

EVILibAAU – Controlling SONY EVI Cameras

M. Andersen and T.B. Moeslund

Lab. of Computer Vision and Media Technology
Aalborg University
Denmark
Email: tbm@imi.aau.dk

Abstract:

This technical report describes the cross platform C++ Library EVILibAAU. This library is under the terms of LGPL and is provided to ease the programming of SONY EVI cameras through their serial Viscas protocol. This report includes example code as well as compilation guides for Windows, Mac, and Linux.

Technical Report: CVMT-10-01 ISSN 1601-3646

April 2010

1 Introduction

EVILibAAU is a cross platform C++ library for controlling the serial visca-interface of Sony EVI video cameras. It is a fork of EVILib-3.0 by Pic Mickael. The library is meant for programmers and does not provide any visual interface to the implemented function nor does it handle any video data from the cameras. EVILibAAU is provided to ease the programming interface to the controlling of the camera (zoom, pan, focus, etc.). A typical system set-up is shown in Figure 1, where the camera is connected to a TV using a Composite video cable and connected to a computer using a visca-RS232 connection¹.

The library supports the camera models EVI-G20(21), EVI-D30(31), EVI-D70(P), and EVI-D100(P). The library, and the provided sample programs, are tested to work on Windows, Mac, and Linux². EVILibAAU is released under LGPL license and can be downloaded from <http://cvmt.aau.dk/~tbn/Sony/>.

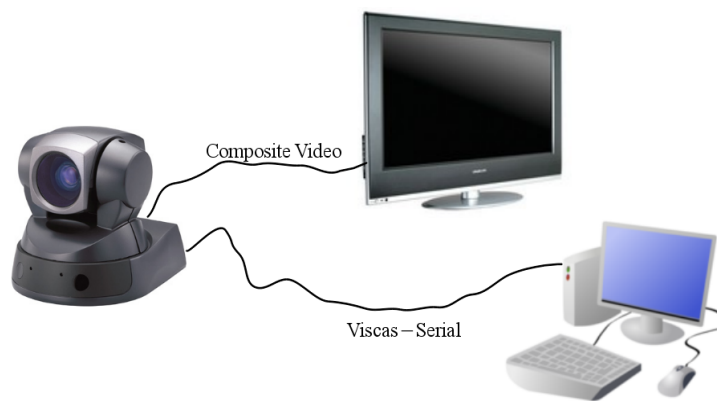


Figure 1: Typical set-up when controlling a Sony EVI camera.

2 Acknowledgement

EVILibAAU is a fork of EVILib by Pic Mickael (<http://sourceforge.net/projects/evilib/>). Thereby, EVILibAAU is not developed from scratch, it is a modification and extension of the EVILib library version 3.0 which is kindly provided under the open source license LGPL. By and large the interface from the EVILib has been kept unchanged. If you are using EVILib and want to switch to EVILibAAU you can see from the code examples that the changes you need to make are minor. The major difference is that EVILibAAU is extended also to run on Windows and Mac, and the library is extended to also support the camera models EVI-G20 and EVI-G21. The windows port is made using the LGPL library Pthreads-w32 (<http://sourceware.org/pthreads-win32/>).

This work is funded by the Danish National Research Councils (FTP) through the project: Big Brother is watching you!

¹On computers without a serial port a USB-serial converter can be used

²EVILibAAU is tested to work with Microsoft Visual C++ 2008 Express Edition at Windows Vista, GCC 4.3.3 at Ubuntu 9.04, and GCC 4.01 at Mac OS X 10.5.7

3 Features

EVILibAAU provides an object oriented C++ interface to the available functions of the supported cameras. Some of the basic functions are:

- Power (ON/OFF)
- Pan
- Tilt
- Zoom
- Focus
- White Balance
- Automatic Exposure

The advanced functions are also supported, e.g. the tracking capability of the EVI-D30(31), the alarm system of the EVI-D70(P) and the advanced picture effects of the EVI-D100(P). For more functions see the header-files of the library presented in Appendix A and the Sony camera documentation of your camera (The camera documentation are available on http://cvmt.aau.dk/~tbn/Sony/Technical_Manuals/).

Besides the function provided by the EVI-camera, the EVILibAAU provides the following functions:

getMax* and getMin* Returns the maximum/minimum value that this camera can handle for this type of command. E.g. `getMaxzoom` returns the maximum value that the function `Zoom(.)` can handle.

setAckTimeoutTime(.) When sending a command to the camera, the library expects to get an acknowledgement signal from the camera. Due to a broken connection this signal might not come and the library has an time out of 10 seconds. This time-out can be changed with the function `setAckTimeoutTime(.)`, and the current value can be read by the function `getAckTimeoutTime()`.

setCommandTimeoutTime(.) When a command is accepted by the camera, the library expects to get a completion signal from the camera, the library has an time out on 60 seconds (The completion time of a slow panning might be several seconds). This time-out can be changed with the function `setCommandTimeoutTime(.)`, and the current value can be read by the function `getCommandTimeoutTime()`.

CommandCancel(.) Used to cancel a command that is send to the camera but not finished yet. Input is the socket number that was used to send the command to the camera. The socket number is either 0 or 1. The socket number used for the last accepted command can be received by calling the function `getLastSocketNum()`.

4 EVILibAAU design

EVILibAAU is designed using virtual classes for common functionalities. In Figure 2 the class diagram of the important classes are shown. If you are looking for a particular camera function a good starting

point will be “in the top” of the class diagram. In the header file of the common virtual class *EVILibAAU* you get an overview of all functions available for more than one of the cameras. (When programming only the relevant file(s) from the bottom layer needs to be included.)

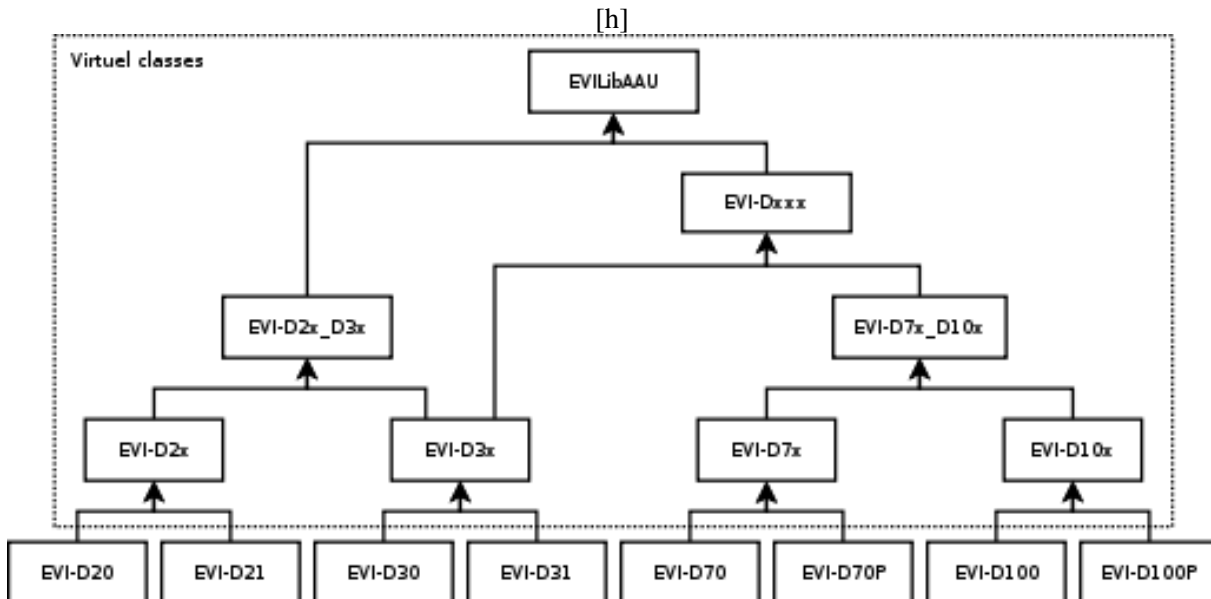


Figure 2: Class diagram containing the important classes of the library.

5 Code Examples

In this section some example code and the sample files included with the library are presented. All examples are made with the EVI-D31 camera, but the functions used are common for all the supported camera types, just replace “D31” with your camera type.

5.1 Code Introduction

The starting point of a program using EVILibAAU is to include the header file corresponding to the camera, e.g. `#include <EVI-D31.h>`. Before you can control the camera you must instance a class for the camera and open the serial port it is connected to:

```

EVI_D31 cam;
cam.Open( "NAME-OF-THE-COM-PORT" );

```

Where *NAME-OF-THE-COM-PORT* Will be something like *COM** on Windows and */dev/tty** on Mac and Linux.

If the camera is in power saving mode it needs to be turned on:

```

cam.Power( EVILIB_ON );

```

Now the initial steps are done and the camera can be controlled. E.g. zooming goes like this:

```

cam.Zoom( EVILIB_DIRECT, cam.getMax_zspeed(),
          cam.getMaxzoom(), EVILIB_NO_WAIT_COMP );

```

Where the first parameter is defining the zooming type, in this case we specify that we want to set the zooming parameters direct in the call. Second and third parameter is specifying the the speed and the amount of zooming. The last parameter specify that we do not want to wait until the zooming is done (this is explained in details in Section 5.3).

To test if the function succeed to do it's job the function return value is used. Normally the function return the value 1 upon success (see the header file to verify the value upon success/failure), e.g. to test if the opening of the camera goes as expected do like this:

```

int output = cam.Open( "NAME-OF-THE-COM-PORT" );
if( output==1 )
    cout << "SUCCESS" << endl;
else
    cout << "FAILED" << endl;

```

5.2 Complete Sample File

In this section a complete sample file with error check is shown (Similar behaviour as the file *samples/simple.cpp* provided with the library).

```

1 #include <EVI-D31.h>
2 #include <iostream>
3 #include <cstdlib>
4 //Typical port name when using a USB-serial cabel:
5 #ifdef __EVILIB_WINDOWS__
6     const char COM.PORT.NAME[] = "COM3";
7 #elif __APPLE__
8     const char COM.PORT.NAME[] = "/dev/tty.PL2303-0000101D";
9 #else
10    const char COM.PORT.NAME[] = "/dev/ttyUSB0";
11 #endif
12 using namespace std;
13
14 void exitProgram(const char* string)
15 {
16     cerr << "Error: " << string << endl;
17     exit(1);
18 }
19
20 int main(int argc, const char *argv[])
21 {
22     int output = 0;
23     EVI.D31 cam;
24
25     cout<<"Opening camera on port "<<<PORT.NAME<<<endl;
26     output = cam.Open(PORT.NAME);
27     if(output != 1) exitProgram("cam.Open");
28
29     cout<<"Turning the camera on"<<<endl;
30     output = cam.Power(EVILIB.ON);
31     if(output != 1) exitProgram("cam Power on");
32
33     cout<<"The camera is powered on and ready"<<<endl;
34

```

```

35 // Get and print camera pan and tilt position
36 float x,y;
37 output = camera.Pan_TiltPosInq(x, y);
38 if(output != 1) exitProgram("cam.Pan_TiltPosInq()");
39 cout<<"Camera position: x = "<< x <<" y = "<< y <<endl;
40
41 cout<<"Moveing"<<endl;
42 output = cam.Pan_TiltDrive(EVILIB_ABSOLUTE, cam.getMax_pspeed(),
    cam.getMax_tspeed(), cam.getMaxpan(), cam.getMintilt(),
    EVILIB_WAIT_COMP);
43 if(output != 1) exitProgram("cam.Pan_TiltDrive()");
44
45 cout<<"Setting the camera focus to automatic"<<endl;
46 output = cam.Focus(EVILIB_AUTO, 0, EVILIB_WAIT_COMP)
47 if(output != 1) exitProgram("cam.Focus()");
48
49 cout<<"Turning off the camera"<<endl;
50 output = cam.Power(EVILIB_OFF);
51 if(output != 1) exitProgram("Power(EVILIB_OFF)");
52
53 return 0;
54 }

```

5.3 Wait for Command Completion

The samples file *wait_vs_noWait.cpp* illustrates how to use the *waitC* parameter of the library functions that sends commands to the camera. The *waitC* parameter is the last parameter of the functions, it can take the values *EVILIB_WAIT_COMP* and *EVILIB_NO_WAIT_COMP*. The Sony EVI cameras supported by this Library have capacity to handle two commands at a time. E.g. if a zoom and a tilt/pan-command are send fast after each other, the camera will do both at same time. If a third command is send, the sending of the third command will be hold back by the library and send when the camera is ready. This means that the called function will not return until one of the first commands are completed and the third command are send to the camera. If two commands that affects the same functions are send, the camera will stop the first command when the second is received by the camera. E.g. if you want the cam to pan an area you can call a function with the parameter *EVILIB_WAIT_COMP*, this function call will not return before this command is actually completed. On the other hand if you want the camera to track something you probably want to send a new position to the camera as fast as possible and do not want to wait until the camera has reached the last position.

Here are the main code from the sample file *wait_vs_noWait.cpp* provided with the library:

```

1 ...
2 int main(int argc, const char *argv [])
3 {
4 ...
5 // Move the camera using a sin() path
6 cout<<"Now moving the camera without waiting for each command to
    complet"<<endl;
7 moveCamUsingSinPath(cam,40,EVILIB_NO_WAIT_COMP);
8 cout<<"Now moving the camera, and waiting for each command to
    complet"<<endl;
9 moveCamUsingSinPath(cam,15,EVILIB_WAIT_COMP);
10 ...

```

```

11 }
12 ...
13 void moveCamUsingSinPath(EVILibAAU& camera, int count, int waitC)
14 {
15     int direction=1, x_speed=camera.getMax_pspeed();
16     int y_speed=camera.getMax_tspeed();
17     float x_max = camera.getMaxpan(), delta_x = x_max/3.0f;
18     float y_max = camera.getMaxtilt(), x = -x_max, y;
19
20     for( int i=count ; i>=0 ; i— )
21     {
22         cout<<"moveCamUsingSinPath count down: "<< i <<endl;
23
24         y = y_max * sinf( x * (PI / 180.0f) );
25         int output = camera.Pan_TiltDrive(EVILIB.ABSOLUTE, x_speed, y_speed, x,
26             y*direction, waitC);
27         if(output != 1) exitProgram("camera.Pan_TiltDrive");
28
29         x = x + delta_x * direction;
30         if( abs(x) >= x_max)
31         {
32             x = x_max*direction;//truncate x value
33             direction = -1*direction;//switch direction
34         }
35     }
36 ...

```

6 Using the Sample Files

The library can be downloaded from <http://cvmt.aau.dk/~tbm/Sony/> as a zip-file. It contains four directories:

- src (containing the library source code)
- include (containing the library header files)
- samples (Containing sample files)
- windows-extra (contains pre build and pthreads-win32 files for Windows)

To use the sample files provided with the library you must set up your compiler and modify the sample code to suit your camera set-up. In the following sections the compiler set-up is described when using Visual C++ Express under Windows and using GCC under Mac and Linux. But first the modification of the sample files are explained.

To use the sample files you must tell the program which COM-port and EVI camera you are using. This is done in the file *samples/samples.h*. The COM-port is changed in this line of the file:

```
const char COM.PORT.NAME[] = "...";
```

Where the dots depends on your operation system. (On Windows something like *COM**, on Mac and Linux something like */dev/tty**). The sample files are written based upon a EVI-D31 camera, but all the used functions are available for all the supported EVI cameras. E.g. if you are using a EVI-G21 camera, just replace the two “D31” with “G21” in the *samples/samples.h* file.

6.1 GCC at Mac and Linux

First you needs to compile the library: Open a terminal and run *make* inside the *src* folder.

Then edit the file *samples/samples.h* to suit your EVI-camera type and your serial port (run *ls /dev/tty** to have a look at the available serial ports).

When the file are edit compile the sample programs by running *make* inside the *samples* folder. You can now run the sample files *simple* and *wait_vs_noWait* from a terminal.

6.2 Microsoft Visual C++ 2008 Express

Together with the library a pre-build static library for Windows is provided. In this section it is explained how to compile the sample code using Visual C++ Express.

6.2.1 Create a new Project

Start Visual C++ Express and make a new Win32 Console Application project:

- *File* → *New* → *Project*
- Select Win32 Console Application, choose a location (e.g. *C:/*) → OK (See Figure 3)
- Press Next
- Select *Console application* and *Empty project* (See Figure 4). Press *Finish*.

Now the Project are made and it is time to configure the settings.

6.2.2 Project Settings

Tell Visual C++ Express where to find the three directories *include*, *windows-extra/include* and *windows-extra/lib*: (In the following it is assumed that the zip-file is unextracted in the folder *C:/*)

- *Tools* → *Options*
- *Projects and Solutions* → *VC++ Directories* (See Figure 5)
- Add *C:/EVILibAAU-1.0/include* and *C:/EVILibAAU-1.0/windows-extra/include* to *Include files*
- Add *C:/EVILibAAU-1.0/windows-extra/lib* to *library files*

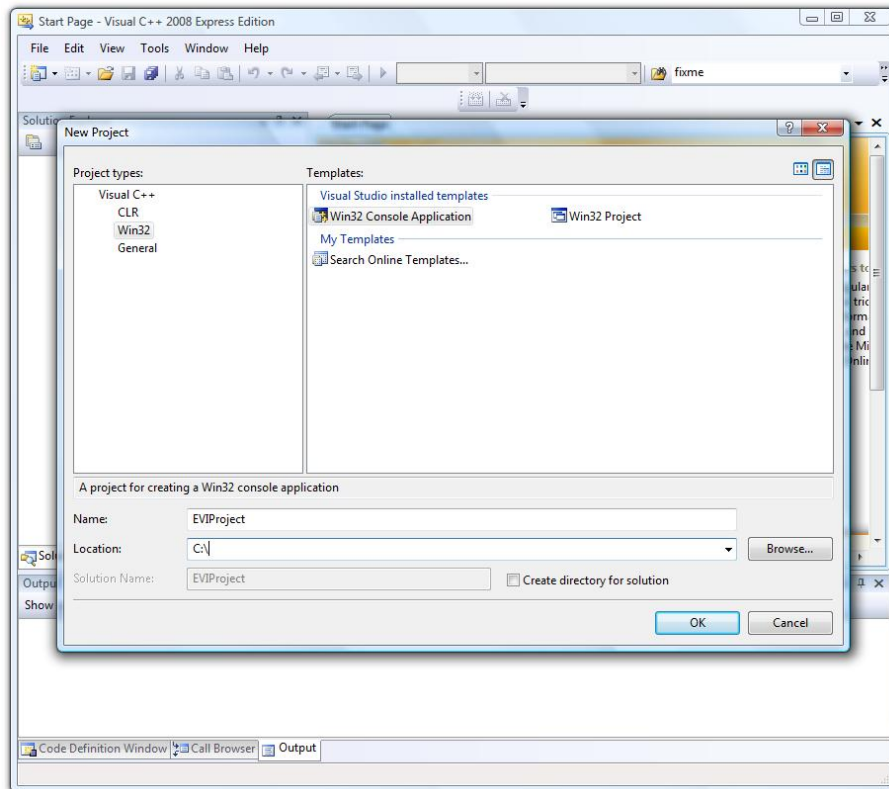


Figure 3: Creating a new Visual C++ Express Project no. 1

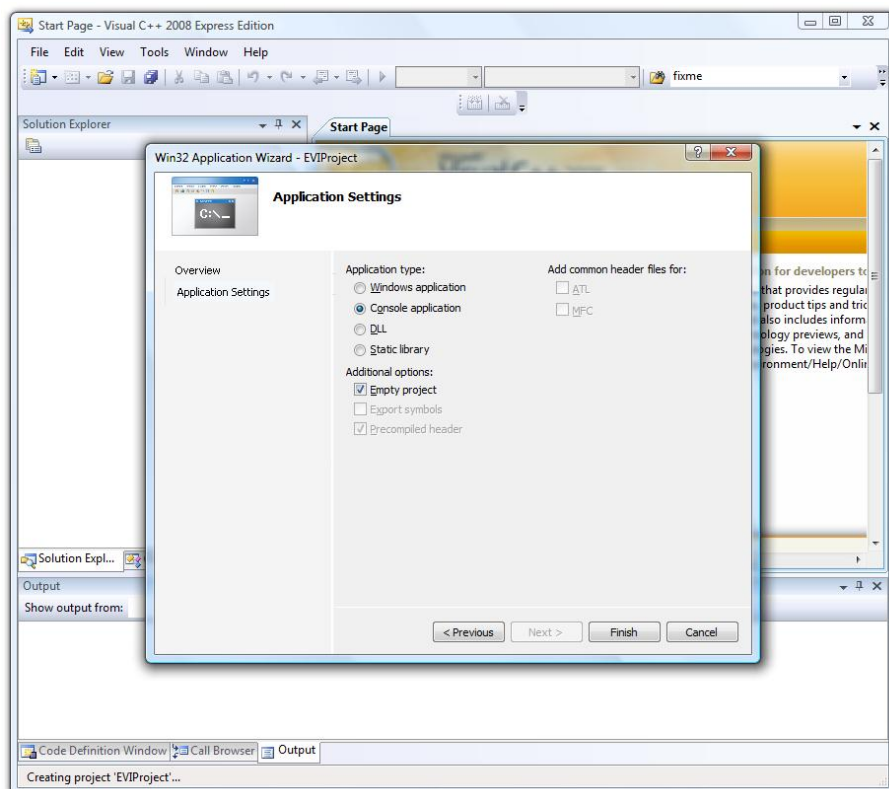


Figure 4: Make a new Visual Visual C++ Express Project no. 2

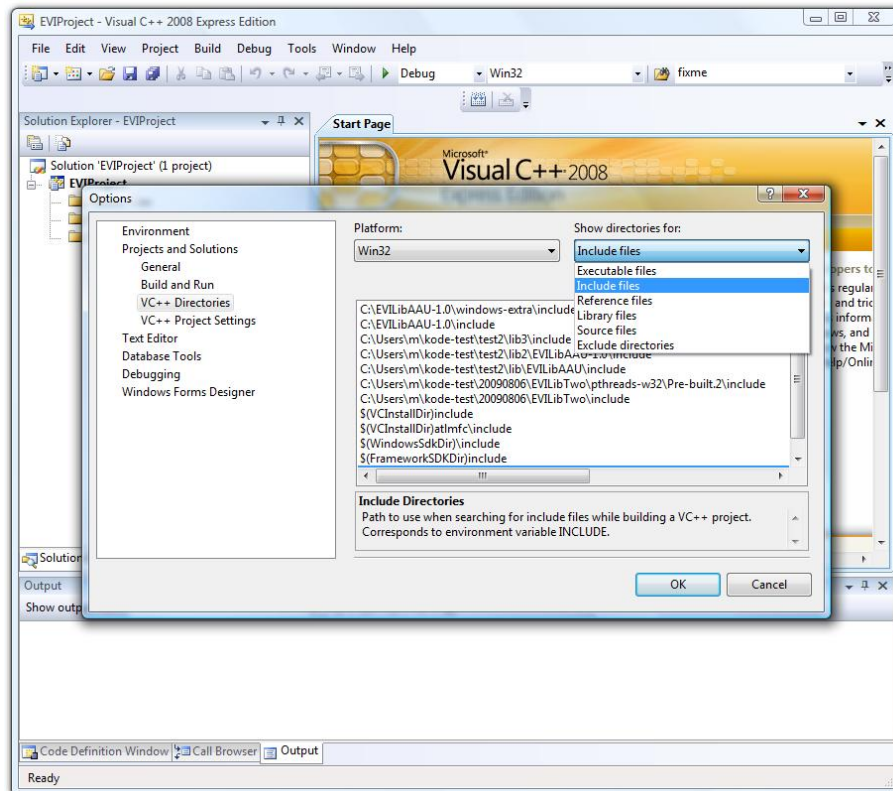


Figure 5: Add the directory containing the header files and the lib files to the Visual C++ Express Project.

- Press OK

Tell the linker to use the library-files.

- *Project* → *Properties*
- *Configuration Properties* → *Linker* → *Input* (See Figure 6)
- Add *pthreadVC2.lib* and *EVLlibAAU-1.0.lib* to the line *Additional Dependencies*
- Press OK

6.2.3 Include and Compile the Sample Files

Include the sample header file *samples.h* and one of the source file, *simple.cpp* or *wait_vs_noWait.cpp*, into the project:

- *Project* → *Add Existing Item* → (add one of the files from *samples/*)

You are now ready to compile and run the program. You just need to copy the file *windows-extra/lib/pthreadVC2.dll* to the directory of the exe-file (e.g. found in *C:/EVIPProject/Debug*).

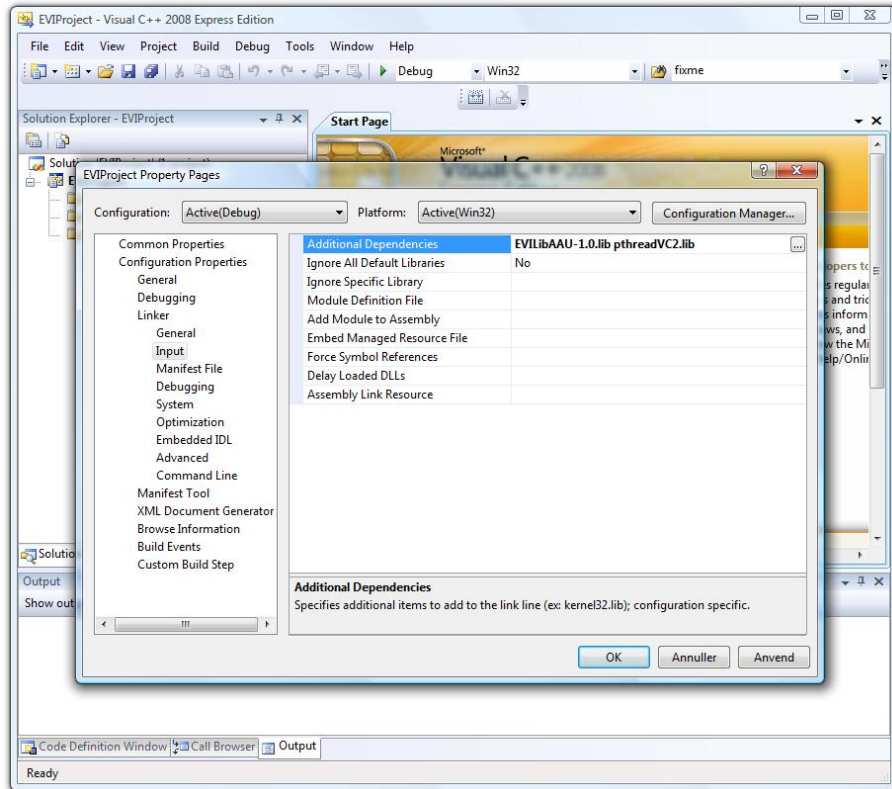


Figure 6: Add the library files to the linker setting in the Visual C++ Express Project.

The binary exe-file (e.g. found in *C:/EVIProject/Debug*) can be run on any other Windows computer (with a similar camera set-up) if you also copy the *pthreadVC2.dll* file into the same directory as the exe-file.

A Public part of the Class h-files

In the following the public part of the Class definition of the user relevant Classes are shown. This code shows the interface to all the functions implemented in the library. To get an overview of which Class headers that can be of interest for you specific EVI camera have a look at Section 4. As mention almost all functionality are made by Pic Mickael, and so are most of the code shown below.³

A.1 EVILibAAU.h

```

1 class EVILibAAU
2 {
3 public :
4     EVILibAAU(); // don't forget to call Open()!
5     virtual ~EVILibAAU();
6
7 // Set/Return Id_cam
8     inline void SetId_Cam(int id){Id_cam = id;}

```

³The code included here are under LGPL, the complete license is included with the source files.

```

9  inline int GetId_Cam() const {return Id_cam;}
10
11 //Set the timeout value used a command is send to the camera. [seconds]
12 // ackTimeoutTime is the time from a command is send until an answer is
    returned.
13 // CommandTimeoutTime is the maximum time a command are allowed to be
    executed on the camera (note that pan/tilt commands may take a while)
14 // return 1 if changed, or 0 if CommandTimeoutTime is becoming larger than
    ackTimeoutTime or smaller than 1[sec]
15 bool setAckTimeoutTime(int time);
16 bool setCommandTimeoutTime(int time);
17 //Get the timeout values [sec]
18 int getAckTimeoutTime();
19 int getCommandTimeoutTime();
20
21 // Functions to get the value of the camera variables parameters
22 const char* getCamera_type() const { return camera_type; }
23 int getMinGain() const { return minGain; }
24 int getMaxGain() const { return maxGain; }
25 float getMinpan() const { return minpan; }
26 float getMaxpan() const { return maxpan; }
27 int getMin_pspeed() const { return min_pspeed; }
28 int getMax_pspeed() const { return max_pspeed; }
29 float getMintilt() const { return mintilt; }
30 float getMaxtilt() const { return maxtilt; }
31 int getMin_tspeed() const { return min_tspeed; }
32 int getMax_tspeed() const { return max_tspeed; }
33 float getMinzoom() const { return minzoom; }
34 float getMaxzoom() const { return maxzoom; }
35 int getMin_zspeed() const { return min_zspeed; }
36 int getMax_zspeed() const { return max_zspeed; }
37 int getMin_iris() const { return min_iris; }
38 int getMax_iris() const { return max_iris; }
39 int getMinFocus() const { return minFocus; }
40 int getMaxFocus() const { return maxFocus; }
41 int getMin_FocusSpeed() const { return min_FocusSpeed; }
42 int getMax_FocusSpeed() const { return max_FocusSpeed; }
43 int getMax_num_memory_positions() const { return
    max_num_memory_positions; }
44
45 // Because the camera is not Highly precise on position , small difference
    can appear.
46 // This test allow to check if the value received is near the one asked
47 // Return 1 on success, 0 otherwise
48 inline int TestPanRange(float ask, float rec){if(ask - EVILIB_RANGE_VALUE
    < rec && rec < ask + EVILIB_RANGE_VALUE) return 1; else return 0;}
49
50 // Put the Camera to the Home position
51 // Return 1 on success, 0 on error
52 inline int Home(){return Pan_TiltDrive(EVILIB_ABSOLUTE, max_pspeed,
    max_tspeed, EVILIB_HomeX, EVILIB_HomeY, EVILIB_WAIT_COMP);}
53
54 // Status of the High-Low signal output
55 // Return 0 if no change occurred
56 // Return 1 if Low -> High edge
57 // Return -1 if High -> Low edge

```

```

58  inline int HighLowSignal(){return highLowSignal;}
59
60 //-----
61 // The following functions exist for all camera
62
63 // Open Serial Port – terminal settings
64 // Return 1 on success, 0 on error
65 // An error is most Likely caused if the cabel is connected badly or the
   portname is wrong
66 // If the camera are conected directly to the computer the id is 1.
67  int Open(int id, const char *portname);
68  int Open(const char *portname){ return Open(1,portname); }
69
70 // Send AddressSet command and IF_Clear command before starting
   communication.
71 // Return 1 on success, 0 on error
72 // There are no need to call this function explicit, unless you know wath
   you are doing
73  int AddressSet();
74
75 // Send AddressSet command and IF_Clear command before starting
   communication.
76 // Return 1 on success, 0 on error
77 // There are no need to call this function explicit, unless you know wath
   you are doing
78  int IF_Clear();
79
80 // socketNum: 0 or 1 (use e.g getLastSocketNum())
81 // Return 1 on success, 0 on error (most Likely if the camera is not
   powered on)
82  int CommandCancel( int socketNum );
83 // Return the socket number on the last used socket on the Camera (0 or 1)
   and -1 if not used yet
84  int getLastSocketNum() const { return lastAckSocketNum; }
85
86 // When camera main power is on, camera can be changed to Power Save Mode
87 // type: EVILIB_ON : set power on
88 //      EVILIB_Off : set power off
89 // Return 1 on success, 0 on error
90 // An error is most Likely caused if Open() is not called.
91  int Power(int type);
92
93 // Backlight compensation.
94 // Gain-up to 6 dB max.
95 // type: EVILIB_ON, EVILIB_OFF
96 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
   EVILIB_WAIT_COMP
97 // Return 1 on success, 0 on error
98 // An error is most Likely caused if the camera is not powered on (done by
   Power(EVILIB_ON))
99  int Backlight(int type, int waitC);
100
101 // Preset memory for memorize camera condition
102 // type: EVILIB_RESET —> need 'position'
103 //      EVILIB_SET —> need 'position'
104 //      EVILIB_RECALL — need 'position'

```

```

105 // position: 0 to maxNumMemoryPositions
106 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
107 // Return 1 on success, 0 on error
108 // An error is most Likely caused if the camera is not powered on (done by
    Power(EVILIB_ON))
109 int Memory(int type, int position, int waitC);
110
111 // type: EVILIB_UP → need 'pan_speed' & 'tilt_speed'
112 //     EVILIB_DOWN → need 'pan_speed' & 'tilt_speed'
113 //     EVILIB_LEFT → need 'pan_speed' & 'tilt_speed'
114 //     EVILIB_RIGHT → need 'pan_speed' & 'tilt_speed'
115 //     EVILIB_UPLEFT → need 'pan_speed' & 'tilt_speed'
116 //     EVILIB_UPRIGHT → need 'pan_speed' & 'tilt_speed'
117 //     EVILIB_DOWNLEFT → need 'pan_speed' & 'tilt_speed'
118 //     EVILIB_DOWNRIGHT → need 'pan_speed' & 'tilt_speed'
119 //     EVILIB_STOP → need 'pan_speed' & 'tilt_speed'
120 //     EVILIB_ABSOLUTE → need 'pan_speed' & 'tilt_speed' & 'pan_pos' &
    'tilt_pos'
121 //     EVILIB_RELATIVE → need 'pan_speed' & 'tilt_speed' & 'pan_pos' &
    'tilt_pos'
122 //     EVILIB_HOME
123 //     EVILIB_RESET
124 // pan_speed: pan speed min_pspeed to max_pspeed
125 // tilt_speed: tilt speed min_tspeed to max_tspeed
126 // pan_pos: pan position: approx. - maxpan to + maxpan
127 // tilt_pos: tilt position: approx. - mintilt to + maxtilt
128 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
129 // Return 1 on success, 0 on error
130 // An error is most Likely caused if the speed is out of range (or cam not
    powered on)
131 int Pan_TiltDrive(int type, int pan_speed, int tilt_speed, float pan_pos,
    float tilt_pos, int waitC);
132
133 // Return on success: EVILIB_ON, EVILIB_OFF
134 // Return on error: 0
135 // An error is most Likely caused if Open() is not called.
136 int PowerInq();
137
138 // Return on success: EVILIB_AUTO, EVILIB_MANUAL
139 // Return on error: 0
140 // An error is most Likely caused if the camera is not powered on (done by
    Power(EVILIB_ON))
141 int FocusModelnq();
142
143 // focus: contain the focus position of the camera
144 // Return 1 on success, 0 on error
145 // An error is most Likely caused if the camera is not powered on (done by
    Power(EVILIB_ON))
146 int FocusPosInq(int &focus);
147
148 // Return on success: EVILIB_AUTO, EVILIB_INDOOR, EVILIB_OUTDOOR
149 //     EVILIB_ONEPUSH_MODE, EVILIB_ATW, EVILIB_MANUAL
150 // Return on error: 0

```

```

151 // An error is most Likely caused if the camera is not powered on (done by
    Power(EVILIB_ON))
152  int WBModelnq();
153
154 // Return on success: EVILIB_AUTO, EVILIB_MANUAL, EVILIB_SHUTTER_PRIO
155 //                    EVILIB_IRIS_PRIO, EVILIB_GAIN_PRIO, EVILIB_BRIGHT
156 //                    EVILIB_SHUTTER_AUTO, EVILIB_IRIS_AUTO,
    EVILIB_GAIN_AUTO
157 // Return on error: 0
158 // An error is most Likely caused if the camera is not powered on (done by
    Power(EVILIB_ON))
159  int AEModelnq();
160
161 // Return on success: EVILIB_ON, EVILIB_OFF
162 // Return on error: 0
163 // An error is most Likely caused if the camera is not powered on (done by
    Power(EVILIB_ON))
164  int BacklightModelnq();
165
166 // gain: contain the iris position of the camera
167 // Return 1 on success, 0 on error
168 // An error is most Likely caused if the camera is not powered on (done by
    Power(EVILIB_ON))
169  int GainPosnq(int &gain);
170
171 // memory: contain the preset memory for memorize camera condition
172 // Return 1 on success, 0 on error
173 // An error is most Likely caused if the camera is not powered on (done by
    Power(EVILIB_ON))
174  int Memorynq(int &memory);
175
176 // After the completion of Pan_TiltModelnq, you can check the status with
    the functions provided
177 // Return 1 on success, 0 on error
178  int Pan_TiltModelnq();
179
180 // Return 1 if the 'Pan Left End' Pan/Tilter Status is true
181 // Return 0 otherwise. Remember to call Pan_TiltModelnq first!!!
182  inline int PanLeftEnd(){return PanTiltStatus[0];}
183
184 // Return 1 if the 'Pan Right End' Pan/Tilter Status is true
185 // Return 0 otherwise. Remember to call Pan_TiltModelnq first!!!
186  inline int PanRightEnd(){return PanTiltStatus[1];}
187
188 // Return 1 if the 'Tilt Upper End' Pan/Tilter Status is true
189 // Return 0 otherwise. Remember to call Pan_TiltModelnq first!!!
190  inline int TiltUpperEnd(){return PanTiltStatus[2];}
191
192 // Return 1 if the 'Tilt Down End' Pan/Tilter Status is true
193 // Return 0 otherwise. Remember to call Pan_TiltModelnq first!!!
194  inline int TiltDownEnd(){return PanTiltStatus[3];}
195
196 // Return 1 if the 'Pan Normal' Pan/Tilter Status is true
197 // Return 0 otherwise. Remember to call Pan_TiltModelnq first!!!
198  inline int PanNormal(){return PanTiltStatus[4];}
199

```

```

200 // Return 1 if the 'Pan Miss Position' Pan/Tilter Status is true
201 // Return 0 otherwise. Remember to call Pan_TiltModelnq first!!!
202 inline int PanMissPosition(){return PanTiltStatus[5];}
203
204 // Return 1 if the 'Pan Mechanical Disorder' Pan/Tilter Status is true
205 // Return 0 otherwise. Remember to call Pan_TiltModelnq first!!!
206 inline int PanMechanicalDisorder(){return PanTiltStatus[6];}
207
208 // Return 1 if the 'Tilt Normal' Pan/Tilter Status is true
209 // Return 0 otherwise. Remember to call Pan_TiltModelnq first!!!
210 inline int TiltNormal(){return PanTiltStatus[7];}
211
212 // Return 1 if the 'Tilt Miss Position' Pan/Tilter Status is true
213 // Return 0 otherwise. Remember to call Pan_TiltModelnq first!!!
214 inline int TiltMissPosition(){return PanTiltStatus[8];}
215
216 // Return 1 if the 'Tilt Mechanical Disorder' Pan/Tilter Status is true
217 // Return 0 otherwise. Remember to call Pan_TiltModelnq first!!!
218 inline int TiltMechanicalDisorder(){return PanTiltStatus[9];}
219
220 // Return 1 if the 'No Drive Command' Pan/Tilter Status is true
221 // Return 0 otherwise. Remember to call Pan_TiltModelnq first!!!
222 inline int NoDriveCommand(){return PanTiltStatus[10];}
223
224 // Return 1 if the 'Pan, Tilt In-Move' Pan/Tilter Status is true
225 // Return 0 otherwise. Remember to call Pan_TiltModelnq first!!!
226 inline int PanTiltInMove(){return PanTiltStatus[11];}
227
228 // Return 1 if the 'Pan, Tilt Drive Completion' Pan/Tilter Status is true
229 // Return 0 otherwise. Remember to call Pan_TiltModelnq first!!!
230 inline int PanTiltDriveCompletion(){return PanTiltStatus[12];}
231
232 // Return 1 if the 'Pan, Tilt Drive failure' Pan/Tilter Status is true
233 // Return 0 otherwise. Remember to call Pan_TiltModelnq first!!!
234 inline int PanTiltDriveFailure(){return PanTiltStatus[13];}
235
236 // Return 1 if the 'Before Initialize' Pan/Tilter Status is true
237 // Return 0 otherwise. Remember to call Pan_TiltModelnq first!!!
238 inline int BeforeInitialize(){return PanTiltStatus[14];}
239
240 // Return 1 if the 'In-Initialize' Pan/Tilter Status is true
241 // Return 0 otherwise. Remember to call Pan_TiltModelnq first!!!
242 inline int InInitialize(){return PanTiltStatus[15];}
243
244 // Return 1 if the 'Initialize complete' Pan/Tilter Status is true
245 // Return 0 otherwise. Remember to call Pan_TiltModelnq first!!!
246 inline int InitializeComplete(){return PanTiltStatus[16];}
247
248 // Return 1 if the 'Initialize failure' Pan/Tilter Status is true
249 // Return 0 otherwise. Remember to call Pan_TiltModelnq first!!!
250 inline int InitializeFailure(){return PanTiltStatus[17];}
251
252 // pan: contain the pan max speed of the camera
253 // tilt: contain the tilt max speed of the camera
254 // Return 1 on success, 0 on error

```

```

255 // An error is most Likely caused if the camera is not powered on (done by
      Power(EVILIB_ON))
256 int Pan_TiltMaxSpeedInq(int &pan, int &tilt);
257
258 // pan: contain the pan position of the camera
259 // tilt: contain the tilt position of the camera
260 // Return 1 on success, 0 on error
261 // An error is most Likely caused if the camera is not powered on (done by
      Power(EVILIB_ON))
262 int Pan_TiltPosInq(float &pan, float &tilt);
263
264 //-----
265 // The following functions exist for all the camera, but might be override
266 // for one of them
267
268 // Override for EVI-D70(P)
269 // zoom: contain the zoom position of the camera
270 // Return 1 on success, 0 on error
271 // An error is most Likely caused if the camera is not powered on (done by
      Power(EVILIB_ON))
272 virtual int ZoomPosInq(float &zoom);
273
274 //-----
275 // The following functions exist for all the camera, but the number of
      valid parameters
276 // may change for each of them
277
278 // Defined in EVI-G2x.D3x & EVI-D7x & EVI-D10x
279 virtual int Zoom(int type, int speed, float zoom, int waitC) = 0;
280 virtual int Focus(int type, int focus, int waitC) = 0;
281 virtual int Bright(int type, int cmd, int waitC) = 0;
282
283 // Defined in EVI-G2x & EVI-D3x & EVI-D7x & EVI-D10x
284 virtual int WB(int type, int waitC) = 0;
285 virtual int AE(int type, int waitC) = 0;
286
287 // Defined in EVI-G20(21) & EVI-D30(31) & EVI-D70(P) & EVI-D100(P)
288 virtual int ShutterPosInq(int &shutter) = 0;
289
290 //-----
291 // The following functions exist for more than one camera, but not for all
      of them
292
293 // Defined in EVI-D30(31) & EVI-D70(P) & EVI-D100(P)
294 virtual int Shutter(int type, int speed, int waitC){std::cout<<" Shutter
      not defined for "<<camera_type<<"\n";return 1;}
295
296 // The following functions exist only for EVI-D30(31) & EVI-D70(P) &
      EVI-D100(P), define in EVI-Dxxx
297 virtual int IrisPosInq(int &iris){std::cout<<" IrisPosInq not defined for
      "<<camera_type<<"\n";return 1;}
298 virtual int Iris(int type, int cmd, int waitC){std::cout<<" Iris not
      defined for "<<camera_type<<"\n";return 1;}
299 virtual int Gain(int type, int setting, int waitC){std::cout<<"Gain not
      defined for "<<camera_type<<"\n";return 1;}

```

```

300 virtual int IR_Receive(int type, int waitC){std::cout<<"IR_Receive not
      defined for "<<<camera_type<<"\n";return 1;}
301 virtual int IR_ReceiveReturn(int type, int
      waitC){std::cout<<"IR_ReceiveReturn not defined for
      "<<<camera_type<<"\n";return 1;}
302 virtual int Pan_TiltLimitSet(int type, int mode, float pan_pos, float
      tilt_pos, int waitC){std::cout<<"Pan_TiltLimitSet not defined for
      "<<<camera_type<<"\n";return 1;}
303 virtual int IR_ReceiveReturn(int
      maxWaitTime){std::cout<<"IR_ReceiveReturn not defined for
      "<<<camera_type<<"\n";return 1;}
304
305 // The following functions exist only for EVI-G20(21) & EVI-D70(P) &
      EVI-D100(P), defined in EVI-G2x & EVI-D7x.D10x
306 virtual int ExpCompModelnq(){std::cout<<"ExpCompModelnq not defined for
      this camera type\n"; return 1;}
307 virtual int ExpComp(int type, int cmd, int waitC){std::cout<<"ExpComp not
      defined for "<<<camera_type<<"\n";return 1;}
308
309 // The following functions exist only for EVI-D70(P) & EVI-D100(P), defined
      in EVI-D7x.D10x
310 virtual int AutoPowerOff(int timer, int waitC){std::cout<<"AutoPowerOff
      not defined for "<<<camera_type<<"\n";return 1;}
311 virtual int RGain(int type, int gain, int waitC){std::cout<<"RGain not
      defined for "<<<camera_type<<"\n";return 1;}
312 virtual int BGain(int type, int gain, int waitC){std::cout<<"BGain not
      defined for "<<<camera_type<<"\n";return 1;}
313 virtual int SlowShutter(int type, int waitC){std::cout<<"SlowShutter not
      defined for "<<<camera_type<<"\n";return 1;}
314 virtual int Aperture(int type, int cmd, int waitC){std::cout<<"Aperture
      not defined for "<<<camera_type<<"\n";return 1;}
315 virtual int LR_Reverse(int type, int waitC){std::cout<<"LR_Reverse not
      defined for "<<<camera_type<<"\n";return 1;}
316 virtual int Freeze(int type, int waitC){std::cout<<"Freeze not defined
      for "<<<camera_type<<"\n";return 1;}
317 virtual int AutoPowerOfflnq(int &timer){std::cout<<"AutoPowerOfflnq not
      defined for "<<<camera_type<<"\n";return 1;}
318 virtual int DZoomModelnq(){std::cout<<"DZoomModelnq not defined for
      "<<<camera_type<<"\n";return 1;}
319 virtual int RGainlnq(int &gain){std::cout<<"RGainlnq not defined for
      "<<<camera_type<<"\n";return 1;}
320 virtual int BGainlnq(int &gain){std::cout<<"BGainlnq not defined for
      "<<<camera_type<<"\n";return 1;}
321 virtual int SlowShutterModelnq(){std::cout<<"SlowShutterModelnq not
      defined for "<<<camera_type<<"\n";return 1;}
322 virtual int Aperturelnq(int &aperture){std::cout<<"Aperturelnq not
      defined for "<<<camera_type<<"\n";return 1;}
323 virtual int LR_ReverseModelnq(){std::cout<<"LR_ReverseModelnq not defined
      for "<<<camera_type<<"\n";return 1;}
324 virtual int FreezeModelnq(){std::cout<<"FreezeModelnq not defined for
      "<<<camera_type<<"\n";return 1;}
325
326 // The following functions exist only for EVI-D70(P) & EVI-D100(P), defined
      in EVI-D7x and EVI-D10x
327 virtual int PictureEffect(int type, int waitC){std::cout<<"PictureEffect
      not defined for "<<<camera_type<<"\n";return 1;}

```

```

328  virtual int AFModelInq() {std::cout<<"AFModelInq not defined for
      "<<camera_type<<"\n";return 1;}
329  virtual int FocusNearLimitInq(int &focus) {std::cout<<"FocusNearLimitInq
      not defined for "<<camera_type<<"\n";return 1;}
330  virtual int BrightPosInq(int &bright) {std::cout<<"BrightPosInq not
      defined for "<<camera_type<<"\n";return 1;}
331  virtual int ExpCompPosInq(int &ExpComp) {std::cout<<"ExpCompPosInq not
      defined for "<<camera_type<<"\n";return 1;}
332  virtual int PictureEffectModelInq() {std::cout<<"PictureEffectModelInq not
      defined for "<<camera_type<<"\n";return 1;}
333
334  // The following functions exist only for EVI-G20(21) & EVI-D30(31) &
      EVI-D70(P), defined in EVI-G2x_D3x & EVI-D7x
335  virtual int KeyLock(int type, int waitC) {std::cout<<"KeyLock not defined
      for "<<camera_type<<"\n";return 1;}
336  virtual int KeyLockInq() {std::cout<<"KeyLockInq not defined for
      "<<camera_type<<"\n";return 1;}
337  virtual int IDInq(int &id) {std::cout<<"IDInq not defined for
      "<<camera_type<<"\n";return 1;}
338
339  // The following functions exist only for EVI-D30(31) and EVI-D100(P),
      defined in EVI-D3x and EVI-D10x
340  virtual int Datascreen(int type, int waitC) {std::cout<<"Datascreen not
      defined for "<<camera_type<<"\n";return 1;}
341  virtual int VideoSystemInq() {std::cout<<"VideoSystemInq not defined for
      "<<camera_type<<"\n";return 1;}
342  virtual int DatascreenInq() {std::cout<<"DatascreenInq not defined for
      "<<camera_type<<"\n";return 1;}
343
344  // The following functions exist only for EVI-G20(21) & EVI-D70(P), defined
      in EVI-G2x and EVI-D7x
345  virtual int IDWrite(int id, int waitC) {std::cout<<"IDWrite not defined
      for "<<camera_type<<"\n";return 1;}
346
347  protected:
348  ...
349  private:
350  ...
351  };

```

A.2 EVI-Dxxx.h

```

1  class EVI_Dxxx : public virtual EVILibAAU
2  {
3  public:
4      EVI_Dxxx();
5      virtual ~EVI_Dxxx();
6
7      // iris: contain the iris position of the camera
8      // Return 1 on success, 0 on error
9      int IrisPosInq(int &iris);
10
11     // Iris Setting. Enable on AE_Manual or Iris_Priority
12     // type: EVILIB_RESET, EVILIB_UP, EVILIB_DOWN, EVILIB_DIRECT (DIRECT need
        'cmd')
13     // cmd: ExpComp Position from min_iris to max_iris

```

```

14 //      Check camera documentation for values
15 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
      EVILIB_WAIT_COMP
16 // Return 1 on success, 0 on error (Remember to call AE(EVILIB_MANUAL) or
      AE(EVILIB_SHUTTER_PRIO))
17 int Iris(int type, int cmd, int waitC);
18
19 // Gain Setting. Enable on AE_Manual only
20 // type: EVILIB_RESET, EVILIB_UP, EVILIB_DOWN, EVILIB_DIRECT (DIRECT need
      'setting ')
21 // setting: from EVILib_minGain to EVILib_maxGain
22 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
      EVILIB_WAIT_COMP
23 // Return 1 on success, 0 on error (Remember to call AE(EVILIB_MANUAL))
24 int Gain(int type, int setting, int waitC);
25
26 // Enable/Disable for IR remote commander
27 // type: EVILIB_ON, EVILIB_OFF, EVILIB_ON_OFF
28 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
      EVILIB_WAIT_COMP
29 // Return 1 on success, 0 on error
30 int IR_Receive(int type, int waitC);
31
32 // Send replies what command received from IR Commander
33 // type: EVILIB_ON, EVILIB_OFF
34 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
      EVILIB_WAIT_COMP
35 // Return 1 on success, 0 on error
36 int IR_ReceiveReturn(int type, int waitC);
37
38 // Pan/Tilt limit set
39 // type: EVILIB_SET —> need 'mode' & 'pan_pos' & 'tilt_pos'
40 //      EVILIB_CLEAR —> need 'mode' & 'pan_pos' & 'tilt_pos'
41 // mode: EVILIB_UPRIGHT, EVILIB_DOWNLEFT
42 // pan_pos: pan position: approx. -maxpan to +maxpan
43 // tilt_pos: tilt position: approx. -mintilt to +maxtilt
44 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
      EVILIB_WAIT_COMP
45 // Return 1 on success, 0 on error
46 int Pan_TiltLimitSet(int type, int mode, float pan_pos, float tilt_pos,
      int waitC);
47
48 // maxWaitTime: the maximum time to wait for data [sec]
49 // Return on success: EVILIB_POWER_ON_OFF, EVILIB_ZOOM_TELE_WIDE,
      EVILIB_AF_ON_OFF,
50 //      EVILIB_CAM_BACKLIGHT, EVILIB_CAM_MEMORY,
      EVILIB_PAN_TILT_DRIVE,
51 //      EVILIB_AT_MODE_ON_OFF, EVILIB_MD_MODE_ON_OFF
52 // Return on error: 0
53 int IR_ReceiveReturn(int maxWaitTime);
54
55 protected :
56
57 };

```

A.3 EVI-D7x_D10x.h

```
1 class EVI_D7x_D10x : virtual public EVI_Dxxx
2 {
3 public:
4     EVI_D7x_D10x();
5     virtual ~EVI_D7x_D10x();
6
7 // Auto Power Off
8 // timer = power off timer parameter 0000 (timer off) to 65535 (FFFF)
9 // minutes
10 // Initial value: 0000
11 // The power automatically turns off if the camera does not receive any
12 // VISCA commands
13 // or signals from the Remote Commander for the duration you set in the
14 // timer
15 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP
16 // EVILIB_WAIT_COMP
17 // Return 1 on success, 0 on error
18 int AutoPowerOff(int timer, int waitC);
19
20 // type: EVILIB_RESET, EVILIB_UP, EVILIB_DOWN, EVILIB_DIRECT
21 // gain: R Gain 0000 to 255 (FF), 256 steps
22 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
23 // EVILIB_WAIT_COMP
24 // Return 1 on success, 0 on error
25 int RGain(int type, int gain, int waitC);
26
27 // type: EVILIB_RESET, EVILIB_UP, EVILIB_DOWN, EVILIB_DIRECT
28 // gain: B Gain 0000 to 255 (FF), 256 steps
29 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
30 // EVILIB_WAIT_COMP
31 // Return 1 on success, 0 on error
32 int BGain(int type, int gain, int waitC);
33
34 // type: EVILIB_AUTO
35 // DERVERY_MANUAL
36 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
37 // EVILIB_WAIT_COMP
38 // Return 1 on success, 0 on error
39 int SlowShutter(int type, int waitC);
40
41 // type: EVILIB_RESET, EVILIB_UP, EVILIB_DOWN, EVILIB_DIRECT
42 // cmd: Aperture Gain 0 to 15 (F), 16 steps, Initial value: 5
43 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
44 // EVILIB_WAIT_COMP
45 // Return 1 on success, 0 on error
46 int Aperture(int type, int cmd, int waitC);
47
48 // Mirror image ON/OFF
49 // type: EVILIB_ON, EVILIB_OFF
50 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
51 // EVILIB_WAIT_COMP
52 // Return 1 on success, 0 on error
53 int LR.Reverse(int type, int waitC);
54
```

```

47 // Still image ON/OFF
48 // type: EVILIB_ON, EVILIB_OFF
49 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
50 // Return 1 on success, 0 on error
51 int Freeze(int type, int waitC);
52
53 // timer: contain the power off timer
54 // Return 1 on success, 0 on error
55 int AutoPowerOffInq(int &timer);
56
57 // Return on success: EVILIB_ON, EVILIB_OFF
58 // Return on error: 0
59 int DZoomModelnq();
60
61 // gain contain the R Gain
62 // Return 1 on success, 0 on error
63 int RGainInq(int &gain);
64
65 // gain contain the B Gain
66 // Return 1 on success, 0 on error
67 int BGainInq(int &gain);
68
69 // Return on success: EVILIB_AUTO, EVILIB_MANUAL
70 // Return 0 on error
71 int SlowShutterModelnq();
72
73 // Return on success: EVILIB_ON, EVILIB_OFF
74 // Return on error: 0
75 int ExpCompModelnq();
76
77 // aperture contain the Aperture Gain
78 // Return 1 on success, 0 on error
79 int ApertureInq(int &aperture);
80
81 // Return on success: EVILIB_ON, EVILIB_OFF
82 // Return on error: 0
83 int LR_ReverseModelnq();
84
85 // Return on success: EVILIB_ON, EVILIB_OFF
86 // Return on error: 0
87 int FreezeModelnq();
88
89 // type: EVILIB_ON, EVILIB_OFF, EVILIB_RESET, EVILIB_UP, EVILIB_DOWN,
    EVILIB_DIRECT (DIRECT need 'cmd')
90 // cmd: ExpComp Position 0 (-7, -10.5 dB) to 14 (0E, 7 10.5dB)
91 // Check camera documentation for values
92 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
93 // Return 1 on success, 0 on error
94 int ExpComp(int type, int cmd, int waitC);
95
96 protected:
97
98 };

```

A.4 EVI-G2x_D3x.h

```
1 class EVI_G2x_D3x : public virtual EVILibAAU
2 {
3 public:
4     EVI_G2x_D3x();
5     virtual ~EVI_G2x_D3x();
6
7 // When turning on to Bright Mode, Iris , Gain and Shutter at the time then
8 // increase or decrease
9 // 3 dB/step using UP/DOWN command
10 // type: EVILIB_RESET, EVILIB_UP, EVILIB_DOWN
11 // cmd: not used
12 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
13 // EVILIB_WAIT_COMP
14 // Return 1 on success, 0 on error (Remember to send AE(EVILIB_BRIGHT)
15 // before Bright(...))
16 int Bright(int type, int cmd, int waitC);
17
18 // Enable/Disable for RS-232C and key control
19 // type: EVILIB_ON, EVILIB_OFF
20 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
21 // EVILIB_WAIT_COMP
22 // Return 1 on success, 0 on error
23 int KeyLock(int type, int waitC);
24
25 // Return on success: EVILIB_ON, EVILIB_OFF
26 // Return on error: 0
27 int KeyLockInq();
28
29 // id: contain the ID of the camera
30 // Return 1 on success, 0 on error
31 int IDInq(int &id);
32
33 // Zoom control.
34 // type: EVILIB_STOP
35 // EVILIB_TELE (Standard)
36 // EVILIB_WIDE (Standard)
37 // EVILIB_TELE (Variable) —> need 'speed'
38 // EVILIB_WIDE (Variable) —> need 'speed'
39 // EVILIB_DIRECT —> need 'zoom'
40 // speed: speed parameter min_zspeed (Low) to max_zspeed (High)
41 // zoom: minzoom (Wide) to maxzoom (Tele)
42 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
43 // EVILIB_WAIT_COMP
44 // Return 1 on success, 0 on error
45 int Zoom(int type, int speed, float zoom, int waitC);
46
47 // Focus control.
48 // When adjust the focus, change the mode to Manual the send Far/Near or
49 // Direct command.
50 // type: EVILIB_STOP, EVILIB_FAR, EVILIB_NEAR, EVILIB_AUTO, EVILIB_MANUAL,
51 // EVILIB_AUTO_MANUAL, EVILIB_DIRECT (DIRECT need 'focus')
52 // focus: infinity = minFocus, close = maxFocus
53 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
54 // EVILIB_WAIT_COMP
```

```

48 // Return 1 on success, 0 on error (Remember to send Focus(EVILIB_MANUAL)
    before Far/Near or Direct)
49 int Focus(int type, int focus, int waitC);
50
51 protected:
52
53 };

```

A.5 EVI-G2x.h

```

1 class EVI_G2x : public virtual EVI_G2x_D3x
2 {
3 public:
4     EVI_G2x(); // don't forget to call Open()!
5     virtual ~EVI_G2x();
6
7 // White Balance Setting.
8 // type: EVILIB_AUTO           : Trace the light source automatically
9 //       EVILIB_INDOOR         : fixed at factory
10 //      EVILIB_OUTDOOR        : fixed at factory
11 //      EVILIB_4000K          : fixed at factory. For a fluorescent light.
12 //      EVILIB_5000K          : fixed at factory. For a fluorescent light.
13 //      EVILIB_ONEPUSH_MODE   : Pull-in to White with a Trigger then hold
    the data until next Trigger coming
14 //      EVILIB_ONEPUSH_TRIGGER
15 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP
16 //                                           EVILIB_WAIT_COMP
17 // Return 1 on success, 0 on error (Remember to send
    WB(EVILIB_ONEPUSH_MODE) before EVILIB_ONEPUSH_TRIGGER)
18 int WB(int type, int waitC);
19
20 // type: EVILIB_AUTO           : Auto Exposure Mode
21 //       EVILIB_BRIGHT         : Bright mode (Manual control)
22 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP
23 //                                           EVILIB_WAIT_COMP
24 // Return 1 on success, 0 on error
25 int AE(int type, int waitC);
26
27 // type: EVILIB_RESET, EVILIB_ON, EVILIB_OFF
28 //       EVILIB_UP           ( 3 dB/step)
29 //       EVILIB_DOWN        (-3 dB/step)
30 //       EVILIB_DIRECT      —> need 'cmd'
31 // cmd: ExpCompnst Position -7 (-21 dB) to 7 (21dB)
32 //       Check camera documentation for values
33 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
34 // Return 1 on success, 0 on error
35 int ExpComp(int type, int cmd, int waitC);
36
37 // type: EVILIB_ON, EVILIB_OFF, EVILIB_ON_OFF
38 // Return 1 on success, 0 on error
39 int FlkrCancel(int type, int waitC);
40
41 // id: 0 (00) to 255 (FF)
42 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP

```

```

43 // Return 1 on success, 0 on error
44   int IDWrite(int id, int waitC);
45
46 //version: contain the version of the camera
47 //Return 1 on success, 0 on error
48   int VersionInq(int &version);
49
50 //Return on success: EVILIB_ON, EVILIB_OFF
51 // Return on error: 0
52   int FlkrCancellInq();
53
54 // Return on success: EVILIB_ON, EVILIB_OFF
55 // Return on error: 0
56   int ExpCompModelInq();
57
58 // Defined in EVI-G20 & EVI-G21
59   virtual int ShutterPosInq(int &shutter) = 0;
60
61 protected:
62
63 };

```

A.6 EVI-G20.h

```

1  class EVI_G20 : virtual public EVI_G2x
2  {
3  public:
4   EVI_G20(); // don't forget to call Open()!
5   ~EVI_G20();
6
7  // shutter: contain the shutter position of the camera
8  // Return 1 on success, 0 on error
9   int ShutterPosInq(int &shutter);
10
11 private:
12
13 };

```

A.7 EVI-G21.h

```

1  class EVI_G21 : virtual public EVI_G2x
2  {
3  public:
4   EVI_G21(); // don't forget to call Open()!
5   ~EVI_G21();
6
7  // shutter: contain the shutter position of the camera
8  // Return 1 on success, 0 on error
9   int ShutterPosInq(int &shutter);
10
11 private:
12
13 };

```

A.8 EVI-D3x.h

```
1 class EVI_D3x : virtual public EVI_Dxxx, virtual public EVI_G2x_D3x
2 {
3 public:
4   EVI_D3x(); // don't forget to call Open()!
5   virtual ~EVI_D3x();
6
7   // Automatic Target Trace ability Compensation when a wide conversion lens
   // is installed
8   // setting: 0 (no conversion) to 7 (0.6 conversion)
9   // Return 1 on success, 0 on error
10  // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP
11  //                                           EVILIB_WAIT_COMP
12  int Wide_conLensSet(int setting, int waitC);
13
14  // Target Tracking Mode ON/OFF
15  // type: EVILIB_ON, EVILIB_OFF, EVILIB_ON_OFF
16  // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
   // EVILIB_WAIT_COMP
17  // Return 1 on success, 0 on error
18  int AT.Mode(int type, int waitC);
19
20  // Auto Exposure for the target
21  // type: EVILIB_ON, EVILIB_OFF, EVILIB_ON_OFF
22  // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
   // EVILIB_WAIT_COMP
23  // Return 1 on success, 0 on error
24  int AT.AE(int type, int waitC);
25
26  // Automatic Zooming for the target
27  // type: EVILIB_ON, EVILIB_OFF, EVILIB_ON_OFF
28  // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
   // EVILIB_WAIT_COMP
29  // Return 1 on success, 0 on error
30  int AT.AutoZoom(int type, int waitC);
31
32  // Sensing Frame Display ON/OFF
33  // type: EVILIB_ON, EVILIB_OFF, EVILIB_ON_OFF
34  // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
   // EVILIB_WAIT_COMP
35  // Return 1 on success, 0 on error
36  int AT.MD.Frame.Display(int type, int waitC);
37
38  // Shifting the Sensing Frame for AR
39  // For Shifting use Pan/Tilt Drive Command
40  // type: EVILIB_ON, EVILIB_OFF, EVILIB_ON_OFF
41  // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
   // EVILIB_WAIT_COMP
42  // Return 1 on success, 0 on error
43  int AT.Offset(int type, int waitC);
44
45  // Tracking or Detecting Start/Stop
46  // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
   // EVILIB_WAIT_COMP
47  // Return 1 on success, 0 on error
```

```

48  int AT_MD_StartStop(int waitC);
49
50 // Select a Tracking Mode
51 // type: EVILIB_CHASE1, EVILIB_CHASE2, EVILIB_CHASE3, EVILIB_CHASE123
52 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
   // EVILIB_WAIT_COMP
53 // Return 1 on success, 0 on error
54  int AT_Chase(int type, int waitC);
55
56 // Select target study mode for AT
57 // type: EVILIB_ENTRY1, EVILIB_ENTRY2, EVILIB_ENTRY3, EVILIB_ENTRY4
58 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
   // EVILIB_WAIT_COMP
59 // Return 1 on success, 0 on error
60  int AT_Entry(int type, int waitC);
61
62 // Motion Detector Mode ON/OFF
63 // type: EVILIB_ON, EVILIB_OFF, EVILIB_ON_OFF
64 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
   // EVILIB_WAIT_COMP
65 // Return 1 on success, 0 on error
66  int MD_Mode(int type, int waitC);
67
68 // Detecting Area Set (Size or Position)
69 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
   // EVILIB_WAIT_COMP
70 // Return 1 on success, 0 on error
71  int MD_Frame(int waitC);
72
73 // Select Detecting Frame (1 or 2 or 1 + 2)
74 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
   // EVILIB_WAIT_COMP
75 // Return 1 on success, 0 on error
76  int MD_Detect(int waitC);
77
78 // Reply a completion when the camera lost the target in AT mode.
79 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
   // EVILIB_WAIT_COMP
80 // Return 1 on success, 0 on error
81  int AT_LostInfo(int waitC);
82
83 // Reply a completion when the camera detected a motion of image in MD
   // mode.
84 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
   // EVILIB_WAIT_COMP
85 // Return 1 on success, 0 on error
86  int MD_LostInfo(int waitC);
87
88 // Set Detecting Condition
89 // type: EVILIB_Y_LEVEL —> need 'setting'
90 // EVILIB_HUE_LEVEL —> need 'setting'
91 // EVILIB_SIZE —> need 'setting'
92 // EVILIB_DISPLAYTIME —> need 'setting'
93 // EVILIB_REFRESH_MODE1, EVILIB_REFRESH_MODE2, EVILIB_REFRESH_MODE3
94 // EVILIB_REFRESH_TIME —> need 'setting'
95 // setting: 0 to 15

```

```

96 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
97 // Return 1 on success, 0 on error
98 int MD_Adjust(int type, int setting, int waitC);
99
100 // Target Condition Measure Mode for More Accurate Setting for Motion
    Detector
101 // type: EVILIB_ON, EVILIB_OFF, EVILIB_ON_OFF
102 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
103 // Return 1 on success, 0 on error
104 int Measure_Mode1(int type, int waitC);
105
106 // Target Condition Measure Mode for More Accurate Setting for Motion
    Detector
107 // type: EVILIB_ON, EVILIB_OFF, EVILIB_ON_OFF
108 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
109 // Return 1 on success, 0 on error
110 int Measure_Mode2(int type, int waitC);
111
112 // lens: contain the lens No.
113 // Return 1 on success, 0 on error
114 int Wide_ConLensInq(int &lens);
115
116 // Return on success: EVILIB_NORMAL, EVILIB_ATMODE, EVILIB.MDMODE
117 // Return on error: 0
118 int ATMD_Modelnq();
119
120 // After the completion of ATModelnq, you can check the status with the
    functions provided
121 // Return 1 on success, 0 on error
122 int ATModelnq();
123
124 // Return 1 if the 'AT frame chase' AT Status is true. Return 0 otherwise
125 inline int ATFrameChase(){return ATStatus[0];}
126
127 // Return 1 if the 'AT pan chase' AT Status is true. Return 0 otherwise
128 inline int ATPanChase(){return ATStatus[1];}
129
130 // Return 1 if the 'AT frame/pan chase' AT Status is true. Return 0
    otherwise
131 inline int ATFramePanChase(){return ATStatus[2];}
132
133 // Return 1 if the 'AT offset' AT Status is true. Return 0 otherwise
134 inline int ATOffset(){return ATStatus[3];}
135
136 // Return 1 if the 'AT AE on/off' AT Status is true. Return 0 otherwise
137 inline int ATAEOff(){return ATStatus[4];}
138
139 // Return 1 if the 'AT zoom on/off' AT Status is true. Return 0 otherwise
140 inline int ATZoomOnOff(){return ATStatus[5];}
141
142 // Return 1 if the 'AT frame display on/off' AT Status is true. Return 0
    otherwise
143 inline int ATFrameDisplayOnOff(){return ATStatus[6];}

```

```

144
145 // Return 1 if the 'AT setting' AT Status is true. Return 0 otherwise
146 inline int ATSetting() {return ATStatus[7];}
147
148 // Return 1 if the 'AT working' AT Status is true. Return 0 otherwise
149 inline int ATWorking() {return ATStatus[8];}
150
151 // Return 1 if the 'AT lost' AT Status is true. Return 0 otherwise
152 inline int ATLost() {return ATStatus[9];}
153
154 // Return 1 if the 'AT memorizing' AT Status is true. Return 0 otherwise
155 inline int ATMemorizing() {return ATStatus[10];}
156
157 // Return on success: EVILIB_ENTRY1, EVILIB_ENTRY2, EVILIB_ENTRY3,
    EVILIB_ENTRY4
158 // Return on error: 0
159 int AT_EntryInq();
160
161 // After the completion of MDModelnq, you can check the status with the
    functions provided
162 // Return 1 on success, 0 on error
163 int MDModelnq();
164
165 // Return 1 if the 'MD detection method' MD Status is true. Return 0
    otherwise
166 inline int MDDetectionMethod() {return MDStatus[0];}
167
168 // Return 1 if the 'MD settind' MD Status is true. Return 0 otherwise
169 inline int MDSetting() {return MDStatus[1];}
170
171 // Return 1 if the 'MD undetect' MD Status is true. Return 0 otherwise
172 inline int MDUndetect() {return MDStatus[2];}
173
174 // Return 1 if the 'MD detecting' MD Status is true. Return 0 otherwise
175 inline int MDDetecting() {return MDStatus[3];}
176
177 // Return 1 if the 'MD memorizing' MD Status is true. Return 0 otherwise
178 inline int MDMemorizing() {return MDStatus[4];}
179
180 // Return 1 if the 'MD frame 1' MD Status is true. Return 0 otherwise
181 inline int MDFrame1() {return MDStatus[5];}
182
183 // Return 1 if the 'MDFrame2' MD Status is true. Return 0 otherwise
184 inline int MDFrame2() {return MDStatus[6];}
185
186 // Return 1 if the 'MD frame 1 or 2' MD Status is true. Return 0 otherwise
187 inline int MDFrame1or2() {return MDStatus[7];}
188
189 // Return 1 if the 'MD frame display' MD Status is true. Return 0 otherwise
190 inline int MDFrameDisplay() {return MDStatus[8];}
191
192 // Dividing a screen by 48*30 pixels, Return the center position of the
    detecting Frame.
193 // x: 4 to 42
194 // y: 3 to 27
195 // status: EVILIB_SETTING, EVILIB_TRACKING, EVILIB_LOST

```

```

196 // Return 1 on success, 0 on error
197 int AT_ObjctPosInq(int &x, int &y, int &status);
198
199 // Dividing a scene by 48*30 pixels, Return the center position of the
    detecting Frame.
200 // x: 4 to 42
201 // y: 3 to 27
202 // status: EVILIB_SETTING, EVILIB_TRACKING, EVILIB_LOST
203 // Return 1 on success, 0 on error
204 int MD_ObjctPosInq(int &x, int &y, int &status);
205
206 // level: 0 to 15
207 // Return 1 on success, 0 on error
208 int MD_YLevelInq(int &level);
209
210 // level: 0 to 15
211 // Return 1 on success, 0 on error
212 int MD_HueLevelInq(int &level);
213
214 // size: 0 to 15
215 // Return 1 on success, 0 on error
216 int MD_SizeInq(int &size);
217
218 // dt: 0 to 15
219 // Return 1 on success, 0 on error
220 int MD_DispTimeInq(int &dt);
221
222 // Return on success: EVILIB_REFRESH_MODE1, EVILIB_REFRESH_MODE2,
    EVILIB_REFRESH_MODE3
223 // Return on error: 0
224 int MD_RefModelInq();
225
226 // rt: 0 to 15
227 // Return 1 on success, 0 on error
228 int MD_RefTimeInq(int &rt);
229
230 // White Balance Setting.
231 // type: EVILIB_AUTO           : Trace the light source automatically
232 //       EVILIB_INDOOR        : fixed at factory
233 //       EVILIB_OUTDOOR       : fixed at factory
234 //       EVILIB_ONEPUSH_MODE  : Pull-in to White with a Trigger then hold
    the data until next Trigger coming
235 //       EVILIB_ONEPUSH_TRIGGER
236 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
237 // Return 1 on success, 0 on error (Remember to send
    WB(EVILIB_ONEPUSH_MODE) before EVILIB_ONEPUSH_TRIGGER)
238 int WB(int type, int waitC);
239
240 // type: EVILIB_AUTO           : Auto Exposure Mode
241 //       EVILIB_MANUAL        : Manual control mode
242 //       EVILIB_SHUTTER_PRIO  : Shutter priority automatic exposure mode
243 //       EVILIB_IRIS_PRIO    : Iris priority automatic exposure mode
244 //       EVILIB_BRIGHT       : Bright mode (Manual control)
245 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP

```

```

246 // Return 1 on success, 0 on error
247   int AE(int type, int waitC);
248
249 // On screen Data Display ON/OFF
250 // type: EVILIB_ON, EVILIB_OFF, EVILIB_ON_OFF
251 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
252 // Return 1 on success, 0 on error
253   int Datascreen(int type, int waitC);
254
255 // Return on success: EVILIB_NTSC, EVILIB_PAL
256 // Return on error: 0
257   int VideoSystemInq();
258
259 // Return on success: EVILIB_ON, EVILIB_OFF
260 // Return on error: 0
261   int DatascreenInq();
262
263 //-----
264
265 // Defined in EVI-D30 & EVI-D31
266   virtual int Shutter(int type, int speed, int waitC) = 0;
267   virtual int ShutterPosInq(int &shutter) = 0;
268
269 protected:
270
271 private:
272 ...
273 };

```

A.9 EVI-D30.h

```

1 class EVI_D30 : virtual public EVI_D3x
2 {
3 public:
4   EVI_D30(); // don't forget to call Open()!
5   ~EVI_D30();
6
7 // Electronic Shutter Setting
8 // Enable on AE.Manual, Shutter_Priority
9 // type: EVILIB_RESET, EVILIB_UP, EVILIB_DOWN, EVILIB_DIRECT (DIRECT need
    'speed')
10 // speed: 1/60 to 1/10000 second
11 // Authorized values: 60 - 75 - 90 - 100 - 125 - 150 - 180 - 215 - 250 -
    300 -
12 //   350 - 425 - 500 - 600 - 725 - 850 - 1000 - 1250 - 1500 - 1750 - 2000 -
    2500 -
13 //   3000 - 3500 - 4000 - 6000 - 10000
14 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
15 // Return 1 on success, 0 on error
16   int Shutter(int type, int speed, int waitC);
17
18 // shutter: contain the shutter position of the camera
19 // Return 1 on success, 0 on error
20   int ShutterPosInq(int &shutter);

```

```

21
22 private :
23
24 };

```

A.10 EVI-D31.h

```

1 class EVI_D31 : virtual public EVI_D3x
2 {
3 public :
4   EVI_D31(); // don't forget to call Open()!
5   ~EVI_D31();
6
7   // Electronic Shutter Setting
8   // Enable on AE_Manual, Shutter_Priority
9   // type: EVILIB_RESET, EVILIB_UP, EVILIB_DOWN, EVILIB_DIRECT (DIRECT need
10  // 'speed')
11  // speed: 1/50 to 1/10000 second
12  // Authorized values: 50 – 75 – 90 – 100 – 120 – 150 – 180 – 215 – 250 –
13  // 300 –
14  // 350 – 425 – 500 – 600 – 725 – 850 – 1000 – 1250 – 1500 – 1750 – 2000 –
15  // 2500 –
16  // 3000 – 3500 – 4000 – 6000 – 10000
17  // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
18  // EVILIB_WAIT_COMP
19  // Return 1 on success, 0 on error
20  int Shutter(int type, int speed, int waitC);
21
22  // shutter: contain the shutter position of the camera
23  // Return 1 on success, 0 on error
24  int ShutterPosInq(int &shutter);

```

A.11 EVI-D7x.h

```

1 class EVI_D7x : virtual public EVI_D7x.D10x
2 {
3 public :
4   EVI_D7x(); // don't forget to call Open()! NOTE: remember to call
5   // DZoomCSModelnq() after power ON
6   virtual ~EVI_D7x();
7
8   //-----
9   // Night Power Off
10  // timer = power off timer parameter 0000 (timer off) to 65535 (FFFF)
11  // minutes
12  // A setting of 0 (zero min.) is equivalent to OFF, and the smallest value
13  // that
14  // can be set is 1 min.
15  // When the Day/Night function is in effect, abd the Night setting as been
16  // made,

```

```

14 // if an operation is not attempted via either a VISCA command or the
    remote
15 // controller, the unit will continue to operate for the time set in the
    timer,
16 // and will then shut off automatically.
17 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
18 // Return 1 on success, 0 on error
19 int NightPowerOff(int timer, int waitC);
20
21 // AF Sensitivity
22 // type: EVILIB_NORMAL, EVILIB_LOW
23 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
24 // Return 1 on success, 0 on error
25 int AF_Sensitivity(int type, int waitC);
26
27 // AF Mode
28 // type: EVILIB_NORMAL, EVILIB_INTERVAL, EVILIB_ZOOM_TRIGGER
29 //     EVILIB_ALTIME —> need 'movementTime' & 'interval'
30 // movementTime: minMovementTime to maxMovementTime
31 // interval: minInterval to maxInterval
32 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
33 // return 1 on success, 0 on error
34 int AFMode(int type, int movementTime, int interval, int waitC);
35
36 // Initialize
37 // type: EVILIB_LENS : Lens Initialization Start
38 //     EVILIB_COMP_SCAN : Correction of CCD pixel blemishes
39 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
40 // Return 1 on success, 0 on error
41 int Initialize(int type, int waitC);
42
43 // Spot Automatic Exposure setting
44 // type: EVILIB_ON, EVILIB_OFF, EVILIB_POSITION (POSITION need 'x' & 'y')
45 // x: 0 to 15 (F)
46 // y: 0 to 15 (F)
47 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
48 // Return 1 on success, 0 on error
49 int SpotAE(int type, int x, int y, int waitC);
50
51 // Infrared mode (some models do not support Infrared Mode)
52 // type: EVILIB_ON, EVILIB_OFF
53 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
54 // Return 1 on success, 0 on error
55 int ICR(int type, int waitC);
56
57 // Auto Infrared mode (some models do not support Infrared Mode)
58 // type: EVILIB_ON, EVILIB_OFF
59 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
60 // Return 1 on success, 0 on error

```

```

61  int AutoICR(int type , int waitC);
62
63 // Display
64 // type: EVILIB_ON, EVILIB_OFF, EVILIB_ON_OFF
65 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
66 // Return 1 on success, 0 on error
67  int Display(int type, int waitC);
68
69 // Title
70 // type: EVILIB_TITLE_SET1 -> need 'data[0]' Vposition, 'data[1]'
    Hposition, 'data[2]' Color, 'data[3]' Blink
71 //     EVILIB_TITLE_SET2 -> need 'data' contain 1st to 10th characters to
    display
72 //     EVILIB_TITLE_SET3 -> need 'data' contain 11st to 20th characters
    to display
73 //     EVILIB_TITLE_CLEAR, EVILIB_ON, EVILIB_OFF
74 // Check camera documentation for values
75 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
76 // Return 1 on success, 0 on error
77  int Title(int type, int data[10], int waitC);
78
79 // Mute
80 // type: EVILIB_ON, EVILIB_OFF, EVILIB_ON_OFF
81 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
82 // Return 1 on success, 0 on error
83  int Mute(int type, int waitC);
84
85 // Camera ID
86 // id: 0 (0000) to 65535 (FFFF)
87 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
88 // Return 1 on success, 0 on error
89  int IDWrite(int id, int waitC);
90
91 // Camera Alarm
92 // type: EVILIB_ON, EVILIB_OFF
93 //     EVILIB_SET_MODE -> data[0]: 0 to 13 (0C)
94 //     EVILIB_SET_DAY_NIGHT_LEVEL -> data[0]: 0 to 4095 Day setting AE
    level (no maximum value given in documentation)
95 //     data[1]: 0 to 4095 Night setting AE
    level (no maximum value given in documentation)
96 // also, this can be called SetTime. The guy who made the technical doc
    should be more carefull...
97 // Check camera documentation for values
98 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
99 // Return 1 on success, 0 on error
100 int Alarm(int type, int data[2], int waitC);
101
102 // timer contain the power off timer
103 // Return 1 on success, 0 on error
104 int NightPowerOffInq(int &timer);
105

```

```

106 // Return on success: EVILIB_COMBINE_MODE, EVILIB_SEPARATE_MODE
107 // Return on error: 0
108 // NOTE: This MUST be called for the EVI-D7x after power(ON) (only for
      D70(P)) FIXME: make note at constructor
109  int DZoomCSModelnq();
110
111 // Return on success: EVILIB_NORMAL, EVILIB_LOW
112 // Return on error: 0
113  int AF_SensitivityInq();
114
115 // movementTime contain the Movement Time
116 // interval contain the interval
117 // Return 1 on success, 0 on error
118  int AFTimeSettingnq(int &movementTime, int &interval);
119
120 // Return on success: EVILIB_ON, EVILIB_OFF
121 // Return on error: 0
122  int SpotAEModelnq();
123
124 // x contain the X position
125 // y contain the Y position
126 // Return 1 on success, 0 on error
127  int SpotAEPosnq(int x, int y);
128
129 // Return on success: EVILIB_ON, EVILIB_OFF
130 // Return on error: 0
131  int PictureFlipModelnq();
132
133 // Return on success: EVILIB_ON, EVILIB_OFF
134 // Return on error: 0
135  int ICRModelnq();
136
137 // Return on success: EVILIB_ON, EVILIB_OFF
138 // Return on error: 0
139  int AutoICRModelnq();
140
141 // Return on success: EVILIB_ON, EVILIB_OFF
142 // Return on error: 0
143  int DisplayModelnq();
144
145 // Return on success: EVILIB_ON, EVILIB_OFF
146 // Return on error: 0
147  int TitleDisplayModelnq();
148
149 // Return on success: EVILIB_ON, EVILIB_OFF
150 // Return on error: 0
151  int MuteModelnq();
152
153 // model contain the Model ID
154 // rom contain the ROM version
155 // socket contain the Socket numer (=2)
156 // Return 1 on success, 0 on error
157  int DeviceTypenq(int &model, int &rom, int &socket);
158
159 // Return on success: EVILIB_ON, EVILIB_OFF
160 // Return on error: 0

```

```

161  int AlarmInq();
162
163  // mode contain the alarm mode
164  // Return 1 on success, 0 on error
165  int AlarmModelnq(int &mode);
166
167  // day contain the Day setting AE level
168  // night contain the Night setting AE level
169  // level contain the current AE level
170  // Return 1 on success, 0 on error
171  int AlarmDayNightLevelInq(int &day, int &night, int &level);
172
173  // Return on success: EVILIB_HIGH, EVILIB_LOW
174  // Return on error: 0
175  int AlarmDetectLevelInq();
176
177  // Picture effect setting
178  // type: EVILIB_OFF, EVILIB_NEGART, EVILIB_BW
179  // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
180  //          EVILIB_WAIT_COMP
181  // Return 1 on success, 0 on error
182  int PictureEffect(int type, int waitC);
183
184  // Enable/Disable for RS-232C and key control
185  // type: EVILIB_ON, EVILIB_OFF
186  // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
187  //          EVILIB_WAIT_COMP
188  // Return 1 on success, 0 on error
189  int KeyLock(int type, int waitC);
190
191  // focus contain the Focus limit position
192  // Return 1 on success, 0 on error
193  int FocusNearLimitInq(int &focus);
194
195  // Return on success: EVILIB_AF_SENS_HIGH, EVILIB_AF_SENS_LOW
196  // Return 0 on error
197  int AFModelnq();
198
199  // bright contain the bright value of the camera
200  // Return 1 on success, 0 on error
201  int BrightPosInq(int &bright);
202
203  // ExpComp contain the ExpComp position
204  // Return 1 on success, 0 on error
205  int ExpCompPosInq(int &ExpComp);
206
207  // Return on success: EVILIB_OFF, EVILIB_NEGART, EVILIB_BW
208  // Return on error: 0
209  int PictureEffectModelnq();
210
211  // Return on success: EVILIB_ON, EVILIB_OFF
212  // Return on error: 0
213  int KeyLockInq();
214
215  // id: contain the ID of the camera

```

```

215 // Return 1 on success, 0 on error
216 int IDInq(int &id);
217
218 //-----
219
220 // Focus control.
221 // When adjust the focus, change the mode to Manual the send Far/Near or
    Direct command.
222 // type: EVILIB_STOP, EVILIB_FAR, EVILIB_NEAR
223 //     EVILIB_FAR_V → focus: speed parameter, min_FocusSpeed (low) to
    max_FocusSpeed (high)
224 //     EVILIB_NEAR_V → focus: speed parameter, min_FocusSpeed (low) to
    max_FocusSpeed (high)
225 //     EVILIB_DIRECT → focus: infinity = minFocus, close = maxFocus
226 //     EVILIB_AUTO, EVILIB_MANUAL, EVILIB_AUTO_MANUAL,
    EVILIB_ONEPUSH_TRIGGER, EVILIB_INFINITY
227 //     EVILIB_NEAR_LIMIT → focus: focus near limit position min_Focus
    (far) to max_Focus (near)
228 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
229 // Return 1 on success, 0 on error (Remember to send Focus(EVILIB_MANUAL)
    before Far/Near or Direct)
230 int Focus(int type, int focus, int waitC);
231
232 // White Balance Setting.
233 // type: EVILIB_AUTO           : Normal Auto
234 //     EVILIB_INDOOR           : Indoor mode
235 //     EVILIB_OUTDOOR          : Outdoor mode
236 //     EVILIB_ONEPUSH_MODE     : One Push WB mode
237 //     EVILIB_ATW              : Auto Tracing White Balance
238 //     EVILIB_MANUAL           : Manual Control mode
239 //     EVILIB_ONEPUSH_TRIGGER  : One Push WB Trigger
240 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
241 // Return 1 on success, 0 on error (Remember to send
    WB(EVILIB_ONEPUSH_MODE) before EVILIB_ONEPUSH_TRIGGER)
242 int WB(int type, int waitC);
243
244 // type: EVILIB_AUTO           : Auto Exposure Mode
245 //     EVILIB_MANUAL           : Manual control mode
246 //     EVILIB_SHUTTER_PRIO    : Shutter priority automatic exposure mode
247 //     EVILIB_IRIS_PRIO      : Iris priority automatic exposure mode
248 //     EVILIB_BRIGHT         : Bright mode (Manual control)
249 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
250 // Return 1 on success, 0 on error
251 int AE(int type, int waitC);
252
253 // When turning on to Bright Mode, Iris, Gain and Shutter at the time then
    increase or decrease
254 // 3 dB/step using UP/DOWN command
255 // type: EVILIB_RESET, EVILIB_UP, EVILIB_DOWN, EVILIB_DIRECT
256 // cmd: 0 (00, close, 0 dB) to 31 (1F, F1.4, 28 dB)
257 //     Check camera documentation for values
258 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP

```

```

259 // Return 1 on success, 0 on error (Remember to send AE(EVILIB_BRIGHT)
    before Bright(...))
260 int Bright(int type, int cmd, int waitC);
261
262 //-----
263
264 // Defined in EVI-D70 & EVI-D70P
265 virtual int Zoom(int type, int speed, float zoom, int waitC) = 0;
266 virtual int ZoomPosInq(float &zoom) = 0;
267 virtual int DZoom(int type, int speed, float zoom, int waitC) = 0;
268 virtual int ZoomFocus(float zoom, int focus, int waitC) = 0;
269 virtual int DZoomPosInq(float &zoom) = 0;
270 virtual int Shutter(int type, int speed, int waitC) = 0;
271
272 protected:
273 // Keep track of the combined/separated mode for the zoom, digital zoom
274 int combined;
275 // zoom parameter specifik for the D70-serie
276 float maxzoomS;
277 int maxZoomS;
278 float mindzoom;
279 float maxdzoom;
280 int maxDZoom;
281 int minMovementTime;
282 int maxMovementTime;
283 int minInterval;
284 int maxInterval;
285 };

```

A.12 EVI-D70.h

```

1 class EVI_D70 : virtual public EVI_D7x
2 {
3 public:
4     EVI_D70(); // don't forget to call Open()! NOTE: remember to call
        DZoomCSModelnq() after power ON
5     ~EVI_D70();
6
7 // Zoom control.
8 // type: EVILIB_STOP
9 //     EVILIB_TELE_S (Standard)
10 //     EVILIB_WIDE_S (Standard)
11 //     EVILIB_TELE_V (Variable) —> need 'speed'
12 //     EVILIB_WIDE_V (Variable) —> need 'speed'
13 //     EVILIB_DIRECT —> need 'zoom'
14 // speed: speed parameter min_zspeed (Low) to max_zspeed (High)
15 // zoom: minzoom (Wide) to maxzoom (Tele)
16 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
        EVILIB_WAIT_COMP
17 // Return 1 on success, 0 on error
18 int Zoom(int type, int speed, float zoom, int waitC);
19
20 // zoom: contain the zoom position of the camera
21 // Return 1 on success, 0 on error
22 int ZoomPosInq(float &zoom);
23

```

```

24 // Digital Zoom
25 // type: EVILIB_ON, EVILIB_OFF
26 //     EVILIB_COMBINE_MODE : Optical/Digital Zoom Combined
27 //     EVILIB_SEPARATE_MODE : Optical/Digital Zoom Separate
28 //     EVILIB_STOP
29 //     EVILIB_TELE_V (variable) —> need 'speed'
30 //     EVILIB_WIDE_V (variable) —> need 'speed'
31 //     EVILIB_x1_MAX      : x1/MAX Magnification Switchover
32 //     EVILIB_DIRECT —> need 'zoom' // you can NOT use DZoom direct in
        combined mode
33 // speed: speed parameter min_dzspeed to max_dzspeed
34 // zoom: mindzoom to maxdzoom
35 // the 'zoom' value as different range according to the Combined/Separated
        mode. Be careful !!
36 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
        EVILIB_WAIT_COMP
37 // Return 1 on success, 0 on error
38 int DZoom(int type, int speed, float zoom, int waitC);
39
40 // ZoomFocus Direct
41 // zoom: minzoom (Wide) to maxzoom (Tele)
42 // focus: infinity = minFocus, close = maxFocus
43 // the 'zoom' value as different range according to the Combined/Separated
        mode. Be careful !!
44 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
        EVILIB_WAIT_COMP
45 // Return 1 on success, 0 on error
46 int ZoomFocus(float zoom, int focus, int waitC);
47
48 // zoom contain the D-Zoom position
49 // Return 1 on success, 0 on error
50 int DZoomPosInq(float &zoom);
51
52 // Electronic Shutter Setting
53 // Enable on AE_Manual, Shutter_Priority
54 // type: EVILIB_RESET, EVILIB_UP, EVILIB_DOWN, EVILIB_DIRECT (DIRECT need
        'speed')
55 // speed: 1/1 to 1/10000 second
56 // Authorized values: 1 – 2 – 4 – 8 – 15 – 30 – 60 – 90 – 100 –
57 // 125 – 180 – 250 – 350 – 500 – 725 – 1000 – 1500 – 2000 –
58 // 3000 – 4000 – 6000 – 10000
59 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
        EVILIB_WAIT_COMP
60 // Return 1 on success, 0 on error
61 int Shutter(int type, int speed, int waitC);
62
63 // shutter: contain the shutter position of the camera
64 // Return 1 on success, 0 on error
65 int ShutterPosInq(int &shutter);
66
67 private :
68
69 };

```

A.13 EVI-D70P.h

```

1 class EVI_D70P : virtual public EVI_D7x
2 {
3 public:
4   EVI_D70P(); // don't forget to call Open()! NOTE: remember to call
      DZoomCSModelnq() after power ON
5   ~EVI_D70P();
6
7 // Zoom control.
8 // type: EVILIB_STOP
9 //     EVILIB_TELE_S (Standard)
10 //     EVILIB_WIDE_S (Standard)
11 //     EVILIB_TELE_V (Variable) —> need 'speed'
12 //     EVILIB_WIDE_V (Variable) —> need 'speed'
13 //     EVILIB_DIRECT —> need 'zoom'
14 // speed: speed parameter min_zspeed (Low) to max_zspeed (High)
15 // zoom: minzoom (Wide) to maxzoom (Tele)
16 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
      EVILIB_WAIT_COMP
17 // Return 1 on success, 0 on error
18 int Zoom(int type, int speed, float zoom, int waitC);
19
20 // zoom: contain the zoom position of the camera
21 // Return 1 on success, 0 on error
22 int ZoomPoslnq(float &zoom);
23
24 // Digital Zoom
25 // type: EVILIB_ON, EVILIB_OFF
26 //     EVILIB_COMBINE_MODE : Optical/Digital Zoom Combined
27 //     EVILIB_SEPARATE_MODE : Optical/Digital Zoom Separate
28 //     EVILIB_STOP
29 //     EVILIB_TELE_V (variable) —> need 'speed'
30 //     EVILIB_WIDE_V (variable) —> need 'speed'
31 //     EVILIB_x1_MAX      : x1/MAX Magnification Switchover
32 //     EVILIB_DIRECT —> need 'zoom'
33 // speed: speed parameter min_dzspeed to max_dzspeed
34 // zoom: _mindzoom to maxdzoom // you can NOT use DZoom direct in combined
      mode
35 // the 'zoom' value as different range according to the Combined/Separated
      mode. Be carefull !!
36 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
      EVILIB_WAIT_COMP
37 // Return 1 on success, 0 on error
38 int DZoom(int type, int speed, float zoom, int waitC);
39
40 // ZoomFocus Direct
41 // zoom: minzoom (Wide) to maxzoom (Tele)
42 // focus: infinity = minFocus, close = maxFocus
43 // the 'zoom' value as different range according to the Combined/Separated
      mode. Be carefull !!
44 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
      EVILIB_WAIT_COMP
45 // Return 1 on success, 0 on error
46 int ZoomFocus(float zoom, int focus, int waitC);
47
48 // zoom contain the D-Zoom position
49 // Return 1 on success, 0 on error

```

```

50  int DZoomPosInq(float &zoom);
51
52 // Electronic Shutter Setting
53 // Enable on AE.Manual, Shutter_Priority
54 // type: EVILIB.RESET, EVILIB.UP, EVILIB.DOWN, EVILIB.DIRECT (DIRECT need
    'speed')
55 // speed: 1/1 to 1/10000 second
56 // Authorized values: 1 - 2 - 3 - 6 - 12 - 25 -50 -75 - 100 -
57 // 120 - 150 - 215 - 300 - 425 - 600 - 1000 - 1250 - 1750 -
58 // 2500 - 3500 - 6000 - 10000
59 // waitC: wait for completion of command: EVILIB.NO_WAIT_COMP,
    EVILIB.WAIT_COMP
60 // Return 1 on success, 0 on error
61  int Shutter(int type, int speed, int waitC);
62
63 // shutter: contain the shutter position of the camera
64 // Return 1 on success, 0 on error
65  int ShutterPosInq(int &shutter);
66
67 private:
68
69 };

```

A.14 EVI-D10x.h

```

1  class EVI_D10x : virtual public EVI_Dxxx
2  {
3  public:
4  EVI_D10x(); // don't forget to call Open()!
5  virtual ~EVI_D10x();
6
7  // Wide mode setting
8  // type: EVILIB.OFF, EVILIB.CINEMA, EVILIB_16_9_FULL
9  // waitC: wait for completion of command: EVILIB.NO_WAIT_COMP,
    EVILIB.WAIT_COMP
10 // Return 1 on success, 0 on error
11  int Wide(int type, int waitC);
12
13 // Digital effect setting
14 // type: EVILIB.OFF, EVILIB.STILL, EVILIB.FLASH, EVILIB.LUMI, EVILIB.TRAIL,
    EVILIB.EFFECTLEVEL (EFFECTLEVEL need 'effect')
15 // effect: effect level 0 (00) to 24 (18) (flash, trail), 0 (00) to 32
    (20) (still, lumi)
16 // waitC: wait for completion of command: EVILIB.NO_WAIT_COMP,
    EVILIB.WAIT_COMP
17 // Return 1 on success, 0 on error
18  int DigitalEffect(int type, int level, int waitC);
19
20 // Return on success: EVILIB.OFF, EVILIB.STILL, EVILIB.FLASH, EVILIB.LUMI,
    EVILIB.TRAIL
21 // Return on error: 0
22  int DigitalEffectModelnq();
23
24 // level contain the Effect level
25 // Return 1 on success, 0 on error
26  int DigitalEffectLevelInq(int &level);

```

```

27
28 // Return on success: EVILIB_OFF, EVILIB_CINEMA, EVILIB_16_9_FULL
29 // Return on error: 0
30 int WideModelInq();
31
32 // vender contain the Vender ID (1: Sony)
33 // model contain the Model ID
34 // rom contain the ROM version
35 // socket contain the Socket numer (=2)
36 // Return 1 on success, 0 on error
37 int DeviceTypeInq(int &vender, int &model, int &rom, int &socket);
38
39 //-----
40
41 // Zoom control.
42 // type: EVILIB_STOP
43 //     EVILIB_TELE_S (Standard)
44 //     EVILIB_WIDE_S (Standard)
45 //     EVILIB_TELE_V (Variable) —> need 'speed'
46 //     EVILIB_WIDE_V (Variable) —> need 'speed'
47 //     EVILIB_DIRECT —> need 'zoom'
48 //     EVILIB_DZOOM_ON
49 //     EVILIB_DZOOM_OFF
50 // speed: speed parameter min_zspeed (Low) to max_zspeed (High)
51 // zoom: minzoom (Wide) to maxzoom (Tele)
52 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
53 //     EVILIB_WAIT_COMP
54 // Return 1 on success, 0 on error
55 int Zoom(int type, int speed, float zoom, int waitC);
56
57 // Focus control.
58 // When adjust the focus, change the mode to Manual the send Far/Near or
59 // Direct command.
60 // type: EVILIB_STOP
61 //     EVILIB_FAR
62 //     EVILIB_NEAR
63 //     EVILIB_FAR_V -> focus: speed parameter, min_FocusSpeed (low) to
64 //     max_FocusSpeed (high)
65 //     EVILIB_NEAR_V -> focus: speed parameter, min_FocusSpeed (low) to
66 //     max_FocusSpeed (high)
67 //     EVILIB_DIRECT -> focus: infinity = minFocus, close = maxFocus
68 //     EVILIB_AUTO
69 //     EVILIB_MANUAL
70 //     EVILIB_AUTO_MANUAL
71 //     EVILIB_ONEPUSH_TRIGGER
72 //     EVILIB_INFINITY
73 //     EVILIB_AF_SENS_HIGH
74 //     EVILIB_AF_SENS_LOW
75 //     EVILIB_NEAR_LIMIT -> focus: focus near limit position min_Focus
76 //     (far) to max_Focus (near)
77 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
78 //     EVILIB_WAIT_COMP
79 // Return 1 on success, 0 on error (Remember to send Focus(EVILIB_MANUAL)
80 // before Far/Near or Direct)
81 int Focus(int type, int focus, int waitC);
82

```

```

76 // White Balance Setting.
77 // type: EVILIB_AUTO           : Trace the light source automatically
78 //     EVILIB_INDOOR          : fixed at factory
79 //     EVILIB_OUTDOOR         : fixed at factory
80 //     EVILIB_ONEPUSH.MODE    : Pull-in to White with a Trigger then hold
    the data until next Trigger coming
81 //     EVILIB_ATW             : Auto tracing white balance
82 //     EVILIB_MANUAL          : Manual control mode
83 //     EVILIB_ONEPUSH.TRIGGER
84 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
85 // Return 1 on success, 0 on error (Remember to send
    WB(EVILIB_ONEPUSH.MODE) before EVILIB_ONEPUSH.TRIGGER)
86 int WB(int type, int waitC);
87
88 // type: EVILIB_AUTO           : Auto Exposure Mode
89 //     EVILIB_MANUAL          : Manual control mode
90 //     EVILIB_SHUTTER_PRIO    : Shutter priority automatic exposure mode
91 //     EVILIB_IRIS_PRIO      : Iris priority automatic exposure mode
92 //     EVILIB_GAIN_PRIO      : Gain priority automatic Exposure Mode
93 //     EVILIB_BRIGHT         : Bright mode (Manual control)
94 //     EVILIB_SHUTTER_AUTO    : Automatic shutter mode
95 //     EVILIB_IRIS_AUTO      : Automatic iris mode
96 //     EVILIB_GAIN_AUTO      : Automatic gain mode
97 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
98 // Return 1 on success, 0 on error
99 int AE(int type, int waitC);
100
101 // When turning on to Bright Mode, Iris, Gain and Shutter at the time then
    increase or decrease
102 // 3 dB/step using UP/DOWN command
103 // type: EVILIB_RESET, EVILIB_UP, EVILIB_DOWN, EVILIB_DIRECT
104 // cmd: 0 (00, close, 0 dB) to 23 (17, F1.8, 18 dB)
105 //     Check camera documentation for values
106 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
107 // Return 1 on success, 0 on error (Remember to send AE(EVILIB_BRIGHT)
    before Bright(...) )
108 int Bright(int type, int cmd, int waitC);
109
110 // On screen Data Display ON/OFF
111 // type: EVILIB_ON, EVILIB_OFF, EVILIB_ON_OFF
112 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
    EVILIB_WAIT_COMP
113 // Return 1 on success, 0 on error
114 int Datascreen(int type, int waitC);
115
116 // Return on success: EVILIB_NTSC, EVILIB_PAL
117 // Return on error: 0
118 int VideoSystemInq();
119
120 // Return on success: EVILIB_ON, EVILIB_OFF
121 // Return on error: 0
122 int DatascreenInq();
123

```

```

124 // Picture effect setting
125 // type: EVILIB_OFF, EVILIB_PASTEL, EVILIB_NEGART, EVILIB_SEPIA, EVILIB_BW,
126 //       EVILIB_SOLARIZE, EVILIB_MOSAIC, EVILIB_SLIM, EVILIB_STRETCH
127 // waitC: wait for completion of command: EVILIB_NO_WAIT_COMP,
128 //       EVILIB_WAIT_COMP
129 // Return 1 on success, 0 on error
130 int PictureEffect(int type, int waitC);
131 // Return on success: EVILIB_AF_SENS_HIGH, EVILIB_AF_SENS_LOW
132 // Return 0 on error
133 int AFModelnq();
134
135 // focus contain the Focus limit position
136 // Return 1 on success, 0 on error
137 int FocusNearLimitnq(int &focus);
138
139 // bright contain the bright value of the camera
140 // Return 1 on success, 0 on error
141 int BrightPosnq(int &bright);
142
143 // ExpComp contain the ExpComp position
144 // Return 1 on success, 0 on error
145 int ExpCompPosnq(int &ExpComp);
146
147 // Return on success: EVILIB_OFF, EVILIB_PASTEL, EVILIB_NEGART,
148 //       EVILIB_SEPIA, EVILIB_BW
149 //       EVILIB_SOLARIZE, EVILIB_MOSAIC, EVILIB_SLIM,
150 //       EVILIB_STRETCH
151 // Return on error: 0
152 int PictureEffectModelnq();
153
154 // -----
155 // Defined in EVI-D100 & EVI-D100P
156 virtual int Shutter(int type, int speed, int waitC) = 0;
157 virtual int ShutterPosnq(int &shutter) = 0;
158
159 protected:
160 };

```

A.15 EVI-D100.h

```

1 class EVI_D100 : virtual public EVI_D10x
2 {
3 public:
4     EVI_D100(); // don't forget to call Open()!
5     ~EVI_D100();
6
7 // Electronic Shutter Setting
8 // Enable on AE_Manual, Shutter_Priority
9 // type: EVILIB_RESET, EVILIB_UP, EVILIB_DOWN, EVILIB_DIRECT (need 'speed')
10 // speed: 1/4 to 1/10000 second
11 // Authorized values: 4 - 8 - 15 - 30 - 60 - 90 - 100 - 125 - 180 - 250 -
12 //                   350 - 500 -
13 //                   725 - 1000 - 1500 - 2000 - 3000 - 4000 - 6000 - 10000

```

```

13 // waitC: wait for completion of command: EVILIB.NO_WAIT_COMP,
    EVILIB.WAIT_COMP
14 // Return 1 on success, 0 on error
15   int Shutter(int type, int speed, int waitC);
16
17 // shutter: contain the shutter position of the camera
18 // Return 1 on success, 0 on error
19   int ShutterPosInq(int &shutter);
20
21 private:
22
23 };

```

A.16 EVI-D100P.h

```

1 class EVI_D100P : virtual public EVI_D10x
2 {
3 public:
4   EVI_D100P(); // don't forget to call Open()!
5   ~EVI_D100P();
6
7 // Electronic Shutter Setting
8 // Enable on AE.Manual, Shutter_Priority
9 // type: EVILIB.RESET, EVILIB.UP, EVILIB.DOWN, EVILIB.DIRECT (DIRECT need
    'speed')
10 // speed: 1/4 to 1/10000 second
11 // Authorized values: 3 – 6 – 12 – 25 – 50 – 75 – 100 – 120 – 150 – 215 –
    300 – 425 –
12 // 600 – 1000 – 1250 – 1750 – 2500 – 3500 – 6000 – 10000
13 // waitC: wait for completion of command: EVILIB.NO_WAIT_COMP,
    EVILIB.WAIT_COMP
14 // Return 1 on success, 0 on error
15   int Shutter(int type, int speed, int waitC);
16
17 // shutter: contain the shutter position of the camera
18 // Return 1 on success, 0 on error
19   int ShutterPosInq(int &shutter);
20
21 private:
22
23 };

```