```
/***********************************************************************
*
*  ldecod.c
*  Main program for H.26L decoder
*  Copyright (C) 1999  Telenor Satellite Services, Norway
*
*  Contacts:
*  Inge Lille-Langøy               <inge.lille-langoy@telenor.com>
*
*  Telenor Satellite Services
*  P.O.Box 6914 St.Olavs plass
*  N-0130 Oslo, Norway
*
***********************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <sys/timeb.h>
#include "global.h"


void main(int argc, char **argv)
{
  FILE *fd;                       /* pointer to input configuration file */
  FILE *p_in;                     /* pointer to input H.26L bitstream file */
  FILE *p_out;                    /* pointer to output YUV file */
  FILE *p_log;                    /* pointer to output log file */
  FILE *p_ref;                    /* pointer to input original reference YUV file
file */

  char config_filename[100];      /* configuration filename */

  time_t ltime1;                  /* for time measurement */
  time_t ltime2;
  struct _timeb tstruct1;
  struct _timeb tstruct2;
  char timebuf[128];
  int tot_time=0;                 /* time for total encoding session */
  int tmp_time;                   /* time used by decoding the last frame */

  struct inp_par    *inp;         /* input parameters from input configuration file
*/
  struct snr_par    *snr;
  struct img_par    *img;

  /* allocate memory for the structures */
  inp =  (struct inp_par *)calloc(1, sizeof(struct inp_par));
  snr =  (struct snr_par *)calloc(1, sizeof(struct snr_par));
```


INTEL-1011
10,142,659

```
  img =  (struct img_par *)calloc(1, sizeof(struct img_par));

  if (argc != 2)
  {
    fprintf(stdout,"Usage: %s <config.dat> \n",argv[0]);
    fprintf(stdout,"<config.dat> defines encoder parameters\n");
    exit(-1);
  }
  strcpy(config_filename,argv[1]);
  if((fd=fopen(config_filename,"r")) == NULL)           /* read the decoder
configuration file */
  {
    fprintf(stdout,"Error: Control file %s not found\n",config_filename);
    exit(0);
  }

  fscanf(fd,"%s",inp->infile);                /* YUV 4:2:2 input format */
  fscanf(fd,"%*[^\n]");

  fscanf(fd,"%s",inp->outfile);               /* H.26L compressed output bitsream
*/
  fscanf(fd,"%*[^\n]");

  fscanf(fd,"%s",inp->reffile);               /* reference file */
  fscanf(fd,"%*[^\n]");

  fscanf(fd,"%d",&inp->postfilter);           /* not used */
  fscanf(fd,"%*[^\n]");

  if ((p_in=fopen(inp->infile,"rb"))==0)
  {
    fprintf(stdout,"Input file %s does not exist \n",inp->infile);
    exit(0);
  }
  if ((p_out=fopen(inp->outfile,"wb"))==0)
  {
    fprintf(stdout,"Error open file %s  \n",inp->outfile);
    exit(0);
  }


fprintf(stdout,"-------------------------------------------------------------------
-------\n");
  fprintf(stdout," Decoder config file                    : %s
\n",config_filename);

fprintf(stdout,"-------------------------------------------------------------------
-------\n");
  fprintf(stdout," Input H.26L bitstream                  : %s \n",inp->infile);
  fprintf(stdout," Output decoded YUV 4:2:0               : %s \n",inp->outfile);
```

```
  fprintf(stdout," Output status file                    : log_dec.dat \n");
  if ((p_ref=fopen(inp->reffile,"rb"))==0)
  {
    fprintf(stdout," Input reference file                  : %s does not exist
\n",inp->reffile);
    fprintf(stdout,"                                         SNR values are not
available\n");
  }
  else
    fprintf(stdout," Input reference file                  : %s \n",inp->reffile);

fprintf(stdout,"-------------------------------------------------------------------
-------\n");


  fprintf(stdout,"Frame  TR    QP   SnrY    SnrU    SnrV   Time(ms)\n");

  init(img);

  img->number=0;
  while ((read_frame(p_in)) != 0)                    /* while not end of seqence,
read next image to buffer */
  {
    _ftime( &tstruct1 );                             /* start time ms */
    time( &ltime1 );                                 /* start time s  */

    if(image(img,inp)==SEARCH_SYNC)                  /* decode next frame */
      fprintf(stdout,"Biterror in frame %d, looking for synkword for next
frame\n",img->number);
    else
    {
      if (p_ref)
        find_snr(snr,img,p_ref,inp->postfilter);     /* if ref sequence exist */

      time( &ltime2 );                               /* end time sec */
      _ftime( &tstruct2 );                           /* end time ms  */
      tmp_time=(ltime2*1000+tstruct2.millitm) - (ltime1*1000+tstruct1.millitm);
      tot_time=tot_time + tmp_time;

      fprintf(stdout," %2d %5d %5d %7.4f %7.4f %7.4f %5d\n",
        img->number,img->tr,img->qp,snr->snr_y,snr->snr_u,snr->snr_v,
        tmp_time);

      write_frame(img,inp->postfilter,p_out);        /* write image to output YUV
file */
    }
    img->number++;                                   /* frame counter */
  }

  fprintf(stdout,"-------------------- Average SNR all frames
```

```
------------------------------\n");
  fprintf(stdout," SNR Y(dB)     : %5.2f\n",snr->snr_ya);
  fprintf(stdout," SNR U(dB)     : %5.2f\n",snr->snr_ua);
  fprintf(stdout," SNR V(dB)     : %5.2f\n",snr->snr_va);

fprintf(stdout,"------------------------------------------------------------------
-------\n");
  fprintf(stdout," Exit Telenor decoder, ver 4.0 \n");

  /*
  write to log file
  */
  if (fopen("log_dec.dat","r")==0)                    /* check if file exist */
  {
    if ((p_log=fopen("log_dec.dat","a"))==0)
    {
      fprintf(stdout,"Error open file %s for appending\n",p_log);
      exit(0);
    }
    else                                              /* Create header to new file
*/
    {
      fprintf(p_log,"
-------------------------------------------------------------------------------
-------\n");
      fprintf(p_log,"|  Decoder statistics. This file is made first time, later
runs are appended               |\n");
      fprintf(p_log,"
-------------------------------------------------------------------------------
------- \n");
      fprintf(p_log,"| Date  | Time  |    Sequence        |#Img|Format|SNRY 1|SNRU
1|SNRC 1|SNRV N|SNRU N|SNRV N|\n");
      fprintf(p_log,"
-------------------------------------------------------------------------------
-------\n");
    }
  }
  else
    p_log=fopen("log_dec.dat","a");                    /* File exist,just open for
appending */

  _strdate( timebuf );
  fprintf(p_log,"| %1.5s |",timebuf );

  _strtime( timebuf);
  fprintf(p_log," % 1.5s |",timebuf);

  fprintf(p_log,"%20.20s|",inp->infile);

  fprintf(p_log,"%3d |",img->number);
```

```
  if(img->format==QCIF)
    fprintf(p_log," QCIF |");
  else
    fprintf(p_log," CIF  |");

  fprintf(p_log,"%6.3f|",snr->snr_y1);
  fprintf(p_log,"%6.3f|",snr->snr_u1);
  fprintf(p_log,"%6.3f|",snr->snr_v1);
  fprintf(p_log,"%6.3f|",snr->snr_ya);
  fprintf(p_log,"%6.3f|",snr->snr_ua);
  fprintf(p_log,"%6.3f|\n",snr->snr_va);
}
/************************************************************************
*
*  Routine      init()
*
*  Description: Initilize some arrays
*
*  Input:       Image parameters
*
*  Output:      none
*
************************************************************************/
 void init(struct img_par *img)
 {
    int i;
    /* initilize quad matrix used in snr routine */

    for (i=0; i <  256; i++)
    {
      img->quad[i]=i*i; /* fix from TML 1, truncation removed */
    }


 }
```