2);
        end
        if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3): ...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3): ...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1: ...
                x_ref+int_vec(1,2,1,3)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1: ...
                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3): ...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1: ...
                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3)+1: ...
                x_ref+int_vec(1,2,1,3)+1+7,2))/4);
        end
    case 11
        if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==0))
            pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,2): ...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                x_ref+int_vec(1,2,1,2)+7,2);
        end
        if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2): ...
                x_ref+int_vec(1,2,1,2)+7,2))/2);
        end
        if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==0))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2): ...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1: ...
                x_ref+int_vec(1,2,1,2)+7+1,2))/2);
```

```
                                    end
                                    if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==1))
                                        pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2):...
                                            y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                                            x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2):...
                                            y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1:...
                                            x_ref+int_vec(1,2,1,2)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1:...
                                            y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2):...
                                            x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1:...
                                            y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2)+1:...
                                            x_ref+int_vec(1,2,1,2)+1+7,2))/4);
                                    end
                                case 12
                                    if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==0))
                                        pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,3):...
                                            y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                                            x_ref+int_vec(1,2,1,3)+7,2);
                                    end
                                    if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==1))
                                        pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                                            y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                                            x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                                            y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3):...
                                            x_ref+int_vec(1,2,1,3)+7,2))/2);
                                    end
                                    if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==0))
                                        pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                                            y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                                            x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3):...
                                            y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1:...
                                            x_ref+int_vec(1,2,1,3)+7+1,2))/2);
                                    end
                                    if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==1))
                                        pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                                            y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                                            x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3):...
                                            y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1:...
                                            x_ref+int_vec(1,2,1,3)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                                            y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3):...
                                            x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                                            y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3)+1:...
                                            x_ref+int_vec(1,2,1,3)+1+7,2))/4);
                                    end
                                end
                            end
                        end
                    end
                end
            end
            p=round((pel_pred_forward+pel_pred_backward)/2);
        end
    end
end

%Valor del bloque
d(y,x)=f(y,x)+p(y,x);

%Saturacion
for y=1:8
    for x=1:8
        if(d(y,x)<0)
            d(y,x)=0;
        end
        if(d(y,x)>255)
            d(y,x)=255;
        end
        x=x+1;
    end
    y=y+1;
end

%Formacion de macros
if(i<=4)
    switch i
        case 1
            macros_luma(1:8,1:8,num_mb)=d;
        case 2
            macros_luma(1:8,9:16,num_mb)=d;
        case 3
            macros_luma(9:16,1:8,num_mb)=d;
        case 4
            macros_luma(9:16,9:16,num_mb)=d;
    end
else if(chroma_format==1) %4:2:0
        switch i
            case 5
```

```
            macro_chroma_b_8x8=d;
            for m=1:8
                for n=1:8
                    fila=(m-1)*2+1;
                    columna=(n-1)*2+1;
                    macros_chroma_b(fila:fila+1,columna:columna+1,num_mb)=macro_chroma_b_8x8(m,n);
                    n=n+1;
                end
                m=m+1;
            end

        case 6
            macro_chroma_r_8x8=d;
            for m=1:8
                for n=1:8
                    fila=(m-1)*2+1;
                    columna=(n-1)*2+1;
                    macros_chroma_r(fila:fila+1,columna:columna+1,num_mb)=macro_chroma_r_8x8(m,n);
                    n=n+1;
                end
                m=m+1;
            end

        end
else if(chroma_format==2) %4:2:2
        switch i
            case 5
                macro_chroma_b_8x8(1:8,1:8)=d;
                for m=1:8
                    for n=1:8
                        fila=m;
                        columna=(n-1)*2+1;
                        macros_chroma_b(fila,columna:columna+1,num_mb)=macro_chroma_b_8x8(m,n);
                        n=n+1;
                    end
                m=m+1;
                end
            case 6
                macro_chroma_r_8x8(1:8,1:8)=d;
                for m=1:8
                    for n=1:8
                        fila=m;
                        columna=(n-1)*2+1;
                        macros_chroma_r(fila,columna:columna+1,num_mb)=macro_chroma_r_8x8(m,n);
                        n=n+1;
                    end
                m=m+1;
                end
            case 7
                macro_chroma_b_8x8(9:15,1:8)=d;
                for m=1:8
                    for n=1:8
                        fila=m+8;
                        columna=(n-1)*2+1;
                        macros_chroma_b(fila,columna:columna+1,num_mb)=macro_chroma_b_8x8(m,n);
                        n=n+1;
                    end
                m=m+1;
                end
            case 8
                macro_chroma_r_8x8(9:16,1:8)=d;
                for m=1:8
                    for n=1:8
                        fila=m+8;
                        columna=(n-1)*2+1;
                        macros_chroma_r(fila,columna:columna+1,num_mb)=macro_chroma_r_8x8(m,n);
                        n=n+1;
                    end
                m=m+1;
                end
        end
    else if(chroma_format==3) %4:4:4
            switch i
                case 5
                    macros_chroma_b(1:8,1:8,num_mb)=d;
                case 6
                    macros_chroma_r(1:8,1:8,num_mb)=d;
                case 7
                    macros_chroma_b(9:16,1:8,num_mb)=d;
                case 8
                    macros_chroma_r(9:16,1:8,num_mb)=d;
                case 9
                    macros_chroma_b(1:8,9:16,num_mb)=d;
                case 10
                    macros_chroma_r(1:8,9:16,num_mb)=d;
```

```
                    case 11
                        macros_chroma_b(9:16,9:16,num_mb)=d;
                    case 12
                        macros_chroma_r(9:16,9:16,num_mb)=d;
                end
            end
        end
    end

end

if(macroblock_address_increment==1)
    macro_luma(:,:)=macros_luma(:,:,num_mb);
    macro_chroma_b(:,:)=macros_chroma_b(:,:,num_mb);
    macro_chroma_r(:,:)=macros_chroma_r(:,:,num_mb);
else
    for j=1:num_mb
        columna=(j-1)*16+1;
        macro_luma(1:16,columna:columna+15)=macros_luma(:,:,j);
        macro_chroma_b(1:16,columna:columna+15)=macros_chroma_b(:,:,j);
        macro_chroma_r(1:16,columna:columna+15)=macros_chroma_r(:,:,j);
    end
end

%una vez decodificados todos los motion vectors presentes en el macroblock
%update el resto de predictores
if(picture_structure==3) %Imagen trama
    if(frame_motion_type==2)
        if(macroblock_intra==1)
            PMV(2,1,1:2)=PMV(1,1,1:2);
            if(concealment_motion_vectors==0)
                PMV(:,:,:)=0;
            end
        else if((macroblock_motion_forward==1)&(macroblock_motion_backward==1))
                PMV(2,1,1:2)=PMV(1,1,1:2);
                PMV(2,2,1:2)=PMV(1,2,1:2);

            else if((macroblock_motion_forward==1)&(macroblock_motion_backward==0))
                    PMV(2,1,1:2)=PMV(1,1,1:2);
                else if((macroblock_motion_forward==0)&(macroblock_motion_backward==1))
                        PMV(2,2,1:2)=PMV(1,2,1:2);
                    else
                        PMV(:,:,:)=0;
                    end
                end
            end
        end
    else if(frame_motion_type==3)
            if((macroblock_motion_forward==1)&(macroblock_motion_backward==0)&(macroblock_intra==0))
                PMV(2,1,1:2)=PMV(1,1,1:2);
            end
        end
    end
else if(field_motion_type==1)
        if(macroblock_intra==1)
            PMV(2,1,1:2)=PMV(1,1,1:2);
            if(concealment_motion_vectors==0)
                PMV(:,:,:)=0;
            end
        else if((macroblock_motion_forward==1)&(macroblock_motion_backward==1))
                PMV(2,1,1:2)=PMV(1,1,1:2);
                PMV(2,2,1:2)=PMV(1,2,1:2);
            else if((macroblock_motion_forward==1)&(macroblock_motion_backward==0))
                    PMV(2,1,1:2)=PMV(1,1,1:2);
                else if((macroblock_motion_forward==0)&(macroblock_motion_backward==1))
                        PMV(2,2,1:2)=PMV(1,2,1:2);
                    else
                        PMV(:,:,:)=0;
                    end
                end
            end
        end
    else if(field_motion_type==3)
            if((macroblock_motion_forward==1)&(macroblock_motion_backward==0)&(macroblock_intra==0))
                PMV(2,1,1:2)=PMV(1,1,1:2);
            end
        end
    end
end

bits_leidos=k-1;
```

```
%*************************************************************************************************************
%                                              Macroblock skipped                                             %
%                                                                                                             %
%                                                                                                             %
%*************************************************************************************************************

function [macro_luma,macro_chroma_b,macro_chroma_r]=macroblock_skipped(block_count,mb_row,mb_column, ...
    im_referencia,chroma_format,half_flag,int_vec,macroblock_motion_forward,macroblock_motion_backward)

y=1:8;
x=1:8;

f(y,x)=0;
for i=1:block_count
    y=1:8;
    x=1:8;

    if(i<=4)
        switch i
           case 1
               y_ref=double((mb_row-1)*16+1);
               x_ref=double((mb_column-1)*16+1);
           case 2
               y_ref=double((mb_row-1)*16+1);
               x_ref=double((mb_column-1)*16+8+1);
           case 3
               y_ref=double((mb_row-1)*16+8+1);
               x_ref=(double(mb_column-1)*16+1);
           case 4
               y_ref=double((mb_row-1)*16+8+1);
               x_ref=double((mb_column-1)*16+8+1);
        end
        pel_ref=im_referencia(:,:,1,:);
        pel_ref(241:256,321:336,:)=int16(0);
    else if(chroma_format==1)
            switch i
                case 5
                    y_ref=double((mb_row-1)*8+1);
                    x_ref=double((mb_column-1)*8+1);
                    for m=1:240
                        for n=1:320
                            fila=floor(m/2)+1;
                            columna=floor(n/2)+1;
                            pel_ref(fila,columna,:)=im_referencia(m,n,2,:);
                        end
                    end
                    pel_ref(121:128,161:168,:)=int16(0);
                case 6
                    y_ref=double((mb_row-1)*8+1);
                    x_ref=double((mb_column-1)*8+1);
                    for m=1:240
                        for n=1:320
                            fila=floor(m/2)+1;
                            columna=floor(n/2)+1;
                            pel_ref(fila,columna,:)=im_referencia(m,n,3,:);
                        end
                    end
                    pel_ref(121:128,161:168,:)=int16(0);
            end
        else if(chroma_format==2)
                switch i
                    case 5
                        y_ref=double((mb_row-1)*8+1);
                        x_ref=double((mb_column-1)*8+1);
                        for m=1:240
                            for n=1:320
                                fila=m;
                                columna=floor(n/2)+1;
                                pel_ref(fila,columna,:)=im_referencia(m,n,2,:);
                            end
                        end
                        pel_ref(121:128,161:168,:)=int16(0);
                    case 6
                        y_ref=double((mb_row-1)*8+1);
                        x_ref=double((mb_column-1)*8+1);
                        for m=1:240
                            for n=1:320
                                fila=m;
                                columna=floor(n/2)+1;
                                pel_ref(fila,columna,:)=im_referencia(m,n,3,:);
                            end
                        end
                        pel_ref(121:128,161:168,:)=int16(0);
                    case 7
                        y_ref=double((mb_row-1)*8+8+1);
```

```
                x_ref=double((mb_column-1)*8+1);
                for m=1:240
                    for n=1:320
                        fila=m;
                        columna=floor(n/2)+1;
                        pel_ref(fila,columna,:)=im_referencia(m,n,2,:)
                    end
                end
                pel_ref(121:128,161:168,:)=int16(0);
            case 8
                y_ref=double((mb_row-1)*8+8+1);
                x_ref=double((mb_column-1)*8+1);
                for m=1:240
                    for n=1:320
                        fila=m;
                        columna=floor(n/2)+1;
                        pel_ref(fila,columna,:)=im_referencia(m,n,3,:);
                    end
                end
                pel_ref(121:128,161:168,:)=int16(0);
        end
    else if(chroma_format==3)
            switch i
                case 5
                    y_ref=double((mb_row-1)*8+1);
                    x_ref=double((mb_column-1)*8+1);
                    pel_ref(:,:,:)=im_referencia(:,:,2,:);
                    pel_ref(121:128,161:168,:)=int16(0);
                case 6
                    y_ref=double((mb_row-1)*8+1);
                    x_ref=double((mb_column-1)*8+1);
                    pel_ref(:,:,:)=im_referencia(:,:,3,:);
                    pel_ref(121:128,161:168,:)=int16(0);
                case 7
                    y_ref=double((mb_row-1)*8+8+1);
                    x_ref=double((mb_column-1)*8+1);
                    pel_ref(:,:,:)=im_referencia(:,:,2,:);
                    pel_ref(121:128,161:168,:)=int16(0);
                case 8
                    y_ref=double((mb_row-1)*8+8+1);
                    x_ref=double((mb_column-1)*8+1);
                    pel_ref(:,:,:)=im_referencia(:,:,3,:);
                    pel_ref(121:128,161:168,:)=int16(0);
                case 9
                    y_ref=double((mb_row-1)*8+1);
                    x_ref=double((mb_column-1)*8+8+1);
                    pel_ref(:,:,:)=im_referencia(:,:,2,:);
                    pel_ref(121:128,161:168,:)=int16(0);
                case 10
                    y_ref=double((mb_row-1)*8+1);
                    x_ref=double((mb_column-1)*8+8+1);
                    pel_ref(:,:,:)=im_referencia(:,:,3,:);
                    pel_ref(121:128,161:168,:)=int16(0);
                case 11
                    y_ref=double((mb_row-1)*8+8+1);
                    x_ref=double((mb_column-1)*8+8+1);
                    pel_ref(:,:,:)=im_referencia(:,:,2,:);
                    pel_ref(121:128,161:168,:)=int16(0);
                case 12
                    y_ref=double((mb_row-1)*8+8+1);
                    x_ref=double((mb_column-1)*8+8+1);
                    pel_ref(:,:,:)=im_referencia(:,:,3,:);
                    pel_ref(121:128,161:168,:)=int16(0);
            end
        end
    end
end
end
if(macroblock_motion_forward==1)
    if(i<=4)
        if((half_flag(1,1,1,1)==0)&(half_flag(1,1,2,1)==0))
            pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,1):...
                y_ref+int_vec(1,1,2,1)+7,x_ref+int_vec(1,1,1,1):...
                x_ref+int_vec(1,1,1,1)+7,1);
        end
        if((half_flag(1,1,1,1)==0)&(half_flag(1,1,2,1)==1))
            pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,1):...
                y_ref+int_vec(1,1,2,1)+7,x_ref+int_vec(1,1,1,1):...
                x_ref+int_vec(1,1,1,1)+7,1)+pel_ref(y_ref+int_vec(1,1,2,1)+1:...
                y_ref+int_vec(1,1,2,1)+1+7,x_ref+int_vec(1,1,1,1):...
                x_ref+int_vec(1,1,1,1)+7,1))/2);
        end
        if((half_flag(1,1,1,1)==1)&(half_flag(1,1,2,1)==0))
            pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,1):...
                y_ref+int_vec(1,1,2,1)+7,x_ref+int_vec(1,1,1,1):...
```

```
            x_ref+int_vec(1,1,1,1)+7,1)+pel_ref(y_ref+int_vec(1,1,2,1): ...
            y_ref+int_vec(1,1,2,1)+7,x_ref+int_vec(1,1,1,1)+1: ...
            x_ref+int_vec(1,1,1,1)+7+1,1))/2);
    end
    if((half_flag(1,1,1,1)==1)&(half_flag(1,1,2,1)==1))
        pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,1): ...
            y_ref+int_vec(1,1,2,1)+7,x_ref+int_vec(1,1,1,1): ...
            x_ref+int_vec(1,1,1,1)+7,1)+pel_ref(y_ref+int_vec(1,1,2,1): ...
            y_ref+int_vec(1,1,2,1)+7,x_ref+int_vec(1,1,1,1)+1: ...
            x_ref+int_vec(1,1,1,1)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,1)+1: ...
            y_ref+int_vec(1,1,2,1)+1+7,x_ref+int_vec(1,1,1,1): ...
            x_ref+int_vec(1,1,1,1)+7,1)+pel_ref(y_ref+int_vec(1,1,2,1)+1: ...
            y_ref+int_vec(1,1,2,1)+1+7,x_ref+int_vec(1,1,1,1)+1: ...
            x_ref+int_vec(1,1,1,1)+1+7,1))/4);
    end
else  if(chroma_format==1)
        switch i
            case 5
                if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==0))
                    pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,2): ...
                        y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                        x_ref+int_vec(1,1,1,2)+7,1);
                end
                if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==1))
                    pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                        y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                        x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                        y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2): ...
                        x_ref+int_vec(1,1,1,2)+7,1))/2);
                end
                if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==0))
                    pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                        y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                        x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2): ...
                        y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1: ...
                        x_ref+int_vec(1,1,1,2)+7+1,1))/2);
                end
                if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==1))
                    pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                        y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                        x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2): ...
                        y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1: ...
                        x_ref+int_vec(1,1,1,2)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                        y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2): ...
                        x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                        y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2)+1: ...
                        x_ref+int_vec(1,1,1,2)+1+7,1))/4);
                end
            case 6
                if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==0))
                    pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,3): ...
                        y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                        x_ref+int_vec(1,1,1,3)+7,1);
                end
                if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==1))
                    pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                        y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                        x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                        y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3): ...
                        x_ref+int_vec(1,1,1,3)+7,1))/2);
                end
                if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==0))
                    pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                        y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                        x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3): ...
                        y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1: ...
                        x_ref+int_vec(1,1,1,3)+7+1,1))/2);
                end
                if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==1))
                    pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                        y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                        x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3): ...
                        y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1: ...
                        x_ref+int_vec(1,1,1,3)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                        y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3): ...
                        x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                        y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3)+1: ...
                        x_ref+int_vec(1,1,1,3)+1+7,1))/4);
                end
        end
    else  if(chroma_format==2)
            switch i
                case 5
                    if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==0))
                        pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,2): ...
```

```
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1);
            end
            if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1:...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1))/2);
            end
            if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==0))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1:...
                    x_ref+int_vec(1,1,1,2)+7+1,1))/2);
            end
            if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1:...
                    x_ref+int_vec(1,1,1,2)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1:...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1:...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2)+1:...
                    x_ref+int_vec(1,1,1,2)+1+7,1))/4);
            end
        case 6
            if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==0))
                pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1);
            end
            if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1:...
                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1))/2);
            end
            if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==0))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1:...
                    x_ref+int_vec(1,1,1,3)+7+1,1))/2);
            end
            if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1:...
                    x_ref+int_vec(1,1,1,3)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1:...
                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1:...
                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3)+1:...
                    x_ref+int_vec(1,1,1,3)+1+7,1))/4);
            end
        case 7
            if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==0))
                pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1);
            end
            if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1:...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1))/2);
            end
            if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==0))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1:...
                    x_ref+int_vec(1,1,1,2)+7+1,1))/2);
            end
            if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1:...
                    x_ref+int_vec(1,1,1,2)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1:...
```

```
                        y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2): ...
                        x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                        y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2)+1: ...
                        x_ref+int_vec(1,1,1,2)+1+7,1))/4);
                end
            case 8
                if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==0))
                    pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,3): ...
                        y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                        x_ref+int_vec(1,1,1,3)+7,1);
                end
                if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==1))
                    pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                        y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                        x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                        y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3): ...
                        x_ref+int_vec(1,1,1,3)+7,1))/2);
                end
                if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==0))
                    pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                        y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                        x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3): ...
                        y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1: ...
                        x_ref+int_vec(1,1,1,3)+7+1,1))/2);
                end
                if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==1))
                    pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                        y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                        x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3): ...
                        y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1: ...
                        x_ref+int_vec(1,1,1,3)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                        y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3): ...
                        x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                        y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3)+1: ...
                        x_ref+int_vec(1,1,1,3)+1+7,1))/4);
                end
        end

    else if(chroma_format==3)
            switch i
                case 5
                    if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==0))
                        pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,2): ...
                            y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                            x_ref+int_vec(1,1,1,2)+7,1);
                    end
                    if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==1))
                        pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                            y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                            x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                            y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2): ...
                            x_ref+int_vec(1,1,1,2)+7,1))/2);
                    end
                    if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==0))
                        pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                            y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                            x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2): ...
                            y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1: ...
                            x_ref+int_vec(1,1,1,2)+7+1,1))/2);
                    end
                    if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==1))
                        pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2): ...
                            y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2): ...
                            x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2): ...
                            y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1: ...
                            x_ref+int_vec(1,1,1,2)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                            y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2): ...
                            x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1: ...
                            y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2)+1: ...
                            x_ref+int_vec(1,1,1,2)+1+7,1))/4);
                    end
                case 6
                    if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==0))
                        pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,3): ...
                            y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                            x_ref+int_vec(1,1,1,3)+7,1);
                    end
                    if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==1))
                        pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3): ...
                            y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3): ...
                            x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1: ...
                            y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3): ...
                            x_ref+int_vec(1,1,1,3)+7,1))/2);
                    end
                    if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==0))
```

```
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1:...
                    x_ref+int_vec(1,1,1,3)+7+1,1))/2);
            end
            if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1:...
                    x_ref+int_vec(1,1,1,3)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1:...
                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1:...
                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3)+1:...
                    x_ref+int_vec(1,1,1,3)+1+7,1))/4);
            end
        case 7
            if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==0))
                pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1);
            end
            if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1:...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1))/2);
            end
            if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==0))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1:...
                    x_ref+int_vec(1,1,1,2)+7+1,1))/2);
            end
            if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1:...
                    x_ref+int_vec(1,1,1,2)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1:...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1:...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2)+1:...
                    x_ref+int_vec(1,1,1,2)+1+7,1))/4);
            end
        case 8
            if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==0))
                pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1);
            end
            if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1:...
                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1))/2);
            end
            if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==0))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1:...
                    x_ref+int_vec(1,1,1,3)+7+1,1))/2);
            end
            if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1:...
                    x_ref+int_vec(1,1,1,3)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1:...
                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1:...
                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3)+1:...
                    x_ref+int_vec(1,1,1,3)+1+7,1))/4);
            end
        case 9
            if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==0))
                pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1);
            end
```

```
            if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1:...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1))/2);
            end
            if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==0))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1:...
                    x_ref+int_vec(1,1,1,2)+7+1,1))/2);
            end
            if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1:...
                    x_ref+int_vec(1,1,1,2)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1:...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1:...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2)+1:...
                    x_ref+int_vec(1,1,1,2)+1+7,1))/4);
            end
        case 10
            if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==0))
                pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1);
            end
            if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1:...
                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1))/2);
            end
            if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==0))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1:...
                    x_ref+int_vec(1,1,1,3)+7+1,1))/2);
            end
            if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3):...
                    y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1:...
                    x_ref+int_vec(1,1,1,3)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1:...
                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3):...
                    x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1:...
                    y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3)+1:...
                    x_ref+int_vec(1,1,1,3)+1+7,1))/4);
            end
        case 11
            if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==0))
                pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1);
            end
            if((half_flag(1,1,1,2)==0)&(half_flag(1,1,2,2)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1:...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1))/2);
            end
            if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==0))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1:...
                    x_ref+int_vec(1,1,1,2)+7+1,1))/2);
            end
            if((half_flag(1,1,1,2)==1)&(half_flag(1,1,2,2)==1))
                pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2):...
                    y_ref+int_vec(1,1,2,2)+7,x_ref+int_vec(1,1,1,2)+1:...
                    x_ref+int_vec(1,1,1,2)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1:...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2):...
                    x_ref+int_vec(1,1,1,2)+7,1)+pel_ref(y_ref+int_vec(1,1,2,2)+1:...
                    y_ref+int_vec(1,1,2,2)+1+7,x_ref+int_vec(1,1,1,2)+1:...
```

```
                                x_ref+int_vec(1,1,1,2)+1+7,1))/4);
                        end
                    case 12
                        if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==0))
                            pel_pred_forward(y,x)=pel_ref(y_ref+int_vec(1,1,2,3):...
                                y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                                x_ref+int_vec(1,1,1,3)+7,1);
                        end
                        if((half_flag(1,1,1,3)==0)&(half_flag(1,1,2,3)==1))
                            pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3):...
                                y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                                x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1:...
                                y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3):...
                                x_ref+int_vec(1,1,1,3)+7,1))/2);
                        end
                        if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==0))
                            pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3):...
                                y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                                x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3):...
                                y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1:...
                                x_ref+int_vec(1,1,1,3)+7+1,1))/2);
                        end
                        if((half_flag(1,1,1,3)==1)&(half_flag(1,1,2,3)==1))
                            pel_pred_forward(y,x)=round((pel_ref(y_ref+int_vec(1,1,2,3):...
                                y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3):...
                                x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3):...
                                y_ref+int_vec(1,1,2,3)+7,x_ref+int_vec(1,1,1,3)+1:...
                                x_ref+int_vec(1,1,1,3)+7+1,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1:...
                                y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3):...
                                x_ref+int_vec(1,1,1,3)+7,1)+pel_ref(y_ref+int_vec(1,1,2,3)+1:...
                                y_ref+int_vec(1,1,2,3)+1+7,x_ref+int_vec(1,1,1,3)+1:...
                                x_ref+int_vec(1,1,1,3)+1+7,1))/4);
                        end
                end
            end
        end
    end
else
    pel_pred_forward=pel_ref(y_ref:y_ref+7,x_ref:x_ref+7);
end
p=pel_pred_forward;
if(macroblock_motion_backward==1)
    if(i<=4)
        if((half_flag(1,2,1,1)==0)&(half_flag(1,2,2,1)==0))
            pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,1):...
                y_ref+int_vec(1,2,2,1)+7,x_ref+int_vec(1,2,1,1):...
                x_ref+int_vec(1,2,1,1)+7,2);
        end
        if((half_flag(1,2,1,1)==0)&(half_flag(1,2,2,1)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,1):...
                y_ref+int_vec(1,2,2,1)+7,x_ref+int_vec(1,2,1,1):...
                x_ref+int_vec(1,2,1,1)+7,2)+pel_ref(y_ref+int_vec(1,2,2,1)+1:...
                y_ref+int_vec(1,2,2,1)+1+7,x_ref+int_vec(1,2,1,1):...
                x_ref+int_vec(1,2,1,1)+7,2))/2);
        end
        if((half_flag(1,2,1,1)==1)&(half_flag(1,2,2,1)==0))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,1):...
                y_ref+int_vec(1,2,2,1)+7,x_ref+int_vec(1,2,1,1):...
                x_ref+int_vec(1,2,1,1)+7,2)+pel_ref(y_ref+int_vec(1,2,2,1):...
                y_ref+int_vec(1,2,2,1)+7,x_ref+int_vec(1,2,1,1)+1:...
                x_ref+int_vec(1,2,1,1)+7+1,2))/2);
        end
        if((half_flag(1,2,1,1)==1)&(half_flag(1,2,2,1)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,1):...
                y_ref+int_vec(1,2,2,1)+7,x_ref+int_vec(1,2,1,1):...
                x_ref+int_vec(1,2,1,1)+7,2)+pel_ref(y_ref+int_vec(1,2,2,1):...
                y_ref+int_vec(1,2,2,1)+7,x_ref+int_vec(1,2,1,1)+1:...
                x_ref+int_vec(1,2,1,1)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,1)+1:...
                y_ref+int_vec(1,2,2,1)+1+7,x_ref+int_vec(1,2,1,1):...
                x_ref+int_vec(1,2,1,1)+7,2)+pel_ref(y_ref+int_vec(1,2,2,1)+1:...
                y_ref+int_vec(1,2,2,1)+1+7,x_ref+int_vec(1,2,1,1)+1:...
                x_ref+int_vec(1,2,1,1)+1+7,2))/4);
        end
    else if(chroma_format==1)
            switch i
                case 5
                    if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==0))
                        pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,2):...
                            y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                            x_ref+int_vec(1,2,1,2)+7,2);
                    end
                    if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==1))
                        pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2):...
                            y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
```

```
                    x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                    y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2): ...
                    x_ref+int_vec(1,2,1,2)+7,2))/2);
                end
                if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==0))
                    pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                        y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                        x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2): ...
                        y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1: ...
                        x_ref+int_vec(1,2,1,2)+7+1,2))/2);
                end
                if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==1))
                    pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                        y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                        x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2): ...
                        y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1: ...
                        x_ref+int_vec(1,2,1,2)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                        y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2): ...
                        x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                        y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2)+1: ...
                        x_ref+int_vec(1,2,1,2)+1+7,2))/4);
                end
            case 6
                if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==0))
                    pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,3): ...
                        y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                        x_ref+int_vec(1,2,1,3)+7,2);
                end
                if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==1))
                    pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3): ...
                        y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                        x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1: ...
                        y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3): ...
                        x_ref+int_vec(1,2,1,3)+7,2))/2);
                end
                if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==0))
                    pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3): ...
                        y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                        x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3): ...
                        y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1: ...
                        x_ref+int_vec(1,2,1,3)+7+1,2))/2);
                end
                if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==1))
                    pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3): ...
                        y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                        x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3): ...
                        y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1: ...
                        x_ref+int_vec(1,2,1,3)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1: ...
                        y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3): ...
                        x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1: ...
                        y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3)+1: ...
                        x_ref+int_vec(1,2,1,3)+1+7,2))/4);
                end
        end
    else if(chroma_format==2)
            switch i
                case 5
                    if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==0))
                        pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,2): ...
                            y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                            x_ref+int_vec(1,2,1,2)+7,2);
                    end
                    if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==1))
                        pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                            y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                            x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                            y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2): ...
                            x_ref+int_vec(1,2,1,2)+7,2))/2);
                    end
                    if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==0))
                        pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                            y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                            x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2): ...
                            y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1: ...
                            x_ref+int_vec(1,2,1,2)+7+1,2))/2);
                    end
                    if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==1))
                        pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                            y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                            x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2): ...
                            y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1: ...
                            x_ref+int_vec(1,2,1,2)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                            y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2): ...
                            x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                            y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2)+1: ...
```

```
                x_ref+int_vec(1,2,1,2)+1+7,2))/4);
        end
    case 6
        if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==0))
            pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2);
        end
        if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2))/2);
        end
        if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==0))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1:...
                x_ref+int_vec(1,2,1,3)+7+1,2))/2);
        end
        if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1:...
                x_ref+int_vec(1,2,1,3)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3)+1:...
                x_ref+int_vec(1,2,1,3)+1+7,2))/4);
        end
    case 7
        if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==0))
            pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,2):...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                x_ref+int_vec(1,2,1,2)+7,2);
        end
        if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2):...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1:...
                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2):...
                x_ref+int_vec(1,2,1,2)+7,2))/2);
        end
        if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==0))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2):...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2):...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1:...
                x_ref+int_vec(1,2,1,2)+7+1,2))/2);
        end
        if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2):...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2):...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1:...
                x_ref+int_vec(1,2,1,2)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1:...
                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2):...
                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1:...
                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2)+1:...
                x_ref+int_vec(1,2,1,2)+1+7,2))/4);
        end
    case 8
        if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==0))
            pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2);
        end
        if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2))/2);
        end
        if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==0))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1:...
                x_ref+int_vec(1,2,1,3)+7+1,2))/2);
        end
        if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==1))
```

```
                    pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3): ...
                        y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                        x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3): ...
                        y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1: ...
                        x_ref+int_vec(1,2,1,3)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1: ...
                        y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3): ...
                        x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1: ...
                        y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3)+1: ...
                        x_ref+int_vec(1,2,1,3)+1+7,2))/4);
                end
            end
        else if(chroma_format==3)
                switch i
                    case 5
                        if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==0))
                            pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,2): ...
                                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                                x_ref+int_vec(1,2,1,2)+7,2);
                        end
                        if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==1))
                            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2): ...
                                x_ref+int_vec(1,2,1,2)+7,2))/2);
                        end
                        if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==0))
                            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2): ...
                                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1: ...
                                x_ref+int_vec(1,2,1,2)+7+1,2))/2);
                        end
                        if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==1))
                            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2): ...
                                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1: ...
                                x_ref+int_vec(1,2,1,2)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2): ...
                                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
                                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2)+1: ...
                                x_ref+int_vec(1,2,1,2)+1+7,2))/4);
                        end
                    case 6
                        if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==0))
                            pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,3): ...
                                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                                x_ref+int_vec(1,2,1,3)+7,2);
                        end
                        if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==1))
                            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3): ...
                                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1: ...
                                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3): ...
                                x_ref+int_vec(1,2,1,3)+7,2))/2);
                        end
                        if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==0))
                            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3): ...
                                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3): ...
                                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1: ...
                                x_ref+int_vec(1,2,1,3)+7+1,2))/2);
                        end
                        if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==1))
                            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3): ...
                                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3): ...
                                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3): ...
                                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1: ...
                                x_ref+int_vec(1,2,1,3)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1: ...
                                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3): ...
                                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1: ...
                                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3)+1: ...
                                x_ref+int_vec(1,2,1,3)+1+7,2))/4);
                        end
                    case 7
                        if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==0))
                            pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,2): ...
                                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                                x_ref+int_vec(1,2,1,2)+7,2);
                        end
                        if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==1))
                            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2): ...
                                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2): ...
                                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1: ...
```

```
                    y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2):...
                    x_ref+int_vec(1,2,1,2)+7,2))/2);
                end
                if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==0))
                    pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2):...
                        y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                        x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2):...
                        y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1:...
                        x_ref+int_vec(1,2,1,2)+7+1,2))/2);
                end
                if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==1))
                    pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2):...
                        y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                        x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2):...
                        y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1:...
                        x_ref+int_vec(1,2,1,2)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1:...
                        y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2):...
                        x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1:...
                        y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2)+1:...
                        x_ref+int_vec(1,2,1,2)+1+7,2))/4);
                end
            case 8
                if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==0))
                    pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,3):...
                        y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                        x_ref+int_vec(1,2,1,3)+7,2);
                end
                if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==1))
                    pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                        y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                        x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                        y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3):...
                        x_ref+int_vec(1,2,1,3)+7,2))/2);
                end
                if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==0))
                    pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                        y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                        x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3):...
                        y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1:...
                        x_ref+int_vec(1,2,1,3)+7+1,2))/2);
                end
                if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==1))
                    pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                        y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                        x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3):...
                        y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1:...
                        x_ref+int_vec(1,2,1,3)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                        y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3):...
                        x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                        y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3)+1:...
                        x_ref+int_vec(1,2,1,3)+1+7,2))/4);
                end
            case 9
                if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==0))
                    pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,2):...
                        y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                        x_ref+int_vec(1,2,1,2)+7,2);
                end
                if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==1))
                    pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2):...
                        y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                        x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1:...
                        y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2):...
                        x_ref+int_vec(1,2,1,2)+7,2))/2);
                end
                if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==0))
                    pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2):...
                        y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                        x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2):...
                        y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1:...
                        x_ref+int_vec(1,2,1,2)+7+1,2))/2);
                end
                if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==1))
                    pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2):...
                        y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                        x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2):...
                        y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1:...
                        x_ref+int_vec(1,2,1,2)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1:...
                        y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2):...
                        x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1:...
                        y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2)+1:...
                        x_ref+int_vec(1,2,1,2)+1+7,2))/4);
                end
            case 10
                if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==0))
```

```
            pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2);
        end
        if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2))/2);
        end
        if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==0))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1:...
                x_ref+int_vec(1,2,1,3)+7+1,2))/2);
        end
        if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1:...
                x_ref+int_vec(1,2,1,3)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3)+1:...
                x_ref+int_vec(1,2,1,3)+1+7,2))/4);
        end
    case 11
        if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==0))
            pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,2):...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                x_ref+int_vec(1,2,1,2)+7,2);
        end
        if((half_flag(1,2,1,2)==0)&(half_flag(1,2,2,2)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2):...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1:...
                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2):...
                x_ref+int_vec(1,2,1,2)+7,2))/2);
        end
        if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==0))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2):...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2):...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1:...
                x_ref+int_vec(1,2,1,2)+7+1,2))/2);
        end
        if((half_flag(1,2,1,2)==1)&(half_flag(1,2,2,2)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,2):...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2):...
                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2):...
                y_ref+int_vec(1,2,2,2)+7,x_ref+int_vec(1,2,1,2)+1:...
                x_ref+int_vec(1,2,1,2)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1:...
                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2):...
                x_ref+int_vec(1,2,1,2)+7,2)+pel_ref(y_ref+int_vec(1,2,2,2)+1:...
                y_ref+int_vec(1,2,2,2)+1+7,x_ref+int_vec(1,2,1,2)+1:...
                x_ref+int_vec(1,2,1,2)+1+7,2))/4);
        end
    case 12
        if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==0))
            pel_pred_backward(y,x)=pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2);
        end
        if((half_flag(1,2,1,3)==0)&(half_flag(1,2,2,3)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3):....
                x_ref+int_vec(1,2,1,3)+7,2))/2);
        end
        if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==0))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1:...
                x_ref+int_vec(1,2,1,3)+7+1,2))/2);
        end
        if((half_flag(1,2,1,3)==1)&(half_flag(1,2,2,3)==1))
            pel_pred_backward(y,x)=round((pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3):...
                x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3):...
                y_ref+int_vec(1,2,2,3)+7,x_ref+int_vec(1,2,1,3)+1:...
```

```
                                    x_ref+int_vec(1,2,1,3)+7+1,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                                    y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3):...
                                    x_ref+int_vec(1,2,1,3)+7,2)+pel_ref(y_ref+int_vec(1,2,2,3)+1:...
                                    y_ref+int_vec(1,2,2,3)+1+7,x_ref+int_vec(1,2,1,3)+1:...
                                    x_ref+int_vec(1,2,1,3)+1+7,2))/4);
                            end
                        end
                    end
                end
            end
        end
        p=round((pel_pred_forward+pel_pred_backward)/2);
end
for y=1:8
    for x=1:8
        d(y,x)=f(y,x)+p(y,x);
        if(d(y,x)<0)
            d(y,x)=0;
        end
        if(d(y,x)>255)
            d(y,x)=255;
        end
        x=x+1;
    end
    y=y+1;
end

if(i<=4)
    switch i
        case 1
            macro_luma(1:8,1:8)=d;
        case 2
            macro_luma(1:8,9:16)=d;
        case 3
            macro_luma(9:16,1:8)=d;
        case 4
            macro_luma(9:16,9:16)=d;
    end
else if(chroma_format==1)
        switch i
            case 5
                macro_chroma_b_8x8=d;
                for m=1:8
                    for n=1:8
                        fila=(m-1)*2+1;
                        columna=(n-1)*2+1;
                        macro_chroma_b(fila:fila+1,columna:columna+1)=macro_chroma_b_8x8(m,n);
                        n=n+1;
                    end
                    m=m+1;
                end

            case 6
                macro_chroma_r_8x8=d;
                for m=1:8
                    for n=1:8
                        fila=(m-1)*2+1;
                        columna=(n-1)*2+1;
                        macro_chroma_r(fila:fila+1,columna:columna+1)=macro_chroma_r_8x8(m,n);
                        n=n+1;
                    end
                    m=m+1;
                end

            end
    else if(chroma_format==2)
            switch i
                case 5
                    macro_chroma_b_8x8(1:8,1:8)=d;
                    for m=1:8
                        for n=1:8
                            fila=m;
                            columna=(n-1)*2+1;
                            macro_chroma_b(fila,columna:columna+1)=macro_chroma_b_8x8(m,n);
                            n=n+1;
                        end
                    m=m+1;
                    end
                case 6
                    macro_chroma_r_8x8(1:8,1:8)=d;
                    for m=1:8
                        for n=1:8
                            fila=m;
                            columna=(n-1)*2+1;
                            macro_chroma_r(fila,columna:columna+1)=macro_chroma_r_8x8(m,n);
```

```
                        n=n+1;
                    end
                m=m+1;
                end
            case 7
                macro_chroma_b_8x8(9:15,1:8)=d;
                for m=1:8
                    for n=1:8
                        fila=m+8;
                        columna=(n-1)*2+1;
                        macro_chroma_b(fila,columna:columna+1)=macro_chroma_b_8x8(m,n);
                        n=n+1;
                    end
                m=m+1;
                end
            case 8
                macro_chroma_r_8x8(9:16,1:8)=d;
                for m=1:8
                    for n=1:8
                        fila=m+8;
                        columna=(n-1)*2+1;
                        macro_chroma_r(fila,columna:columna+1)=macro_chroma_r_8x8(m,n);
                        n=n+1;
                    end
                m=m+1;
                end
        end
    else if(chroma_format==3)
            switch i
                case 5
                    macro_chroma_b(1:8,1:8)=d;
                case 6
                    macro_chroma_r(1:8,1:8)=d;
                case 7
                    macro_chroma_b(9:16,1:8)=d;
                case 8
                    macro_chroma_r(9:16,1:8)=d;
                case 9
                    macro_chroma_b(1:8,9:16)=d;
                case 10
                    macro_chroma_r(1:8,9:16)=d;
                case 11
                    macro_chroma_b(9:16,9:16)=d;
                case 12
                    macro_chroma_r(9:16,9:16)=d;
            end
        end
    end
  end
end
```

```
%*************************************************************************************************
%                                          Macroblock modes                                      %
%                                                          Nº of bits          Mnemonic          %
%   macroblock_modes() {                                                                         %
%       macroblock_type                                       1-9              vlclbf           %
%       if((spatial_temporal_weight_code_flag==1)&&                                              %
%           (spatial_temporal_weight_code_table_index!='00')) {                                  %
%           spatial_temporal_weight_code                        2              uimsbf           %
%       }                                                                                        %
%       if(macroblock_motion_forwardd||                                                          %
%           (macroblock_motion_bakward) {                                                        %
%           if(picture_structure=='frame') {                                                     %
%               if(frame_pred_frame_dct==0)                     2              uimsbf           %
%                   frame_motion_type                                                          %
%           } else {                                                                           %
%               field_motion_type                               2              uimsbf           %
%           }                                                                                  %
%       }                                                                                      %
%       if((picture_structure=="Frame picture")&&                                              %
%           (frame_pred_frame_dct==0)&&                                                        %
%           (macroblock_intra||macroblock_pattern)){                                           %
%           dct_type                                            1              uimsbf           %
%       }                                                                                      %
%   }                                                                                          %
%*************************************************************************************************

%Parametros de entrada
%   bitstream
%   picture_coding_type
%   picture_structure
%   frame_pred_frame_dct

%Parametros de salida
%   bits_leidos
%   macroblock_quant
%   macroblock_motion_forward
%   macroblock_motion_bakcward
%   macroblock_pattern
%   macroblock_intra
%   motion_vector_count
%   mv_format
%   dmv

function [bits_leidos,macroblock_quant,macroblock_motion_forward,macroblock_motion_backward, ...
    macroblock_pattern,macroblock_intra,frame_motion_type,motion_vector_count,mv_format,dmv]= ...
    macroblock_modes(bitstream,picture_coding_type,picture_structure,frame_pred_frame_dct)

k=1;

if(picture_coding_type==1) %Imagen I
    table_b2;   %Variable length codes for macroblock_type in I-pictures
    i=1;
    flag=0;

    while(flag==0)
        if(bitstream(k:k+num_bits(i)-1)==cod_long_var(i,1:num_bits(i)))
            macroblock_quant=macroblock_quant(i);
            macroblock_motion_forward=macroblock_motion_forward(i);
            macroblock_motion_backward=macroblock_motion_backward(i);
            macroblock_pattern=macroblock_pattern(i);
            macroblock_intra=macroblock_intra(i);
            spatial_temporal_weight_code_flag=spatial_temporal_weight_code_flag(i);
            permitted_spatial_temporal_weight_classes=permitted_spatial_temporal_weight_classes(i);
            spatial_temporal_weight_class=permitted_spatial_temporal_weight_classes;
            k=k+num_bits(i);
            flag=1;
        else
            i=i+1;
        end
    end
else if(picture_coding_type==2) %Imagen P
        table_b3;   %Variable length codes for macroblock_type in P-pictures
        i=1;
        flag=0;

        while(flag==0)
            if(bitstream(k:k+num_bits(i)-1)==cod_long_var(i,1:num_bits(i)))
                macroblock_quant=macroblock_quant(i);
                macroblock_motion_forward=macroblock_motion_forward(i);
                macroblock_motion_backward=macroblock_motion_backward(i);
                macroblock_pattern=macroblock_pattern(i);
                macroblock_intra=macroblock_intra(i);
                spatial_temporal_weight_code_flag=spatial_temporal_weight_code_flag(i);
                permitted_spatial_temporal_weight_classes=permitted_spatial_temporal_weight_classes(i);
                spatial_temporal_weight_class=permitted_spatial_temporal_weight_classes;
```

```
                k=k+num_bits(i);
                flag=1;
            else
                i=i+1;
            end
        end
    else    %Imagen B
        table_b4;   %Variable length codes for macroblock_type in B-pictures
        i=1;
        flag=0;

        while(flag==0)
            if(bitstream(k:k+num_bits(i)-1)==cod_long_var(i,1:num_bits(i)))
                macroblock_quant=macroblock_quant(i);
                macroblock_motion_forward=macroblock_motion_forward(i);
                macroblock_motion_backward=macroblock_motion_backward(i);
                macroblock_pattern=macroblock_pattern(i);
                macroblock_intra=macroblock_intra(i);
                spatial_temporal_weight_code_flag=spatial_temporal_weight_code_flag(i);
                permitted_spatial_temporal_weight_classes=permitted_spatial_temporal_weight_classes(i);
                spatial_temporal_weight_class=permitted_spatial_temporal_weight_classes;
                k=k+num_bits(i);
                flag=1;
            else
                i=i+1;
            end
        end
    end
end

if(spatial_temporal_weight_code_flag==1)
    bits=2;
    aux=bitstream(k:k+bits-1);
    spatial_temporal_weight_code=uint8(bin2dec(aux));
    k=k+bits;
end

if(macroblock_motion_forward==1 | macroblock_motion_backward==1)
    if(picture_structure==3) %Imagen trama
        if(frame_pred_frame_dct==0)
            bits=2;
            aux=bitstream(k:k+bits-1);
            frame_motion_type=uint8(bin2dec(aux));
            if(frame_motion_type==1)
                if(spatial_temporal_weight_class==0)
                    motion_vector_count=uint8(2);
                    mv_format='field';
                    dmv=uint8(0);
                else
                    motion_vector_count=uint8(1);
                    mv_format='field';
                    dmv=uint8(0);
                end
            else if(frame_motion_type==2)
                    motion_vector_count=uint8(1);
                    mv_format='frame';
                    dmv=uint8(0);
                else if(frame_motion_type==3)
                        motion_vector_count=uint8(1);
                        mv_format='field';
                        dmv=uint8(1);
                    end
                end
            end
            k=k+bits;
        else
            frame_motion_type=uint8(2);
            motion_vector_count=uint8(1);
            mv_format='frame';
            dmv=uint8(0);
        end
        if((macroblock_intra==1)&(concealment_motion_vectors==1))
            frame_motion_type=uint8(2);
            motion_vector_count=uint8(1);
            mv_format='frame';
            dmv=uint8(0);
        end
    else    %Imagen campo
        bits=2;
        aux=bitstream(k:k+bits-1);
        field_motion_type=uint8(bin2dec(aux));
        if(field_motion_type==1)
            motion_vector_count=uint8(1);
            mv_format='field';
            dmv=uint8(0);
```

```
        else if(frame_motion_type==2)
                motion_vector_count=uint8(2);
                mv_format='field';
                dmv=uint8(0);
            else if(frame_motion_type==3)
                   motion_vector_count=1;
                   mv_format='field';
                   dmv=uint8(1);
                end
            end
        end
        k=k+bits;
        if((macroblock_intra==1)&(concealment_motion_vectors=='1'))
            field_motion_type=1;
            motion_vector_count=uint8(1);
            mv_format='field';
            dmv=uint8(0);
        end
    end
else
    frame_motion_type=2;
    motion_vector_count=uint8(1);
    mv_format='frame';
    dmv=uint8(0);
end

if((picture_structure==3)&(frame_pred_frame_dct==0)&((macroblock_intra==1)|(macroblock_pattern==1)))
    bits=1;
    aux=bitstream(k:k+bits-1);
    dct_type=uint8(bin2dec(aux));
    k=k+bits;
else
    if(picture_structure~=3)
        dct_type='unused';%there is no frame/field distinction in a field picture
    end

    if(frame_pred_frame_dct==1)
        dct_type=uint8(0);
    end

    if(~((macroblock_intra==1)|(macroblock_pattern==1)))
        dct_type='unused';%macroblock is not coded
    end

end

bits_leidos=k-1;
```

```
%*************************************************************************************************************
%                                              Block                                                        %
%                                                                    Nº of bits          Mnemonic         %
%   block() {                                                                                             %
%       if(pattern_code[i] {                                                                              %
%           if(macroblock_intra) {                                                                        %
%               if(i<4) {                                             2-9               vlclbf           %
%                   dct_dc_size_luminance                                                                 %
%                   if(dct_dc_size_lumnancd!=0)                                                           %
%                       dct_dc_differential                           1-11              uimsbf           %
%               } else {                                                                                  %
%                   dct_dc_size_chrominance                           2-10              vlclbf           %
%                   if(dct_dc_size_chrominance!=0)                                                        %
%                       dct_dc_differenctial                          1-11              uimsbf           %
%               }                                                                                         %
%           } else {                                                                                      %
%               First DCT coefficent                                  2-24                               %
%           }                                                                                             %
%           while(nextbits()!=End of block)                                                               %
%               Subsequent DCT coefficcients                          3-24                               %
%           End of block                                              2 or 4            vlclbf           %
%       }                                                                                                 %
%   }                                                                                                     %
%*************************************************************************************************************

%Parametros de entrada
%   i
%   bitstream
%   pattern_code
%   macroblock_intra
%   dc_dct_pred
%   alternate_scan
%   precision
%   intra_quantiser_matrix
%   non_intra_quantiser_matrix
%   q_scale_type
%   quantiser_scale_code
%   intra_vlc_format

%Parametros de salida
%   f
%   bits_leidos
%   dc_dct_pred

function [f,bits_leidos,dc_dct_pred]=block(i,bitstream,pattern_code,macroblock_intra,dc_dct_pred, ...
    alternate_scan,precision,intra_quantiser_matrix,non_intra_quantiser_matrix,q_scale_type, ...
    quantiser_scale_code,intra_vlc_format)

k=1;

%matrices de cuantizacion
W_0=intra_quantiser_matrix;
W_1=non_intra_quantiser_matrix;

if(pattern_code==1)
    if(macroblock_intra==1)
        n=1;
        if(i<=4)
            cc=uint8(1); %Componente de luminancia

            table_b12;  %Variable length codes for dct_dc_size_luminance
            i=1;
            flag=0;
            while(flag==0)
                if(bitstream(k:k+num_bits(i)-1)==cod_long_var(i,1:num_bits(i)))
                    dct_dc_size_luminance=dct_dc_size_luminance(i);
                    k=k+num_bits(i);
                    flag=1;
                else
                i=i+1;
                end
            end

            if(dct_dc_size_luminance~=0)
                bits=double(dct_dc_size_luminance);
                aux=bitstream(k:k+bits-1);
                dct_dc_differential=uint16(bin2dec(aux));
                k=k+bits;
            end

            if(dct_dc_size_luminance==0)
                dct_diff=int16(0);
            else
                half_range=uint16(2^(dct_dc_size_luminance-1));
                if(dct_dc_differential>=half_range)
```

```
            dct_diff=int16(dct_dc_differential);
        else
            dct_diff=int16(dct_dc_differential+1)-int16(2*half_range);
        end
    end

    QFS(n)=dc_dct_pred(cc)+dct_diff;
    dc_dct_pred(cc)=QFS(1);
    n=n+1;
else
    if(i==5)    %Componente de chrominancia b
        cc=uint8(2);
    else if(i==6)
            cc=uint8(3);   %Componente de chrominancia r
        end
    end

    table_b13;  %Variable length codes for dct_dc_size_chrominance
    i=1;
    longitud_tabla=size(num_bits);
    flag=0;
    while(flag==0)
        if(bitstream(k:k+num_bits(i)-1)==cod_long_var(i,1:num_bits(i)))
            dct_dc_size_chrominance=dct_dc_size_chrominance(i);
            k=k+num_bits(i);
            flag=1;
        else
        i=i+1;
        end
    end

    if(dct_dc_size_chrominance~=0)
        bits=double(dct_dc_size_chrominance);
        aux=bitstream(k:k+bits-1);
        dct_dc_differential=uint16(bin2dec(aux));
        k=k+bits;
    end

    if(dct_dc_size_chrominance==0)
        dct_diff=int16(0);
    else
        half_range=uint16(2^(dct_dc_size_chrominance-1));
        if(dct_dc_differential>=half_range)
            dct_diff=int16(dct_dc_differential);
        else
            dct_diff=int16(dct_dc_differential+1)-int16(2*half_range);
        end
    end

    QFS(n)=dc_dct_pred(cc)+dct_diff;
    dc_dct_pred(cc)=QFS(n);
    n=n+1;
end
else %First DCT coefficient
    n=1;

    table_b14bis    %DCT coefficients Table zero. NOTE 3- This table shall be sused for the first
    flag=0;                                     %coefficient in the block (macroblock_intra!=0)
    i=1;
    while(flag==0)
        if(bitstream(k:k+num_bits(i)-1)==cod_long_var(i,1:num_bits(i)))
                if(i==31)
                    bits=6;
                    %ESCAPE code '0000 01'
                    k=k+bits;

                    bits=6;
                    aux=bitstream(k:k+bits-1);
                    run=uint8(bin2dec(aux));
                    k=k+bits;

                    bits=12;
                    aux=bitstream(k:k+bits-1);
                    aux_dec=uint16(bin2dec(aux));
                    if((aux_dec>=1)&(aux_dec<=2047))
                        level=int16(aux_dec);
                    else if((aux_dec>=2049)&(aux_dec<=4095))
                            level=int16(aux_dec)-int16(4096);
                        end
                    end
                    k=k+bits;
                    flag=1;

                else
                    run=valor_run(i);
```

```
                level=valor_level(i);
                k=k+num_bits(i);
                flag=1;
            end

        else
            i=i+1;
        end
    end

    for m=int8(0):int8(run)-int8(1)
        QFS(n)=int16(0);
        n=n+1;
    end

    QFS(n)=level;
    n=n+1;
end

eob_not_read=1;
if(intra_vlc_format==0)
    table_b14;  %DCT coefficientes Table zero
    end_of_block='10';
    bits_end=2;
else if(macroblock_intra==1)
        table_b15;  %DCT coefficents Table one
        end_of_block='0110';
        bits_end=4;
    else
        table_b14;  %DCT coefficients Table zero
        end_of_block='10';
        bits_end=2;
    end
end

while(eob_not_read==1)
    flag=0;
    i=1;
    if(bitstream(k:k+bits_end-1)==end_of_block)
        %end_of_block
        k=k+bits_end;
        eob_not_read=0;
        while(n<=64)
            QFS(n)=int16(0);
            n=n+1;
        end
    else
        while(flag==0)
            if(bitstream(k:k+num_bits(i)-1)==cod_long_var(i,1:num_bits(i)))
                if(i==31)
                    bits=6;
                    %ESCAPE code '0000 01'
                    k=k+bits;

                    bits=6;
                    aux=bitstream(k:k+bits-1);
                    run=uint8(bin2dec(aux));
                    k=k+bits;

                    bits=12;
                    aux=bitstream(k:k+bits-1);
                    aux_dec=uint16(bin2dec(aux));
                    if((aux_dec>=1)&(aux_dec<=2047))
                        level=int16(aux_dec);
                    else if((aux_dec>=2049)&(aux_dec<=4095))
                            level=int16(aux_dec)-int16(4096);
                        end
                    end
                    k=k+bits;
                    flag=1;
                else
                    run=valor_run(i);
                    level=valor_level(i);
                    k=k+num_bits(i);
                    flag=1;
                end
            else
            i=i+1;
            end
        end

        for m=int8(0):int8(run)-int8(1)
            QFS(n)=int16(0);
            n=n+1;
        end
```

```
        QFS(n)=level;
        n=n+1;

    end
end

if(alternate_scan==0)
    table_alternate_scan_0;
    for v=1:8
        for u=1:8
            QF(v,u)=QFS(scan_0(v,u));
        end
    end
else
    table_alternate_scan_1;
    for v=1:8
        for u=1:8
            QF(v,u)=QFS(scan_1(v,u));
        end
    end
end

if(macroblock_intra==1)
    if(precision==8)
        intra_dc_mult=uint8(8);
    else if(precision==9)
            intra_dc_mult=int8(4);
        else if(precision==10)
                intra_dc_mult=uint8(2);
            else
                intra_dc_mult=uint8(1);
            end
        end
    end
end

if(q_scale_type==0)
    quantiser_scale=quantiser_scale_code*2;
else
    quantiser_scale=uint8([0 1 2 3 4 5 6 7 8 10 12 14 16 18 20 22 24 28 32 36 40 44 48 52 56 64,...
                           72 80 88 96 104 112]);
    quantiser_scale=quantiser_scale(quantiser_scale_code+1);
end

for v=1:8
    for u=1:8
        if(((u==1)&(v==1))&(macroblock_intra==1))
            Fprimaprima(v,u)=int16(intra_dc_mult)*QF(v,u);
        else
            if(macroblock_intra==1)
                Fprimaprima(v,u)=(QF(v,u)*int16(W_0(v,u))*int16(quantiser_scale))*2/32;
            else
                Fprimaprima(v,u)=(((QF(v,u)*2)+sign(QF(v,u)))*int16(W_1(v,u))*int16(quantiser_scale))/32;
            end
        end
        u=u+1;
    end
    v=v+1;
end

sum=int16(0);
for v=1:8
    for u=1:8
        if(Fprimaprima(v,u)>2047)
            Fprima(v,u)=int16(2047);
        else if(Fprimaprima(v,u)<(-2048))
                Fprima(v,u)=int16(-2048);
            else
                Fprima(v,u)=Fprimaprima(v,u);
            end
        end
        sum=sum+Fprima(v,u);
        F(v,u)=Fprima(v,u);
        u=u+1;
    end
    v=v+1;
end

sum=(uint16(dec2bin(sum))-48);
bits_sum=size(sum);
uno_s(bits_sum(2))=uint16(1);
uno_s(1:bits_sum(2)-1)=0;

if(bitand(sum,uno_s)==0)
```

```
        F_comparo=(uint16(dec2bin(F(8,8)))-48);
        bits_F=size(F_comparo);
        uno_F(bits_F(2))=uint16(1);
        uno_F(1:bits_F(2)-1)=0;
        if(bitand(F_comparo,uno_F)~=0)
            F(8,8)=Fprima(8,8)-1;
        else
            F(8,8)=Fprima(8,8)+1;
        end
    end

    y=1:8;
    x=1:8;

    f(y,x)=idct2(F);

    for y=1:8
        for x=1:8
            if(f(y,x)<-256)
                f(y,x)=256;
            else if(f(y,x)>255)
                    f(y,x)=255;
                else
                    f(y,x)=f(y,x);
                end
            end
            x=x+1;
        end
    y=y+1;
    end
else
    x=1:8;
    y=1:8;
    f(y,x)=0;
end

f=int16(f);

bits_leidos=k-1;
```

```
%*************************************************************************************************
%                                          Motion vectors                                        %
%                                                                  Nº of bits          Mnemonic   %
%   motion_vectors(s) {                                                                           %
%       if(motion_vector_count==1) {                                                              %
%           if((mv_format==field)&&(dmv!=1))                                                      %
%               motion_vertical_field_select[0][s]                      1             uimsbf     %
%           motion_vector(0x,)                                                                    %
%       }else                                                                                     %
%           motion_vertical_field_select[0][s]                          1             uimsbf     %
%           motion_vector(0,s)                                                                    %
%           motion_vertical_field_select[1][s]                          1             uimsbf     %
%           motion_vector(1,s)                                                                    %
%   }                                                                                             %
%*************************************************************************************************

%Parametros de entrada
%   s
%   bitstream
%   motion_vector_count
%   mv_format
%   dmv
%   f_code
%   PMV
%   picture_structure
%   chroma_format

%Parametros de salida
%   bits_leidos
%   vector_Y
%   vector_Cb
%   vector_cr
%   PMV

function [bits_leidos,vector_Y,vector_Cb,vector_Cr,PMV]=motion_vectors(s,bitstream,motion_vector_count, ...
    mv_format,dmv,f_code,PMV,picture_structure,chroma_format)

k=1;
long_bitstream=size(bitstream);

if(s==2)
    s=1;
end

if(motion_vector_count==1)
    if((mv_format=='field')&(dmv~=1))
        bits=1;
        motion_vertical_field_select(1,s)=bitstream(k:k+bits-1);
        k=k+bits;
    end

    r=uint8(1); %Primer motion vector en Macroblock

    [bits,vector_prima(r,s,1:2),PMV]=motion_vector(r,s,bitstream(k:long_bitstream(2)),f_code,dmv,PMV, ...
        mv_format,picture_structure);
    k=k+bits;

    r=uint8(2); %No hay pero lo ponemos a 0
    vector_prima(r,s,1:2)=0;
else
    bits=1;
    motion_vertical_field_select(1,s)=bitstream(k:k+bits-1);
    k=k+bits;

    r=uint8(1); %Primer motion vector en Macroblock
    [bits,vector_prima_2,PMV]=motion_vector(r,s,bitstream(k:long_bitstream(2)),f_code,dmv,PMV, ...
        mv_format,picture_structure);
    vector_prima(r,s,1:2)=vector_prima_2;
    k=k+bits;

    bits=1;
    motion_vertical_field_select(2,s)=bitstream(k:k+bits-1);
    k=k+bits;

    r=uint8(2); %Segunado motion vector en Macroblock
    [bits,vector_prima,PMV]=motion_vector(r,s,bitstream(k:long_bitstream(2)),f_code,dmv,PMV, ...
        mv_format,picture_structure);
    vector_prima(r,s,1:2)=vector_prima;
    k=k+bits;

end

if(chroma_format==1)                %4:2:0
    vector_Y(1:2,s,1:2)=vector_prima(1:2,s,1:2);
    vector_Cb(1:2,s,1:2)=vector_prima(1:2,s,1:2)/2;
```

```
    vector_Cr(1:2,s,1:2)=vector_prima(1:2,s,1:2)/2;
else if(chroma_format==2)          %4:2:3
        vector_Y(1:2,s,1:2)=vector_prima(1:2,s,1:2);
        vector_Cb(1:2,s,1)=vector_prima(1:2,s,1)/2;
        vector_Cb(1:2,s,2)=vector_prima(1:2,s,2);
        vector_Cr(1:2,s,1)=vector_prima(1:2,s,1)/2;
        vector_Cr(1:2,s,2)=vector_prima(1:2,s,2);
    else if(chroma_format==3)      %4:4:4
            vector_Y(1:2,s,1:2)=vector_prima(1:2,s,1:2);
            vector_Cb(1:2,s,1:2)=vector_prima(r,s,1:2);
            vector_Cr(1:2,s,1:2)=vector_prima(1:2,s,1:2);
        end
    end
end

bits_leidos=k-1;
```

```
%*************************************************************************************************************
%                                              Motion vector                                                  %
%                                                                    Nº of bits          Mnemonic   %
%   motion_vectors() {                                                                                       %
%       motion_code[r][s][0]                                          1-10             vlclbf     %
%       if((f_code[s][0]!=1)&&(motion_code[r][s][0]!=0))                                                     %
%           motion_residual[r][s][0]                                   1-8              uimsbf     %
%       if(dmv==1)                                                                                           %
%           dmvector[0]                                                1-2              vlclbf     %
%       motion_code[r][s][1]                                          1-10             vlclbf     %
%       if((f_code[][1]!=1)&&(motion_coder[r][s][1]!=0))                                                     %
%           motion_residual[r][s][1]                                   1-8              uimsbf     %
%       if(dmv==1)                                                                                           %
%           dmvector[1]                                                1-2              vlclbf     %
%   }                                                                                                        %
%*************************************************************************************************************

%Parametros de entrada
%   r
%   s
%   bitstream
%   f_code
%   dmv
%   PMV
%   mv_format
%   picture_structure

%Parametros de salida
%   bits_leidos
%   vector_prima
%   PMV

function [bits_leidos,vector_prima,PMV]=motion_vector(r,s,bitstream,f_code,dmv,PMV,mv_format, ...
    picture_structure)

k=1;

t=uint8(1); %HORIZONTAL COMPONENT
r_size=f_code(s,t)-1;

table_b10;  %Variable lenght codes for motion_code
i=1;
flag=0;

while(flag==0)
    if(bitstream(k:k+num_bits(i)-1)==cod_long_var(i,1:num_bits(i)))
        motion_code(r,s,t)=valor_motion_code(i);
        k=k+num_bits(i);
        flag=1;
    else
        i=i+1;
    end
end

if((f_code(s,t)~=1)&(motion_code(r,s,t)~=0))
    bits=double(r_size);
    aux=bitstream(k:k+bits-1);
    motion_residual(r,s,t)=uint16(bin2dec(aux));
    k=k+bits;
else
    motion_residual(r,s,t)=uint16(0);
end

if(dmv==1)
    table_b11;
    i=1;
    flag=0;

    while(flag==0)
        if(bitstream(k:k+num_bits(i)-1)==cod_long_var(i,1:num_bits(i)))
            dmvector(t)=valor_dmvector(i);
            k=k+num_bits(i);
            flag=1;
        else
            i=i+1;
        end
    end
end

f=bitshift(1,r_size);
high=(16*f)-1;
low=((-16)*f);
range=(32*f);

if((f==1)|(motion_code(r,s,t)==0))
```

```
    delta=double(motion_code(r,s,t));
else
    delta=double(((abs(motion_code(r,s,t))-1)*f))+double(motion_residual(r,s,t))+1;
    if(motion_code(r,s,t)<0)
        delta=-delta;
    end
end
prediction=PMV(r,s,t);
if((mv_format== 'field')&(t==1)&(picture_structure==3))
    prediction=floor(PMV(r,s,t)/2);
end

vector_prima(r,s,t)=double(prediction+delta);
if(vector_prima(r,s,t)<low)
    vector_prima(r,s,t)=vector_prima(r,s,t)+range;
end
if(vector_prima(r,s,t)>high)
    vector_prima(r,s,t)=vector_prima(r,s,t)-range;
end

if((mv_format== 'field')&(t==1)&(picture_structure==3))
    PMV(r,s,t)=vector_prima(r,s,t)*2;
else
    PMV(r,s,t)=vector_prima(r,s,t);
end

t=uint8(2); %VERTICAL COMPONENT
r_size=f_code(s,t)-1;

table_b10;
i=1;
flag=0;

while(flag==0)
    if(bitstream(k:k+num_bits(i)-1)==cod_long_var(i,1:num_bits(i)))
        motion_code(r,s,t)=valor_motion_code(i);
        k=k+num_bits(i);
        flag=1;
    else
        i=i+1;
    end
end

if((f_code(s,t)~=1)&(motion_code(r,s,t)~=0))
    bits=double(r_size);
    aux=bitstream(k:k+bits-1);
    motion_residual(r,s,t)=uint16(bin2dec(aux));
    k=k+bits;
end

if(dmv==1)
    table_b11;
    i=1;
    flag=0;

    while(flag==0)
        if(bitstream(k:k+num_bits(i)-1)==cod_long_var(i,1:num_bits(i)))
            dmvector(t)=valor_dmvector(i);
            k=k+num_bits(i);
            flag=1;
        else
            i=i+1;
        end
    end
end

f=bitshift(1,r_size);
high=(16*f)-1;
low=((-16)*f);
range=(32*f);

if((f==1)|(motion_code(r,s,t)==0))
    delta=double(motion_code(r,s,t));
else
    delta=double(((abs(motion_code(r,s,t))-1)*f))+double(motion_residual(r,s,t))+1;
    if(motion_code(r,s,t)<0)
        delta=-delta;
    end
end
prediction=PMV(r,s,t);
if((mv_format== 'field')&(t==1)&(picture_structure==3))
    prediction=floor(PMV(r,s,t)/2);
end

vector_prima(r,s,t)=prediction+delta;
```

```
if(vector_prima(r,s,t)<low)
    vector_prima(r,s,t)=vector_prima(r,s,t)+range;
end
if(vector_prima(r,s,t)>high)
    vector_prima(r,s,t)=vector_prima(r,s,t)-range;
end

if((mv_format== 'field')&(t==1)&(picture_structure==3))
    PMV(r,s,t)=vector_prima(r,s,t)*2;
else
    PMV(r,s,t)=vector_prima(r,s,t);
end

bits_leidos=k-1;
```

```
%*************************************************************************************************************
%                                          Coded block pattern                                                %
%                                                                  Nº of bits          Mnemonic               %
%   coded_block_pattern() {                                                                                   %
%       coded_block_pattern_420                                      1-11              vlclbf                 %
%       if(chroma_format==4:2:2)                                                                              %
%           coded_block_pattern_1                                       2              uimsbf                 %
%       if(chroma_format==4:4:4)                                                                              %
%           coded_block_pattern_2                                       6              uimsbf                 %
%   }                                                                                                         %
%*************************************************************************************************************

%Parametros de entrada
%   bitstream
%   chroma_format

%Parametros de salida
%   bits_leidos
%cbp

function [bits_leidos,cbp]=coded_block_pattern(bitstream,chroma_format)

k=1;

table_b9;   %Variable length codes for coded_block_pattern
i=1;
flag=0;

while(flag==0)
    if(bitstream(k:k+num_bits(i)-1)==cod_long_var(i,1:num_bits(i)))
        cbp=cbp(i);
        k=k+num_bits(i);
        flag=1;
    else
        i=i+1;
    end
end
if(chroma_format==2) %4:2:2
    bits=2;
    aux=bitstream(k:k+bits-1);
    coded_block_pattern_1=uint8(bin2dec(aux));
    k=k+bits;
end
if(chroma_format==3) %4:4:4
    bits=6;
    aux=bitstream(k:k+bits-1);
    coded_block_pattern_2=uint8(bin2dec(aux));
    k=k+bits;
end

bits_leidos=k-1;
```

```
%*************************************************************************************************************%
%                    next start code                                                                          %
%                                                                                                             %
%   next_start_code() {                                                                                       %
%       while(!bytealigned())                                                                                 %
%           zero_bit                                                                                          %
%       while(nextbits()!='0000 0000 0000 0000 0000 0001')                                                    %
%           zero_byte                                                                                         %
%   }                                                                                                         %
%*************************************************************************************************************%

%Parametros de entrada
%   a=bytes del archivo completo
%   indice_actual

%Parametros de salida
%   indice

function indice=next_start_code(a,indice_actual)

i=indice_actual;
indice=0;

while (indice~=i)
    if ((a(i)==0) & (a(i+1)==0) & (a(i+2)==1))
        indice=i;
    else
        i=i+1;
    end
end
```

```
%*************************************************************************************************************%
%                                                                                                             %
%                    next start code de un valor                                                              %
%                                                                                                             %
%*************************************************************************************************************%

%Parametros de entrada
%   a=bytes del archivo completo
%   indice_actual
%   start_code_value

%Parametros de salida
%   indice

function indice=next_start_code_value(a,indice_actual,start_code_value)

tam_archivo=size(a);

i=indice_actual;
indice=0;
while (indice~=i)
    if(i<tam_archivo(1)-3)
        if ((a(i)==0) & (a(i+1)==0) & (a(i+2)==1) & a(i+3)==start_code_value)
            indice=i;
        else
        i=i+1;
        end
    else
        indice=tam_archivo(1);
        i=indice;
    end
end
```

```
%*************************************************************************************************************
%                     Table alternate scan_0                                                                 %
%*************************************************************************************************************

scan_0=[1 2 6 7 15 16 28 29
    3 5 8 14 17 27 30 43
    4 9 13 18 26 31 42 44
    10 12 19 25 32 41 45 54
    11 20 24 33 40 46 53 55
    21 23 34 39 47 52 56 61
    22 35 38 48 51 57 60 62
    36 37 49 50 58 59 63 64];
```

```
%*************************************************************************************************************
%                     Table alternate scan_1                                                                 %
%*************************************************************************************************************


scan_1=[1 5 7 21 23 37 39 53
    2 6 8 22 24 38 40 54
    3 9 20 25 35 41 51 55
    4 10 19 26 36 42 52 56
    11 18 27 31 43 47 57 61
    12 17 28 32 44 48 58 62
    13 16 29 33 45 49 59 63
    14 15 30 34 46 50 60 64];
```

```
%*************************************************************************************************
%                     Table B-1 --- Varible lengh codes for macroblock_address_increment               %
%*************************************************************************************************

%Numero de bits de cada palabra de codigo variable
num_bits=[1;3;3;4;4;6;6;7;7;8;8;8;8;8;8;10;10;10;10;10;10;11;11;11;11;11;11;11;11;11;11;11;11];

%Palabra de codigo variable
cod_long_var=char('1',...
                  '011',...
                  '010',...
                  '0011',...
                  '0010',...
                  '00011',...
                  '00010',...
                  '0000111',...
                  '0000110',...
                  '00001011',...
                  '00001010',...
                  '00001001',...
                  '00001000',...
                  '00000111',...
                  '00000110',...
                  '0000010111',...
                  '0000010110',...
                  '0000010101',...
                  '0000010100',...
                  '0000010011',...
                  '0000010010',...
                  '00000100011',...
                  '00000100010',...
                  '00000100001',...
                  '00000100000',...
                  '00000011111',...
                  '00000011110',...
                  '00000011101',...
                  '00000011100',...
                  '00000011011',...
                  '00000011010',...
                  '00000011001',...
                  '00000011000',...
                  '00000001000');

%Valor de increment_value
increment_value=uint8([1;2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17
    18;19;20;21;22;23;24;25;26;27;28;29;30;31;32;33;34]);
```

```
%*************************************************************************************************
%                     Table B-2 --- Varible lengh codes for macroblock_type in I-pictures              %
%*************************************************************************************************

%Numero de bits de cada palabra de codigo variable
num_bits=[1;2];

%Palabra de codigo variable
cod_long_var=char('1',...
                  '01');


%Valores a asignar

macroblock_quant=uint8([0;1]);

macroblock_motion_forward=uint8([0;0]);

macroblock_motion_backward=uint8([0;0]);

macroblock_pattern=uint8([0;0]);

macroblock_intra=uint8([1;1]);

spatial_temporal_weight_code_flag=uint8([0;0]);

permitted_spatial_temporal_weight_classes=uint8([0;0]);
```

```
%*************************************************************************************************
%                     Table B-3 --- Varible lengh codes for macroblock_type in P-pictures              %
%*************************************************************************************************

%Numero de bits de cada palabra de codigo variable
num_bits=[1;2;3;5;5;5;6];

%Palabra de codigo variable
cod_long_var=char('1',...
                  '01',...
                  '001',...
                  '00011',...
                  '00010',...
                  '00001',...
                  '000001');
%Valores a asignar

macroblock_quant=uint8([0;0;0;0;1;1;1]);

macroblock_motion_forward=uint8([1;0;1;0;1;0;0]);

macroblock_motion_backward=uint8([0;0;0;0;0;0;0]);

macroblock_pattern=uint8([1;1;0;0;1;1;0]);

macroblock_intra=uint8([0;0;0;1;0;0;1]);

spatial_temporal_weight_code_flag=uint8([0;0;0;0;0;0;0]);

permitted_spatial_temporal_weight_classes=uint8([0;0;0;0;0;0;0]);
```

```
%*************************************************************************************************
%                     Table B-4 --- Varible lengh codes for macroblock_type in B-pictures              %
%*************************************************************************************************

%Numero de bits de cada palabra de codigo variable
num_bits=[2;2;3;3;4;4;5;5;6;6;6];

%Palabra de codigo variable
cod_long_var=char('10',...
                  '11',...
                  '010',...
                  '011',...
                  '0010',...
                  '0011',...
                  '00011',...
                  '00010',...
                  '000011',...
                  '000010',...
                  '000001');

%Valores a asignar

macroblock_quant=uint8([0;0;0;0;0;0;0;1;1;1;1]);

macroblock_motion_forward=uint8([1;1;0;0;1;1;0;1;1;0;0]);

macroblock_motion_backward=uint8([1;1;1;1;0;0;0;1;0;1;0]);

macroblock_pattern=uint8([0;1;0;1;0;1;0;1;1;1;0]);

macroblock_intra=uint8([0;0;0;0;0;0;1;0;0;0;1]);

spatial_temporal_weight_code_flag=uint8([0;0;0;0;0;0;0;0;0;0;0]);

permitted_spatial_temporal_weight_classes=uint8([0;0;0;0;0;0;0;0;0;0;0]);
```

```
%*************************************************************************************************************
%                     Table B-9 --- Varible lengh codes for coded_block_pattern                              %
%*************************************************************************************************************

%Numero de bits de cada palabra de codigo variable
num_bits=[3;4;4;4;4;5;5;5;5;5;5;5;5;5;5;5;6;6;6;6;7;7;7;7;7;7;7;8;8;8;8
    8;8;8;8;8;8;8;8;8;8;8;8;8;8;8;8;8;8;8;8;8;8;8;9;9;9;9;9;9];

%Palabra de codigo variable
cod_long_var=char('111',...
                  '1101',...
                  '1100',...
                  '1011',...
                  '1010',...
                  '10011',...
                  '10010',...
                  '10001',...
                  '10000',...
                  '01111',...
                  '01110',...
                  '01101',...
                  '01100',...
                  '01011',...
                  '01010',...
                  '01001',...
                  '01000',...
                  '001111',...
                  '001110',...
                  '001101',...
                  '001100',...
                  '0010111',...
                  '0010110',...
                  '0010101',...
                  '0010100',...
                  '0010011',...
                  '0010010',...
                  '0010001',...
                  '0010000',...
                  '00011111',...
                  '00011110',...
                  '00011101',...
                  '00011100',...
                  '00011011',...
                  '00011010',...
                  '00011001',...
                  '00011000',...
                  '00010111',...
                  '00010110',...
                  '00010101',...
                  '00010100',...
                  '00010011',...
                  '00010010',...
                  '00010001',...
                  '00010000',...
                  '00001111',...
                  '00001110',...
                  '00001101',...
                  '00001100',...
                  '00001011',...
                  '00001010',...
                  '00001001',...
                  '00001000',...
                  '00000111',...
                  '00000110',...
                  '00000101',...
                  '00000100',...
                  '000000111',...
                  '000000110',...
                  '000000101',...
                  '000000100',...
                  '000000011',...
                  '000000010');

%Valor de cbp
cbp=uint8([60;4;8;16;32;12;48;20;40;28;44;52;56;1;61;2;62;24;36;3;63;5;9;17;33;6;10;18;34;7;11;19;35;13
    49;21;41;14;50;22;42;15;51;23;43;25;37;26;38;29;45;53;57;30;46;54;58;31;47;55;59;27;39]);
```

```
%*************************************************************************************************
%                     Table B-10 --- Varible lengh codes for motion_code                         %
%*************************************************************************************************

%Numero de bits de cada palabra de codigo variable
num_bits=[1;3;3;4;4;5;5;7;7;8;8;8;8;8;8;10;10;10;10;10;10;11;11;11;11;11;11;11;11;11;11;11];

%Palabra de codigo variable
cod_long_var=char('1',...
                  '010',...
                  '011',...
                  '0010',...
                  '0011',...
                  '00010',...
                  '00011',...
                  '0000110',...
                  '0000111',...
                  '00001010',...
                  '00001011',...
                  '00001000',...
                  '00001001',...
                  '00000110',...
                  '00000111',...
                  '0000010110',...
                  '0000010111',...
                  '0000010100',...
                  '0000010101',...
                  '0000010010',...
                  '0000010011',...
                  '00000100010',...
                  '00000100011',...
                  '00000100000',...
                  '00000100001',...
                  '00000011110',...
                  '00000011111',...
                  '00000011100',...
                  '00000011101',...
                  '00000011010',...
                  '00000011011',...
                  '00000011000',...
                  '00000011001');

%Valor de motion_code
valor_motion_code=int8([0;1;-1;2;-2;3;-3;4;-4;5;-5;6;-6;7;-7
    8;-8;9;-9;10;-10;11;-11;12;-12;13;-13;14;-14;15;-15;16;-16]);
```

```
%*************************************************************************************************
%                     Table B-11 --- Varible lengh codes for dmvector[t]                         %
%*************************************************************************************************

%Numero de bits de cada palabra de codigo variable
bits=[2;1;2];

%Palabra de codigo variable
var_long_code=char( '11',...
                    '0',...
                    '10');

%Valor de dmvector
valor_dmvector=[-1;0;1];
```

```
%*************************************************************************************************
%                     Table B-12 --- Varible lengh codes for dct_dc_size_luminance                 %
%*************************************************************************************************

%Numero de bits de cada palabra de codigo variable
num_bits=[3;2;2;3;3;4;5;6;7;8;9;9];

%Palabra de codigo variable
cod_long_var=char('100',...
                  '00',...
                  '01',...
                  '101',...
                  '110',...
                  '1110',...
                  '11110',...
                  '111110',...
                  '1111110',...
                  '11111110',...
                  '111111110',...
                  '111111111');

%Valor de dct_dc_size_luminance
dct_dc_size_luminance=uint8([0;1;2;3;4;5;6;7;8;9;10;11]);
```

```
%*************************************************************************************************
%                     Table B-13 --- Varible lengh codes for dct_dc_size_chrominance                %
%*************************************************************************************************

%Numero de bits de cada palabra de codigo variable
num_bits=[2;2;2;3;4;5;6;7;8;9;10;10];

%Palabra de codigo variable
cod_long_var=char('00',...
                  '01',...
                  '10',...
                  '110',...
                  '1110',...
                  '11110',...
                  '111110',...
                  '1111110',...
                  '11111110',...
                  '111111110',...
                  '1111111110',...
                  '1111111111');

%Valor de dct_dc_size_luminance
dct_dc_size_chrominance=[0;1;2;3;4;5;6;7;8;9;10;11];
```

```
%*************************************************************************************************************
%                     Table B-14 --- DCT coefficients Table zero                                                %
%*************************************************************************************************************

%Numero de bits de cada palabra de codigo variable
num_bits=[3;3;4;4;5;5;5;5;6;6;6;6;6;6;7;7;7;7;7;7;7;8;8;8;8;8;8;8;8;6;9;9;9;9;9;9;9;9;9;9;9;9
    9;9;9;11;11;11;11;11;11;11;11;11;11;11;11;11;11;11;11;13;13;13;13;13;13;13;13;13;13;13;13;13;13;13
    13;13;13;13;13;13;13;13;13;13;13;13;13;13;13;13;13;13;13;14;14;14;14;14;14;14;14;14;14;14;14;14;14;14
    14;14;14;14;14;14;14;14;14;14;14;14;14;14;14;14;14;14;14;15;15;15;15;15;15;15;15;15;15;15;15;15;15;15
    15;15;15;15;15;15;15;15;15;15;15;15;15;15;15;15;16;16;16;16;16;16;16;16;16;16;16;16;16;16;16;16;16;16
    16;16;16;16;16;16;16;16;16;16;16;16;16;16;16;16;17;17;17;17;17;17;17;17;17;17;17;17;17;17;17;17;17;17
    17;17;17;17;17;17;17;17;17;17;17;17;17;17;17;17];

%Palabra de codigo variable
cod_long_var=char('110',...
                  '111',...
                  '0110',...
                  '0111',...
                  '01000',...
                  '01001',...
                  '01010',...
                  '01011',...
                  '001010',...
                  '001011',...
                  '001110',...
                  '001111',...
                  '001100',...
                  '001101',...
                  '0001100',...
                  '0001101',...
                  '0001110',...
                  '0001111',...
                  '0001010',...
                  '0001011',...
                  '0001000',...
                  '0001001',...
                  '00001100',...
                  '00001101',...
                  '00001000',...
                  '00001001',...
                  '00001110',...
                  '00001111',...
                  '00001010',...
                  '00001011',...
                  '000001',...
                  '001001100',...
                  '001001101',...
                  '001000010',...
                  '001000011',...
                  '001001010',...
                  '001001011',...
                  '001001000',...
                  '001001001',...
                  '001001110',...
                  '001001111',...
                  '001000110',...
                  '001000111',...
                  '001000100',...
                  '001000101',...
                  '001000000',...
                  '001000001',...
                  '00000010100',...
                  '00000010101',...
                  '00000011000',...
                  '00000011001',...
                  '00000010110',...
                  '00000010111',...
                  '00000011110',...
                  '00000011111',...
                  '00000010010',...
                  '00000010011',...
                  '00000011100',...
                  '00000011101',...
                  '00000011010',...
                  '00000011011',...
                  '00000010000',...
                  '00000010001',...
                  '0000000111010',...
                  '0000000111011',...
                  '0000000110000',...
                  '0000000110001',...
                  '0000000100110',...
                  '0000000100111',...
                  '0000000100000',...
                  '0000000100001',...
```

```
'0000000110110',...
'0000000110111',...
'0000000101000',...
'0000000101001',...
'0000000111000',...
'0000000111001',...
'0000000100100',...
'0000000100101',...
'0000000111100',...
'0000000111101',...
'0000000101010',...
'0000000101011',...
'0000000100010',...
'0000000100011',...
'0000000111110',...
'0000000111111',...
'0000000110100',...
'0000000110101',...
'0000000110010',...
'0000000110011',...
'0000000101110',...
'0000000101111',...
'0000000101100',...
'0000000101101',...
'00000000110100',...
'00000000110101',...
'00000000110010',...
'00000000110011',...
'00000000110000',...
'00000000110001',...
'00000000101110',...
'00000000101111',...
'00000000101100',...
'00000000101101',...
'00000000101010',...
'00000000101011',...
'00000000101000',...
'00000000101001',...
'00000000100110',...
'00000000100111',...
'00000000100100',...
'00000000100101',...
'00000000100010',...
'00000000100011',...
'00000000100000',...
'00000000100001',...
'00000000111110',...
'00000000111111',...
'00000000111100',...
'00000000111101',...
'00000000111010',...
'00000000111011',...
'00000000111000',...
'00000000111001',...
'00000000110110',...
'00000000110111',...
'00000000001111110',...
'00000000001111111',...
'00000000001111100',...
'00000000001111101',...
'00000000001111010',...
'00000000001111011',...
'00000000001111000',...
'00000000001111001',...
'00000000001110110',...
'00000000001110111',...
'00000000001110100',...
'00000000001110101',...
'00000000001110010',...
'00000000001110011',...
'00000000001110000',...
'00000000001110001',...
'00000000001101110',...
'00000000001101111',...
'00000000001101100',...
'00000000001101101',...
'00000000001101010',...
'00000000001101011',...
'00000000001101000',...
'00000000001101001',...
'00000000001100110',...
'00000000001100111',...
'00000000001100100',...
'00000000001100101',...
'00000000001100010',...
```

```
                    '0000000000100011',...
                    '0000000000100000',...
                    '0000000000100001',...
                    '0000000000110000',...
                    '0000000000110001',...
                    '0000000000101110',...
                    '0000000000101111',...
                    '0000000000101100',...
                    '0000000000101101',...
                    '0000000000101010',...
                    '0000000000101011',...
                    '0000000000101000',...
                    '0000000000101001',...
                    '0000000000100110',...
                    '0000000000100111',...
                    '0000000000100100',...
                    '0000000000100101',...
                    '0000000000100010',...
                    '0000000000100011',...
                    '0000000000100000',...
                    '0000000000100001',...
                    '0000000000111110',...
                    '0000000000111111',...
                    '0000000000111100',...
                    '0000000000111101',...
                    '0000000000111010',...
                    '0000000000111011',...
                    '0000000000111000',...
                    '0000000000111001',...
                    '0000000000110110',...
                    '0000000000110111',...
                    '0000000000110100',...
                    '0000000000110101',...
                    '0000000000110010',...
                    '0000000000110011',...
                    '0000000000100110',...
                    '0000000000100111',...
                    '0000000000100100',...
                    '0000000000100101',...
                    '0000000000100010',...
                    '0000000000100011',...
                    '0000000000100000',...
                    '0000000000100001',...
                    '0000000000101000',...
                    '0000000000101001',...
                    '0000000000110100',...
                    '0000000000110101',...
                    '0000000000110010',...
                    '0000000000110011',...
                    '0000000000110000',...
                    '0000000000110001',...
                    '0000000000101110',...
                    '0000000000101111',...
                    '0000000000101100',...
                    '0000000000101101',...
                    '0000000000101010',...
                    '0000000000101011',...
                    '0000000000111110',...
                    '0000000000111111',...
                    '0000000000111100',...
                    '0000000000111101',...
                    '0000000000111010',...
                    '0000000000111011',...
                    '0000000000111000',...
                    '0000000000111001',...
                    '0000000000110110',...
                    '0000000000110111');

%Valores a asignar

valor_run=uint8([0;0;1;1;0;0;2;2;0;0;3;3;4;4;1;1;5;5;6;6;7;7;0;0;2;2;8;8;9;9;0;0;0;0;0;1;1;3;3;10;10;11
    11;12;12;13;13;0;0;1;1;2;2;4;4;5;5;14;14;15;15;16;16;0;0;0;0;0;0;0;0;1;1;1;2;3;3;4;4;6;6;7;7;8;8;17
    17;18;18;19;19;20;20;21;21;0;0;0;0;0;0;0;0;1;1;1;1;2;2;3;3;5;5;9;9;10;10;22;22;23;23;24;24;25;25;26
    26;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0
    0;0;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;6;6;11;11;12;12;13;13;14;14;15;15;16;16;27;27;28;28
    29;29;30;30;31;31]);

valor_level=int16([1;-1;1;-1;2;-2;1;-1;3;-3;1;-1;1;-1;2;-2;1;-1;1;-1;1;-1;4;-4;2;-2;1;-1;1;-1;0;5;-5;6;-6
    3;-3;2;-2;1;-1;1;-1;1;-1;1;-1;7;-7;4;-4;3;-3;2;-2;2;-2;1;-1;1;-1;1;-1;8;-8;9;-9;10;-10;11;-11;5;-5;4
    -4;3;-3;3;-3;2;-2;2;-2;2;-2;1;-1;1;-1;1;-1;1;-1;1;-1;12;-12;13;-13;14;-14;15;-15;6;-6;7;-7;5;-5;4;-4
    3;-3;2;-2;2;-2;1;-1;1;-1;1;-1;1;-1;1;-1;16;-16;17;-17;18;-18;19;-19;20;-20;21;-21;22;-22;23;-23;24
    -24;25;-25;26;-26;27;-27;28;-28;29;-29;30;-30;31;-31;32;-32;33;-33;34;-34;35;-35;36;-36;37;-37;38
    -38;39;-39;40;-40;8;-8;9;-9;10;-10;11;-11;12;-12;13;-13;14;-14;15;-15;16;-16;17;-17;18;-18;3;-3;2;-2
    2;-2;2;-2;2;-2;2;-2;2;-2;1;-1;1;-1;1;-1;1;-1;1;-1]);
```

```
%*************************************************************************************************************
%                     Table B-15 --- DCT coefficients Table one                                                %
%*************************************************************************************************************

%Numero de bits de cada palabra de codigo variable
num_bits=[3;3;4;4;4;4;6;6;5;5;6;6;7;7;6;6;7;7;8;8;8;8;6;6;8;8;8;8;8;8;6;6;6;7;7;8;8;9;9;8;8;9;9;9
    9;9;9;7;7;9;9;9;9;9;9;10;10;10;10;10;10;11;11;8;8;8;8;9;9;9;9;9;11;11;13;13;13;13;13;13;13;13
    13;13;13;13;13;13;13;13;13;13;13;9;9;9;9;9;9;9;14;14;14;14;14;14;14;14;14;14;14;14;14;14;14;14
    14;14;14;14;14;14;14;15;15;15;15;15;15;15;15;15;15;15;15;15;15;15;15;15;15;15;15;15;15;15;15;15
    15;15;15;15;15;16;16;16;16;16;16;16;16;16;16;16;16;16;16;16;16;16;16;16;16;16;16;16;16;16;16;16
    16;16;16;16;16;17;17;17;17;17;17;17;17;17;17;17;17;17;17;17;17;17;17;17;17;17;17;17;17;17;17;17
    17;17;17;17];

%Palabra de codigo variable
cod_long_var=char( '100',...
                   '101',...
                   '0100',...
                   '0101',...
                   '1100',...
                   '1101',...
                   '001010',...
                   '001011',...
                   '01110',...
                   '01111',...
                   '001110',...
                   '001111',...
                   '0001100',...
                   '0001101',...
                   '001100',...
                   '001101',...
                   '0001110',...
                   '0001111',...
                   '00001100',...
                   '00001101',...
                   '00001000',...
                   '00001001',...
                   '111000',...
                   '111001',...
                   '00001110',...
                   '00001111',...
                   '00001010',...
                   '00001011',...
                   '11110000',...
                   '11110001',...
                   '000001',...
                   '111010',...
                   '111011',...
                   '0001010',...
                   '0001011',...
                   '11110010',...
                   '11110011',...
                   '001001100',...
                   '001001101',...
                   '11110100',...
                   '11110101',...
                   '001000010',...
                   '001000011',...
                   '001001010',...
                   '001001011',...
                   '001001000',...
                   '001001001',...
                   '0001000',...
                   '0001001',...
                   '001001110',...
                   '001001111',...
                   '111111000',...
                   '111111001',...
                   '111111010',...
                   '111111011',...
                   '0000001000',...
                   '0000001001',...
                   '0000001010',...
                   '0000001011',...
                   '0000001110',...
                   '0000001111',...
                   '00000011010',...
                   '00000011011',...
                   '11110110',...
                   '11110111',...
                   '11111000',...
                   '11111001',...
                   '001000110',...
                   '001000111',...
                   '001000100',...
                   '001000101',...
```

```
            '001000000',...
            '001000001',...
            '000000011000',...
            '000000011001',...
            '0000000111000',...
            '0000000111001',...
            '0000000100100',...
            '0000000100101',...
            '0000000111100',...
            '0000000111101',...
            '0000000101010',...
            '0000000101011',...
            '0000000100010',...
            '0000000100011',...
            '0000000111110',...
            '0000000111111',...
            '0000000110100',...
            '0000000110101',...
            '0000000110010',...
            '0000000110011',...
            '0000000101110',...
            '0000000101111',...
            '0000000101100',...
            '0000000101101',...
            '111110100',...
            '111110101',...
            '111110110',...
            '111110111',...
            '111111100',...
            '111111101',...
            '111111110',...
            '111111111',...
            '000000000101100',...
            '000000000101101',...
            '000000000101010',...
            '000000000101011',...
            '000000000101000',...
            '000000000101001',...
            '000000000100110',...
            '000000000100111',...
            '000000000100100',...
            '000000000100101',...
            '000000000100010',...
            '000000000100011',...
            '000000000100000',...
            '000000000100001',...
            '000000000111110',...
            '000000000111111',...
            '000000000111100',...
            '000000000111101',...
            '000000000111010',...
            '000000000111011',...
            '000000000111000',...
            '000000000111001',...
            '000000000110110',...
            '000000000110111',...
            '0000000000111110',...
            '0000000000111111',...
            '0000000000111100',...
            '0000000000111101',...
            '0000000000111010',...
            '0000000000111011',...
            '0000000000111000',...
            '0000000000111001',...
            '0000000000110110',...
            '0000000000110111',...
            '0000000000110100',...
            '0000000000110101',...
            '0000000000110010',...
            '0000000000110011',...
            '0000000000110000',...
            '0000000000110001',...
            '0000000000101110',...
            '0000000000101111',...
            '0000000000101100',...
            '0000000000101101',...
            '0000000000101010',...
            '0000000000101011',...
            '0000000000101000',...
            '0000000000101001',...
            '0000000000100110',...
            '0000000000100111',...
            '0000000000100100',...
            '0000000000100101',...
            '0000000000100010',...
```

```
                    '0000000000100011',...
                    '0000000000100000',...
                    '0000000000100001',...
                    '00000000000110000',...
                    '00000000000110001',...
                    '00000000000101110',...
                    '00000000000101111',...
                    '00000000000101100',...
                    '00000000000101101',...
                    '00000000000101010',...
                    '00000000000101011',...
                    '00000000000101000',...
                    '00000000000101001',...
                    '00000000000100110',...
                    '00000000000100111',...
                    '00000000000100100',...
                    '00000000000100101',...
                    '00000000000100010',...
                    '00000000000100011',...
                    '00000000000100000',...
                    '00000000000100001',...
                    '00000000000111110',...
                    '00000000000111111',...
                    '00000000000111100',...
                    '00000000000111101',...
                    '00000000000111010',...
                    '00000000000111011',...
                    '00000000000111000',...
                    '00000000000111001',...
                    '00000000000110110',...
                    '00000000000110111',...
                    '00000000000110100',...
                    '00000000000110101',...
                    '00000000000110010',...
                    '00000000000110011',...
                    '000000000000100110',...
                    '000000000000100111',...
                    '000000000000100100',...
                    '000000000000100101',...
                    '000000000000100010',...
                    '000000000000100011',...
                    '000000000000100000',...
                    '000000000000100001',...
                    '000000000000101000',...
                    '000000000000101001',...
                    '000000000000110100',...
                    '000000000000110101',...
                    '000000000000110010',...
                    '000000000000110011',...
                    '000000000000110000',...
                    '000000000000110001',...
                    '000000000000101110',...
                    '000000000000101111',...
                    '000000000000101100',...
                    '000000000000101101',...
                    '000000000000101010',...
                    '000000000000101011',...
                    '000000000000111110',...
                    '000000000000111111',...
                    '000000000000111100',...
                    '000000000000111101',...
                    '000000000000111010',...
                    '000000000000111011',...
                    '000000000000111000',...
                    '000000000000111001',...
                    '000000000000110110',...
                    '000000000000110111');

%Valores a asignar

valor_run=uint8([0;0;1;1;0;0;2;2;0;0;3;3;4;4;1;1;5;5;6;6;7;7;0;0;2;2;8;8;9;9;0;0;0;0;0;1;1;3;3;10;10;11
    11;12;12;13;13;0;0;1;1;2;2;4;4;5;5;14;14;15;15;16;16;0;0;0;0;0;0;0;0;1;1;2;2;3;3;4;4;6;6;7;7;8;8;17
    17;18;18;19;19;20;20;21;21;0;0;0;0;0;0;0;0;1;1;1;1;2;2;3;3;5;5;9;9;10;10;22;22;23;23;24;24;25;25;26
    26;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0
    0;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;6;6;11;11;12;12;13;13;14;14;15;15;16;16;27;27;28;28;29
    29;30;30;31;31]);

valor_level=int16([1;-1;1;-1;2;-2;1;-1;3;-3;1;-1;1;-1;2;-2;1;-1;1;-1;1;-1;4;-4;2;-2;1;-1;1;-1;0;5;-5;6;-6
    3;-3;2;-2;1;-1;1;-1;1;-1;1;-1;7;-7;4;-4;3;-3;2;-2;2;-2;1;-1;1;-1;1;-1;8;-8;9;-9;10;-10;11;-11;5;-5;4
    -4;3;-3;3;-3;2;-2;2;-2;2;-2;1;-1;1;-1;1;-1;1;-1;1;-1;12;-12;13;-13;14;-14;15;-15;6;-6;7;-7;5;-5;4;-4
    3;-3;2;-2;2;-2;1;-1;1;-1;1;-1;1;-1;1;-1;16;-16;17;-17;18;-18;19;-19;20;-20;21;-21;22;-22;23;-23;24
    -24;25;-25;26;-26;27;-27;28;-28;29;-29;30;-30;31;-31;32;-32;33;-33;34;-34;35;-35;36;-36;37;-37;38
    -38;39;-39;40;-40;8;-8;9;-9;10;-10;11;-11;12;-12;13;-13;14;-14;15;-15;16;-16;17;-17;18;-18;3;-3;2;-2
    2;-2;2;-2;2;-2;2;-2;2;-2;1;-1;1;-1;1;-1;1;-1;1;-1]);
```

```
%*************************************************************************************************************
%                       Table Stream ID                                                                      %
%*************************************************************************************************************


id_13818_2=uint8([224 239]);

picture_start_code=uint8(0);
slice_start_code=uint8([1 175]);
user_data_start_code=uint8(178);
sequence_header_code=uint8(179);
sequence_error_code=uint8(180);
extension_start_code=uint8(181);
sequence_end_code=uint8(183);
group_start_code=uint8(184);

Sequence_Display_Extension_ID='0010';
Quant_Matrix_Extension_ID='0011';
Copyright_Extension_ID='0100';
Picture_Display_Extension_ID='0111';
Picture_Spatial_Scalabe_Extension_ID='1001';
```