User's Manual

ADVANSEE
Software Development Kit

Reference	ADV_UM_9009_advansee_sdk
Author	E.Poirier
Revision	1.1
Date	Oct 4th, 2013

ADV_UM - 1.3 - 20120608

ADVANSEE – 9 rue Alfred Kastler – CS30750 – 44307 Nantes cedex 3– France
Tél. : 33 (0)2 4050 2173 – Email : contact@advansee.com – www.advansee.com
SARL au capital de 40 200€ - 510 580 814 RCS Nantes - SIRET 510 580 814 00034- TVAI : FR 31 510 580 814

Document history

Version	Date	Author	Comments
1.0	Feb 14th 2013	E.Pauchard	Document first release
1.1	Oct 4th, 2013	E. Poirier	Updated for version 2 of AdvSDK

Table of Contents

1 Introduction

This manual is a guide to use Advansee's Software Development Kit version 2 (hereafter called “AdvSDK”). AdvSDK allows user to easily and quickly compile C or C++ applications for Advansee's Mipsee or Seeklop platforms.

AdvSDK is a set a pre-installed tools:

- Toolchains and staging directories for cross-compiling to Seeklop and Mipsee
- Advansee Library and its dependencies
- Kate text editor
- Advansee's project template

All these tools need a Linux machine to run. To ease AdvSDK integration on different platforms, we have created a virtual machine using VMware Player (http://www.vmware.com/products/player/). This virtual machine can be launched from a Windows (or Linux) interface, and will start a virtual computer running Linux with AdvSDK properly installed.

For a quick start on compiling code for Seeklop or Mipsee, please refer to section 5.

2 References

[Ref 1] ADV_UM_9009_LibAdvansee.pdf → Advansee library user guide

3 Package content

- advSDK_virtualmachine.tar.bz2 → Advansee's virtual machine
- ADV_UM_9009_advansee_sdk.pdf → AdvSDK user's manual
- ADV_UM_9009_LibAdvansee.pdf → Advansee's Library User's Manual

4 Virtual Machine installation

4.1 Install VmPlayer

Download VMware Player from VMware download center:

https://my.vmware.com/fr/web/vmware/free#desktop_end_user_computing/vmware_player

and install VMware Player on your computer.

4.2 Extract Advansee Virtual Machine

- Download and install 7-Zip from
  http://www.7-zip.org/download.html
  if you don't already have it (7-zip is a free open-source file archiver)
- Extract advSDK_virtualmachine.tar.bz2 to C:/VirtualMachine/AdvSDK/

4.3 Open Virtual Machine

- Launch VMware Player
- Click on “Open Virtual Machine”, and select Advansee's virtual machine file: AdvSDKv2.vmx
- Click on “Start Virtual Machine”

Your virtual machine running Ubuntu 13.04 should be starting.

4.4 Virtual Machine User

User is: **user**

Root password is: **user**

Default desktop interface is LXDE (but Unity is also installed).

5 Creating a new project

Note: This section describes the quickest way to get your application ready for Mipsee or Seeklop. If you want more details about Advansee's SDK, please refer to section 6.

To create a new project, it is recommended to copy one of the provided templates and then to add your own files. These templates are located in Advansee/Projects folder.

For a C project, enter:

```
cp -r ~/Documents/Advansee/Projects/Template_C  ~/Documents/MyProjects/MyProject1
```

For a C++ project, enter:

```
cp -r ~/Documents/Advansee/Projects/Template_Cpp  ~/Documents/MyProjects/MyProject1
```

5.1 Naming your project

Open “configure.ac” located at the root of your project:

in **AC_INIT** macro (line 5), replace:

- first argument (advansee-project) with the name of your project
- second argument (1.00) with the version of your project
- third argument (bug-report-address) with the contact email address

These parameters are important as:

- the name of your application will be the same as the name of your project.
- project name and version are stored in “config.h” file, so you can use these values in your application using PACKAGE_NAME, PACKAGE_VERSION definitions (among others).
- The main page of your project's documentation will contain project's name and version.

5.2 Adding files to your project

A template “main.c” is provided. In this file, **everything is already set up**: capture source, image transmission from platform to pc, program arguments... (for more information on Advansee library, please refer to [Ref 1]).

This file is ready to be compiled for a stand-alone application.

If you want to add files to your project, please follow this procedure:

1. Open “Makefile.am” located at the root of your project, and look for the line

```
template_SOURCES        =               $(srcdir)/src/main.c
```

2. Add your files to this variable. Example:

```
template_SOURCES        =               $(srcdir)/src/main.c \
                                        $(srcdir)/src/file1.c \
                                        $(srcdir)/src/file2.c \
                                        $(srcdir)/src/file1.h \
                                        $(srcdir)/src/file2.h
```

Note: the '\' character must be inserted between each line excepted at the end of the last one.

5.3 Adding libraries to your project

Using default configuration automatically includes following libraries:

- Advansee library
- Opencv (GUI, CORE and IMGPROC)

- Pthread
- VPU and IPU on Seeklop

If you need additional libraries:

1. Open “Makefile.am” located at the root of your project, and look for the lines

```
template_LDADD        =
template_CPPFLAGS     =
template_LDFLAGS      =
```

2. Add needed libraries to **template_LDADD**, using gcc syntax '-l'. Example:

```
template_LDADD              =       -ljpeg
```

3. Add your library headers search path to **template_CPPFLAGS,** using gcc syntax '-D'. Example:

```
template_CPPFLAGS           =       -D/home/advansee/jpeglib/include
```

4. Add your library binaries search path to **template_LDFLAGS**, using gcc syntax '-L'. Example:

```
template_LDFLAGS            =       -L/home/advansee/jpeglib/lib/
```

5.4 Quick Compile

This section describes all steps to compile an application for Advansee's platform:

Run commands from a terminal located in your project directory.

5.4.1 Seeklop

```
./adv_config -f adv_seeklop.conf -b seeklop -r release -O releaseSeeklop
cd releaseSeeklop
make clean all
make upload ADVHOST=seeklopAABBCC
```

- creates a directory named “releaseSeeklop”. This directory will be configured to build “release” target of project intended to be used on Seeklop platform.
- compiles your source code and create Seeklop executable.
- uploads application using SSH to your Seeklop (if connected to same network). Please replace *seeklopAABBCC* by your actual Seeklop's hostname.

5.4.2 Mipsee

```
./adv_config -f adv_mipsee.conf -b mipsee -r release -O releaseMipsee
cd releaseMipsee
make clean all
make upload ADVHOST=mipseeAABBCC
```

- creates a directory named “releaseMipsee”. This directory will be configured to build “release” target of project intended to be used on Mipsee platform.
- compiles your source code and create Mipsee executable.

- uploads application using SSH to your Mipsee (if connected to same network). Please replace *mipseeAABBCC* by your actual Mipsee's hostname (eg: mipsee00025d).

5.4.3 PC

```
./adv_config -f adv_pc.conf -b pc -r release -O releasePc
cd releaseMipsee
make clean all
```

- creates a directory named “releasePc”. This directory will be configured to build “release” target of project intended to be used on PC.
- compiles your source code and create executable for PC.

5.5 Running your application on Seeklop or Mipsee

The procedure described here is exactly the same for Mipsee or Seeklop.

5.5.1 Connect to your board using SSH

```
ssh root@seeklopAABBCC
```

Please replace *seeklopAABBCC* by your actual Seeklop or Mipsee hostname. If you don't know this hostname, try “seeklop” or “mipsee” followed by the last 6 numbers of the platform's MAC address (ex: seeklop000228).

Password is: **root**

5.5.2 Execute your application

```
./advansee-project
```

Note: the name of your application will be the same as your project's name, as defined in **configure.ac**.

5.5.3 Add your program to the automatic start-up list

This section shows how to automatically start your program when your platform is powered up.

On Seeklop or Mipsee, you just need to create an executable bash script named “**S99yourproject**” in the directory “**/etc/init.d/**”.

All scripts beginning by 'S' located in this directory will be executed on start-up.

All scripts beginning by 'K' located in this directory will be executed on system power-off.

Note: on Seeklop, you have to start camera driver before your program is executed. Use installed scripts in /root/ to do so.

6 AdvSDK usage

This section gives a more precise insight of AdvSDK internal functions.

6.1 Installed packages

These are all the packages needed to use AdvSDK (all these packages are already installed in your virtual machine):

- automake
- libtool
- doxygen
- opencv (2.4.2)
- pthread

6.2 Directories structure

In the following we describe all important directories installed on your Virtual Machine.

6.2.1 Compilation tools

- Staging directories

To cross-compile a project for Mipsee or Seeklop, compiler needs standard libraries and headers already compiled to desired platform. All these files are located in the “Staging Directory”, or “Sysroot”.

On AdvSDK, two of these directories (one for Mipsee and one for Seeklop) are available in:

```
/home/advansee/Documents/Advansee/StagingDirs
```

- Cross-Compile toolchains

Cross-compilators for Mipsee and Seeklop are already installed in:

```
/opt/adv-tools/
```

- **arm-advseeklop-linux-gnueabi** is the toolchain prefix for Seeklop
- **arm-1136jf_s_r1_vfp_hard_float-linux-gnueabi** is the toolchain prefix for Mipsee

For example, cross-compiler for Seeklop is: **arm-advseeklop-linux-gnueabi-gcc**

6.2.2 Advansee Library

Version 1.10 of Advansee Library is available in:

```
/home/advansee/Documents/Advansee/Library
```

This directory contains:

```
Library
└── libadvansee.1.10
    ├── documentation
```

```
│   └── html
├── include
│   ├── advCommandProtocol.h
│   ├── advFileAccess.h
│   ├── advSynchronize.h
│   ├── commandTcpServer.h
│   ├── trace.h
│   ├── videoSinkSeeklop.h
│   ├── videoSource.h
│   └── videoTcpFlow.h
├── libadvansee_mipsee
│   ├── debug
│   │   └── libadvansee.a
│   └── libadvansee.a
├── libadvansee_pc
│   ├── debug
│   │   └── libadvansee.a
│   └── libadvansee.a
├── libadvansee_seeklop
│   ├── debug
│   │   └── libadvansee.a
│   └── libadvansee.a
└── libadvansee_v1.10.tar.bz2
```

The Advansee Library has been compiled statically and the binary file is called **libadvansee.a**

Binaries are provided for 3 targets :

- PC (x86 32bits)
- Seeklop (Arm cortex A-8)
- Mipsee (Arm-11)

For each target, the **binary** is placed in the corresponding folder :

- libadvansee_mipsee : binary for MIPSEE
- libadvansee_seeklop : binary for SEEKLOP
- libadvansee_pc : binary for PC

For each target, a « debug » version is provided in the « debug » directory. The debug version of the library outputs more informations on stdout during execution of programs.

Library **Headers** are located in “include” directory.

Library's **documentation** in HTML format is available in **documentation/html/index.html**

Library's **user guide** in pdf format is available in the library's root directory.

Current library version is 1.10.

6.3 Project templates

A project template is given for C and C++ coding. These templates are available in:

```
/home/advansee/Documents/Advansee/Projects/Template_C
/home/advansee/Documents/Advansee/Projects/Template_Cpp
```

Projects are managed using GNU's Autotools, as any Linux package. This allow fast and easy configuration to multiple platforms and automatic creation of documentation.

6.3.1 Project Structure

Template projects has the following structure:

```
.
├── adv_config
├── adv_mipsee.conf
├── adv_pc.conf
├── adv_seeklop.conf
├── AUTHORS
├── ChangeLog
├── configure.ac
├── COPYING
├── documentation
│   ├── doxyfile.in
│   ├── logo_advansee_521x77.png
│   ├── mainpage.h.in
│   └── Makefile.am
├── INSTALL
├── Makefile.am
├── NEWS
├── README
└── src
    └── main.c
```

Most important files are indicated by the green color in above list. Their usage will be detailed later in this document, but to sum up:

- **Makefile.am** and **configure.ac** are the most important files of the project. They determine which source code should be used, where to find it, which libraries to use, … It is only necessary to modify these files on two situations:
  - If you add source files to your project, add them to Makefile.am
  - If you need additional library in your project, add them to configure.ac
- The three **.conf** files contains all information to compile a project for a specific platform. They are already set up, so you do not need to modify them at first.
- The **mainpage.h.in** is your documentation's main page (see section “Documentation”).
- The **src** directory should contain all your source code, but this is not mandatory

6.3.2 Template main.c

Provided main.c already includes several functions:

- state-machine system (init, operational, uninit)
- Using libadvansee:
  - image acquisition

- image streaming (the images can be displayed on a PC using the application AdvanseeGUI: change Hostname to comply with actual board, set video port to 5412, set cmd port to 4305) Nota: only the display part of AdvanseeGUI is usable with the application example provided by this main.c (the other controls need a specific software in the board).

- signal handling
- optional arguments parsing

To use the template program:

- use with -h option to see application usage:

```
./advansee-package -h
****************************************
*             Advansee                 *
* Template for Image Processing Project *
*                                      *
****************************************

Version: 1.00

-I- System Init
Usage: ./advansee-project [options]

Options:
-w | --webcam         Use a webcam as input source
-v | --video          Use a video as input source
-s | --still          Use image sequences as input source
-o | --ov7962         Use OV7962 as input source (only on advansee's Seeklop or Mipsee)
-t | --tcp            Use TCP Image Stream as input source (Advansee's Protocol)
-f | --file           Source file name {For file image source: filename; For Mipsee's ov7962: Device
name; For TCP: streaming server hostname}
-F | --format         Source file format {For File image source: file extension; For OV7962: image color
type (0-5)}
-S | --start          First sequence number
-E | --end            Last sequence number
-h | --help           Print this message
```

For TCP or OV7962 sources, you can specify expected image format using -F option, indicating the index number of image_type structure (videoSource.h):

```
-F image format indexes are:
0 → FORMAT_ADV_JPG
1 → FORMAT_ADV_GRAY
2 → FORMAT_ADV_UYVY
3 → FORMAT_ADV_YUV420
4 → FORMAT_ADV_NV12
5 → FORMAT_ADV_RGB565
6 → FORMAT_ADV_RGB
7 → FORMAT_ADV_BGR
```

With these options, you can dynamically change video source, for example:

- sequence of image files stored in /media/images/, named “images_XX.jpg” where XX is a number from 0 to 100:

```
./advansee-project -s -f /media/images/images_ -S 0 -E 100 -F jpg
```

- Advansee TCP of jpg image stream coming from distant board “seeklopAABBCC”:

```
./advansee-project -t -f seeklopAABBCC -F 0
```

- Grayscale Images from OV7962:

```
./advansee-project -o -f /dev/video0 -F 1
```

6.4 Configuring details

The first step is called “configuring”. In this step you will create Makefiles using your platform's information (cross-compiler, staging directories) and your projects info: name, version, dependencies, compile options,...

During configure step, you will create a build directory containing your final Makefile to compile your application. It is common to create such a build directory to maintain a clear separation between source code and build objects.

- Automatic Configuration:

You can start configuring by using advansee's script: adv_config. This script reads platform configuration values in adv_*.conf (ex: adv_seeklop.conf) and execute all steps to create your build directory.

Enter -h argument to get some help about this script:

```
advansee@advansee-vm:~/Documents/Advansee/Projects/Template$ ./adv_config -h
./adv_config version 110
Usage: ./adv_config [options]
Available options are
-f    configuration file: Configuration file in advansee format, containing all platform-dependant paths
and configuration values
-O    Output directory [default: build]
-b    Host board: pc, seeklop or mipsee [default pc]
-r    Debug/Release build: {debug, release}, [default: debug]
-h    Display this message
```

Specify with -f argument the .conf file you want to use, and with the -b argument the target platform (seeklop, mipsee or pc).

- Configuration files:

Unless you want to use specific configuration options (different cross-compiler, etc...), you should use provided configuration files (adv_seeklop.conf, adv_pc.conf, adv_mipsee.conf).

In that case, you need not to use “-f” option in the adv_config script.

- Example:

To create a build directory named “buildSeeklop” for target Seeklop in a debug version, use command:

```
./adv_config -b seeklop -O buildSeeklop -r debug
```

- When to configure:

Configuration step is required at first for build directory creation, but also each time you modify following files:

- configure.ac
- Makefile.am

Each new configuration in an existing folder will erase previous configuration.

6.4.1 Compiling and Linking options

All compiler options should be put in Makefile.am variables:

- **template_LDADD:** → additional libraries (use -l). Ex: -ljpeg
- **template_CPPFLAGS** → pre-processor options. Ex: -DDEBUG
- **template_CFLAGS** → C Compiler options. Ex: -std=c99
- **template_CXXFLAGS** → C++ Compiler options. Ex: -O3
- **template_LDFLAGS** → linker options. Ex: -g

6.4.2 Documentation

In your build directory, you will find a “documentation” folder. This folder contains your own project's documentation, generated by Doxygen (http://www.stack.nl/~dimitri/doxygen/)

Advansee library documentation is also automatically included to your project's documentation.

This documentation is created by analyzing your source code and in particular the doxygen special comment blocks (usually beginning by '/**').

To learn how to write your doxygen comments, please refer to doxygen website, or look at libadvansee's headers for examples.

Your documentation's mainpage is based on the template file “mainpage.h.in” located in the documentation folder **at the root of your project**.

- Creating your documentation

Your documentation is automatically created at compile time. If you want to force a complete update of your documentation, please use command:

```
make documentation
```

6.5 Compiling your code

Once your build directory is configured, it is easy to compile it:

in a terminal running inside your build directory, run:

```
make
```

to start compiling your project, and:

```
make clean
```

to clean your project (will remove all object files, cache and temporary files).

6.6 Using your applications

On PC, your application is directly usable.

On Seeklop and Mipsee, you first have to upload your application file and then to execute it on the platform.

6.6.1 Upload executable

From your Seeklop or Mipsee build directory, run:

```
make upload ADVHOST=seeklopAABBCC
```

This will upload your project's executable (using default project name) to the /root/ directory of the platform designed in the ADVHOST option.

Please replace seeklopAABBCC by the actual hostname of your Mipsee or Seeklop board.

6.6.1.1 Using your application on remote platform

Please refer to section 5.5.

7 Global Example

AdvSDK is installed with an already compilable source code. This code has different functions according to the platform it is running on:

- On Seeklop or Mipsee, it captures images from the OV7962 sensor and sends them to network
- On PC, it receives this network stream and display it on screen

In this example, we are going to install, configure, compile and execute this application.

7.1 Source Code

This source code is already provided in template “main.c”:

```
// Standard lib
#include <stdio.h>
#include <unistd.h>
#include <getopt.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>

// Opencv
#include "opencv/cv.h"
#include "opencv/highgui.h"

// Advansee lib headers:
#include "trace.h"
#include "videoSource.h"
#include "videoTcpFlow.h"

// Package header
#include "config.h"

//  Useful macros
#define _STRING(X) #X
#define STR(X) _STRING(X)
#define CLEAR(a) memset( &(a), 0, sizeof((a)))

/*! \brief System states
 *
 * Enumeration of possible states in main function
 *
 */
typedef enum
{
      INIT, OPERATIONAL, UNINIT
}SystemeStatesEnum;


/*! \brief usage function
 *
 * This function describes to user all possible arguments
 *
 */
static void usage(FILE *fp, int argc, char **argv)
```

```
{
  fprintf (fp,
                "Usage: %s [options]\n\n"
                "Options:\n"
                "-w | --webcam              Use a webcam as input source\n"
                "-v | --video        Use a video as input source\n"
                "-s | --still        Use image sequences as input source\n"
                "-o | --ov7962       Use OV7962 as input source (only on advansee's Seeklop or
Mipsee)\n"
                "-t | --tcp          Use TCP Image Stream as input source (Advansee's Protocol)\n"
                "-f | --file         Source file name {For file image source: filename; For Mipsee's
ov7962: Device name; For TCP: streaming server hostname}\n"
                "-F | --format       Source file format {For File image source: file extension; For
OV7962: image color type (0-5)}\n"
                "-S | --start        First sequence number\n"
                "-E | --end          Last sequence number\n"
                "-h | --help         Print this message\n"
                "",
                argv[0]);
}

/*! \brief Application arguments: short
 *
 * This structure defines all possible arguments for application
 *
 */
static const char short_options [] = "wvsf:F:S:E:e:hot";


/*! \brief Application arguments: long
 *
 * This structure defines all possible arguments for application
 *
 */
static const struct option long_options [] = {
  { "webcam",  no_argument,                    NULL,           'w' },
  { "video",   no_argument,                    NULL,           'v' },
  { "still",   no_argument,                    NULL,           's' },
  { "file",            required_argument,      NULL,           'f' },
  { "image type",      required_argument,      NULL,           'F' },
  { "file extension",  required_argument,      NULL,           'e' },
  { "start",   required_argument,      NULL,           'S' },
  { "end",             required_argument,      NULL,           'E' },
  { "help",            no_argument,            NULL,           'h' },
  { "ov7962",          no_argument,            NULL,           'o' },
  { "tcp",             no_argument,            NULL,           't' },
  { 0, 0, 0, 0 }
};


/*! \brief Static exit variable
 *
 * used to terminate application on certain signal
*/
static int exitFlag = 0;
```

```
/*! \brief Handle linux signal
 *
 * Callback function used with sigaction to redirect signal to specific actions
 * You can add special actions for SIGINT (Ctrl+C), SIGSEV (Segmentation Fault) or others
 *
 * \param signal      signal received
 *
 */
void adv_signal_handler(int signal)
{
      switch (signal)
      {
              case (SIGINT):
              {
                      TRACE_INFO("Interruption Signal\n");
                      exitFlag = 1;
              }break;
              case (SIGTSTP):
              {
                      TRACE_INFO("Stop Signal\n");
                      exit(-1);
              }break;
              case (SIGSEGV):
              {
                      TRACE_FATAL("Segmentation Fault\n");
                      exit(-1);
              }break;
      }
}



/** \brief Entry point of program
 *
* Main function is structured as a state machine:
*     - First state is INIT (variable init, only called once)
*     - Second state is OPERATIONAL (image processing, called in a loop)
*     - Last state is UNINIT (memory liberation, only called once)
*
* States are defined in the enum SystemeStatesEnum in main.c.
*
* Variables must be defined before entering state machine.
*
*/
int main(int argc, char *argv[])
{

      printf("****************************************\n");
      printf("*              Advansee                *\n");
      printf("* Template for Image Processing Project *\n");
      printf("*                                       *\n");
      printf("****************************************\n");
      printf("\nVersion: %s\n\n", PACKAGE_VERSION);

      /* Callback Signal Handler      */
      struct sigaction advSignalAction;
```

```
CLEAR(advSignalAction);
advSignalAction.sa_handler = adv_signal_handler;
sigaction(SIGINT, &advSignalAction, NULL);
sigaction(SIGTSTP, &advSignalAction, NULL);
sigaction(SIGSEGV, &advSignalAction, NULL);


/* Init Algorithm       */
SystemeStatesEnum localSystemState = INIT;

/* Alloc structures and variables       */
struct AdvVideoSource *videoSourceConf = adv_videoSourceCreate();
if (videoSourceConf == NULL)
{
        TRACE_ERROR("Could not create video source\n");
        return EXIT_FAILURE;
}

IplImage *grabFrame = NULL;
struct timeval tv;

struct AdvStreamServer *streamingConf = NULL;
struct AdvStreamBuffer *dataHeader = NULL;

/* Frame counter        */
int imageCnter = 0;
int imageErrorCnt=0;

while(1)
{
        switch(localSystemState)
        {

        /**------------------------------------------------------------------**/
        /**             INIT STATE: Initialize all variables                 **/
        /**             Only called once at first execution                  **/
        /**------------------------------------------------------------------**/
                case INIT:
                {

                        TRACE_INFO("System Init\n");

                        /* Set Camera Default Parameters */
                        videoSourceConf->filename = "/dev/video0";
                        videoSourceConf->extName = "jpg";
                        videoSourceConf->videoSourceType = VIDEO_SOURCE_PLATFORM_CAMERA;
                        videoSourceConf->width = 640;
                        videoSourceConf->height = 480;
                        videoSourceConf->start = 0;
                        videoSourceConf->stop = 55000;
                        videoSourceConf->sens = 1;
                        videoSourceConf->loop = 1;
                        videoSourceConf->imageType = FORMAT_ADV_GRAY;
                        videoSourceConf->fps = 30;

                        /* Get optionnal arguments and flags */
```

```
                        for (;;) {
                                int index;
                                int c;

                                c = getopt_long (argc, argv,short_options, long_options,&index);
                                if (-1 == c)
                                        break;

                                switch (c) {
                                        case 0: /* getopt_long() flag */
                                                break;

                                        case 'w':
                                                videoSourceConf->videoSourceType =
VIDEO_SOURCE_WEBCAM;
                                                break;

                                        case 'v':
                                                videoSourceConf->videoSourceType =
VIDEO_SOURCE_VIDEO;
                                                break;

                                        case 's':
                                                videoSourceConf->videoSourceType =
VIDEO_SOURCE_STILLS;
                                                break;

                                        case 'o':
                                                videoSourceConf->videoSourceType =
VIDEO_SOURCE_PLATFORM_CAMERA;
                                                break;

                                        case 't':
                                                videoSourceConf->videoSourceType =
VIDEO_SOURCE_TCP;
                                                break;

                                        case 'f':
                                                videoSourceConf->filename = optarg;
                                                break;

                                        case 'F':
                                                videoSourceConf->imageType = (int)atoi(optarg);
                                                break;

                                        case 'e':
                                                videoSourceConf->extName = optarg;
                                                break;

                                        case 'S':
                                                videoSourceConf->start = atoi(optarg);
                                                break;

                                        case 'E':
                                                videoSourceConf->stop = atoi(optarg);
                                                break;
```

```
                                        case 'h':
                                                usage (stdout, argc, argv);
                                                exit (EXIT_SUCCESS);

                                        default:
                                                usage (stderr, argc, argv);
                                                exit (EXIT_FAILURE);
                                }
                        }

                        /* Init structures */

                        /* Ipl Image */
                        if ((grabFrame = cvCreateImageHeader( cvSize(videoSourceConf->width,
videoSourceConf->height), IPL_DEPTH_8U, videoSourceConf->imageType == FORMAT_ADV_GRAY ? 1 : 3)) == NULL)
                        {
                                TRACE_ERROR("Memory allocation failed for %s in %s\n", "grabFrame",
__func__);
                                localSystemState=UNINIT;
                                break;
                        }

                        /* Video Source */
                        if (adv_videoSourceInit(videoSourceConf)<0)
                        {
                                TRACE_ERROR("Could not initialize video source\n");
                                return EXIT_FAILURE;
                        }

                        /* Streaming server */
                        streamingConf = adv_streamCreate(STREAM_PORT);

                        /* Streaming data structure */
                        dataHeader = adv_streamCreateImageHeader(videoSourceConf->imageType,
videoSourceConf->width, videoSourceConf->height, (videoSourceConf->height*videoSourceConf->width*3));


                        /* Go to next state */
                        localSystemState = OPERATIONAL;
                        TRACE_INFO("System Operationnal\n");
                }
                break;

/**-------------------------------------------------------------------------------------------**/
/**           OPERATIONAL STATE: Perform image processing loop                          **/
/**           This state is called for each image processed using the while(1) loop           **/
/**-------------------------------------------------------------------------------------------**/
                case OPERATIONAL:
                {
/**    START IMAGE PROCESSING LOOP     **/
                        /* Check for user interrupt */
                        if (exitFlag)
                        {
                                TRACE_INFO("Stopping Application...\n");
                                localSystemState = UNINIT;
                                break;
                        }
```

```
                        /* Get Frame and check error code */
                        if (adv_videoSourceGetFrame(videoSourceConf, grabFrame)!=0)
                        {
                                TRACE_INFO("Image Capture Error\n");

                                // Above 5 consecutive errors, stop application
                                if (imageErrorCnt++>5)
                                  localSystemState = UNINIT;
                                break;
                        }
                        imageErrorCnt = 0;

/** FROM THIS POINT, AN IPLIMAGE NAMED grabFrame IS FILLED WITH CAPTURED IMAGE **/
/** CODE INSERTED HERE WILL BE REPEATEDLY EXECUTED FOR EACH NEW CAPTURED FRAME **/

#ifndef PC
                        /* On Mipsee or Seeklop: */
                        /* send image via TCP */
                        memcpy(dataHeader->imagePtr, (unsigned char*)grabFrame->imageData,
grabFrame->imageSize);
                        gettimeofday(&tv, NULL);
                        adv_streamNewData(streamingConf, dataHeader, grabFrame->imageSize, &tv,
NULL, NULL);
#endif


#ifdef PC
                        /* On PC, display image */
                        cvShowImage("Test", grabFrame);
                        char esc ='1';
                        esc = cvWaitKey(33);

                        if (esc == 'q')
                                localSystemState = UNINIT;
#endif

                        /* Increment image counter      */
                        imageCnter++;
/**   END OF IMAGE PROCESSING LOOP    **/
                    }
                    break;

/**-------------------------------------------------------------------------------**/
/**   UNINIT STATE : Release all memory allocated during                          **/
/**          the init phase and exit program                                      **/
/**-------------------------------------------------------------------------------**/
                    case UNINIT:
                    {
                        TRACE_INFO("System Uninit\n");

                        /* release all structures */
                        adv_videoSourceClose(videoSourceConf);
                        adv_streamClose(streamingConf);
                        adv_streamReleaseImageHeader(dataHeader);
                        cvReleaseImageHeader(&grabFrame);
```

```
                        printf("Application Ends:\n");

                        /* end application */
                        return 0;
                    }
                    break;
                } // switch(localSystemState)
            }//while(1)
} // main
```

7.2 Installation

7.2.1 Creating project

```
cp -r ~/Documents/Advansee/Projects/Template_C  ~/Documents/AdvExample
```

7.2.2 Creating PC build directory

```
cd ~/Documents/AdvExample
./adv_config -b pc -O buildPC
```

7.2.3 Creating Seeklop build directory

```
cd ~/Documents/AdvExample
./adv_config -b seeklop -O buildSeeklop
```

7.2.4 Compiling code for PC

```
cd ~/Documents/AdvExample/buildPC
make
```

7.2.5 Compiling and uploading code for Seeklop

Connect a Seeklop to your local network. In this example, its hostname is seeklop000223

```
cd ~/Documents/AdvExample/buildSeeklop
make
make upload ADVHOST=seeklop000223
```

7.2.6 Executing on Seeklop

From one terminal, launch:

```
ssh root@seeklop000223
./advansee-project
```

7.2.7 Executing on PC

From an other terminal, launch:

```
cd buildPC
./advansee-project -t -f seeklop000223
```

Note: option -t indicates a TCP source, option -f indicates TCP source's hostname