CU17 Project

Structure of the CU17 project

CU17 project contains modules for interface and command creating functionality, used for access to a hardware device via USB. The interface to the device is realized using the **CU80USB1.dll** library. A wrapper project was created (**Cu17Wrap.dll)**, which provides C interface to CU80USB1.dll, using only basic types (suitable for access from other applications like LabView, etc.).
An example Win32 application was prepared, which is illustrating the basic functions. A Delphi example is available also.

CU80USB1.DLL

The **CU80USB1.DLL** is an interface dynamic linked library, which contains the functions, performing all the necessary steps for work with the hardware device, connected to an USB port of the computer: creating and closing a connection to it, performing of step or continuous movements in all directions, etc. This is a module, developed in Delphi. The CU80USB1.Dll exports following functions:

CU80Open ()

Definition:

C / C++

```
_stdcall void CU80Open (R256 * res, DeviceRec * pdevRec );
```

Delphi

```
function  CU80Open (var pdevRec:DeviceRec):PChar256;  stdcall
```

The function opens a connection to the hardware through the selected USB port. All the settings for the connection are collected in DeviceRec structure, which has the following form:

C / C++

```
typedef struct delphiDevRec
{
  DWORD USBInstance;
  DWORD USBVersion;
  DWORD DevID;
  DWORD EEID;
} DeviceRec;
```

Delphi

```
DeviceRec = record USBInstance,USBVersion,DevID,EEID:dword;
end;
```

The properly initialized DeviceRec structure is later used in all other functions, exported from CU80USB1.dll to determine the connection to be used.
USBInstance allowed values (0..15)
EEID allowed values (0..15)

Parameters:

pDevRec – a pointer to a DeviceRec structure, which contains the settings.

Returns:

A string, representing the error if an error occurs or an empty string if the connection was opened successfully.

Example:

C++

```
DeviceRec ddr;
ddr.DevID = 2;
ddr.EEID = 0;
ddr.USBInstance = 0;
ddr.USBVersion = 1;
R256 tempR;
CU80Open(& tempR, &ddr);
if((tempR.A!=NULL)&&( tempR.A!='\0'))
{
   MessageBox(hDlg, tempR.A,"USB Error",MB_OK);
}
```

Delphi

```
USB01: DeviceRec;
USB01.USBInstance:=0;
USB01.USBVersion:=1;
USB01.DevID:=2;
USB01.EEID:=0;

sc:= CU80Open(USB01);
s:=string(sc);

if s<>'' then
begin
  if Application.MessageBox(PChar(s),'USB Error',MB_OK) = IDOK
then halt;
end;
```

CU80Close ()

Definition:

C / C++

```
void _stdcall CU80Close (DeviceRec devRec );
```

Delphi

```
procedure Cu80Close (devRec:DeviceRec);  stdcall
```

The function closes a connection to the hardware through the selected USB port, opened with previous call of CU80Open() function. All the settings for the connection are collected in DeviceRec structure, initialized during CU80Open() call.

Parameters:

devRec – DeviceRec structure, which contains the settings of the opened connection.

Returns:

Example:

C++

```
DeviceRec ddr;
CU80Close(ddr);
```

Delphi

```
USB01: DeviceRec;
CU80Close(USB01);
```

CU80PiezoStop ()

Definition:

C / C++

```
void _stdcall CU80PiezoStop(DeviceRec devRec );
```

Delphi

```
procedure CU80PiezoStop (devRec:DeviceRec);  stdcall
```

The function instantly terminates any movement of the selected hardware. All the settings for the connection are collected in DeviceRec structure, initialized during CU80Open() call.

Parameters:

devRec – DeviceRec structure, which contains the settings of the opened connection.

Returns:

Example:

C++

```
DeviceRec ddr;
CU80PiezoStop CU80PiezoStop(ddr);
```

Delphi

```
USB01: DeviceRec;
CU80PiezoStop (USB01);
```

CU80DCDCon ()

Definition:

C / C++

```
void _stdcall CU80DCDCon (DeviceRec devRec );
```

Delphi

```
procedure CU80DCDCon (devRec:DeviceRec);  stdcall
```

The function switches DCDC-converter on for the selected hardware. All the settings for the connection are collected in DeviceRec structure, initialized during CU80Open() call.

Parameters:

devRec – DeviceRec structure, which contains the settings of the opened connection.

Returns:

Example:

C++

```
DeviceRec ddr;
CU80DCDCon (ddr);
```

Delphi

```
USB01: DeviceRec;
CU80DCDCon(USB01);
```

CU80DCDCoff ()

Definition:

C / C++

```
void _stdcall CU80DCDCoff (DeviceRec devRec );
```

Delphi

```
procedure CU80DCDCoff (devRec:DeviceRec);  stdcall
```

The function switches DCDC-converter off for the selected hardware. All the settings for the connection are collected in DeviceRec structure, initialized during CU80Open() call.

Parameters:

devRec – DeviceRec structure, which contains the settings of the opened connection.

Returns:

Example:

C++

```
DeviceRec ddr;
CU80DCDCoff(ddr);
```

Delphi

```
USB01: DeviceRec;
CU80DCDCoff(USB01);
```

Echo()

Definition:

C / C++

```
DWORD _stdcall Echo(DeviceRec ddr, DWORD w);
```

Delphi

```
function  Echo(MUSBDeviceID:DeviceRec;w: dword):dword; stdcall
```

The function sends a number to the selected connection and performs listening on the connection. All the settings for the connection are collected in DeviceRec structure, initialized during CU80Open() call.

Parameters:

ddr – DeviceRec structure, which contains the settings of the opened connection.
W – double word value

Returns:

Arbitrary returns the delivery dword

Example:

C++

```
DeviceRec ddr;
DWORD x = Echo(ddr, 0);
```

Delphi

```
USB01: DeviceRec;
X := Echo(USB01, 0);
```

CU80PiezoMove()

Definition:

C / C++

```
void _stdcall CU80PiezoMove (DeviceRec ddr, INT32 Axis,
INT32 Vel, INT32 Volt);
```

Delphi

```
procedure CU80PiezoMove( MUSBDeviceID:DeviceRec;Axis,Vel,Volt:
longint);  stdcall
```

The function performs movement along one of the axes. The movement could be stopped using subsequental call of CU80PiezoStop (). All the settings for the connection are collected in DeviceRec structure, initialized during CU80Open() call.

Parameters:

ddr – DeviceRec structure, which contains the settings of the opened connection.
Axis – Determines the axis of the movement (Axis = 1 – X Axis, Axis = 2 – Y Axis, Axis = 3 – Z Axis)
Vel – Velocity in X-, Y-, Z- direction (-63 .. 63) (0 = stop) (±1 = single step)
Volt – Voltage in X-, Y-, Z- direction (22 …82)

Returns:

Example:

C++

```
DeviceRec ddr;
CU80PiezoMove(ddr,1,-63,40);
```

Delphi

```
USB01: DeviceRec;
CU80PiezoMove (USB01,1,-63,40);
```

GetUSBEEPromInfo ()

Definition:

C / C++

```
void _stdcall  GetUSBEEPromInfo(USBEEProm * res, DeviceRec
devRec );
```

Delphi

```
function  GetUSBEEPromInfo(MUSBDeviceID:DeviceRec): USBEEProm;
stdcall
```

The function retrieves information from USB board, connected to the current connection. All the information, returned from this function is collected in USBEEProm structure, which has the following form:

C / C++

```
typedef struct delphiIEE
{
  DWORD USBVendorID;
  DWORD USBProductID;
  DWORD USBDeviceID;
  DWORD DeviceID;
  DWORD EEPromID;
  DWORD Version;
  DWORD SerialNumber;
  DWORD CustomerID;
  R16   Company;
  R16   Date;
  R32   ProductStr;
  R32   Customer;
  R32   CustomerStr;
} USBEEProm;
```

Delphi

```
USBEEProm = record
        USBVendorID:   dword;
        USBProductID:  dword;
        USBDeviceID:   dword;
        DeviceID:      dword;
        EEPromID:      dword;
        Version:       dword;
        SerialNumber:  dword;
        CustomerID:    dword;
        Company:       PChar16;
        Date:          PChar16;
        ProductStr:    PChar32;
        Customer:      PChar32;
        CustomerStr:   PChar32;
     end;
```

C / C++

```
typedef struct r16
{
  char A[32];
} R16;
```

C / C++

```
typedef struct r32
{
  char A[32];
} R32;
```

Parameters:

DevRec –DeviceRec structure, which contains the settings.

Returns:

A USBEEProm structure, containing the data from the board.

Example:

C++

```
DeviceRec ddr;
USBEEProm usbEEprom;
GetUSBEEPromInfo(&usbEEprom, ddr);
```

Delphi

```
USB01: DeviceRec;
EP:=GetUSBEEPromInfo(USB01);
```

Cu17Wrap.dll Wrapper library

The **Cu17Wrap.dll** is an interface dynamic linked library, which encapsulates the interface to **CU80USB1.dll**. It exports functions, which works only with basic types (int, DWORD, double, etc, i.e. without using of any structures or records. The exportable functions are defined as __stdcall, so they can be accessed from C source as well. The Cu17Wrap.DLL is suitable when using of structures, needed in interface to CU80USB1.dll is not possible or there exist big difficulties to use such structures (when calling from different program language or third-party product with different structure member alignment etc.). The wrapper dll contains the functions, performing all the

necessary steps for work with the hardware device, connected to an USB port of the computer: creating and closing connection to it, performing of step or continuous movements in all directions, etc. The Cu17Wrap.dll exports following functions:

CU80WOpen ()

Definition:

C / C++

```
extern "C" CU80WRAP_API char * __stdcall CU80WOpen(DWORD*
USBInstance, DWORD* USBVersion, DWORD* DevID, DWORD* EEID);
```

The function opens a connection to the hardware through the selected USB port. All the settings for the connection are collected in DeviceRec structure, which has the following form:

C / C++

```
typedef struct delphiDevRec
{
  DWORD USBInstance;
  DWORD USBVersion;
  DWORD DevID;
  DWORD EEID;
} DeviceRec;
```

Parameters:

USBInstance, USBVersion, DevID, EEID – represent the corresponding fields of a DeviceRec structure, which contains the settings.

Returns:

A string, representing the error if an error occurs or an empty string if the connection was opened successfully.

CU80WRAP_API : `__declspec(dllexport) or __declspec(dllimport)`

Example:

C / C++

```
USBInstance = 0;
USBVersion = 1;
DevID = 2;
EEID = 0;
char * tempR = CU80WOpen (&USBInstance, &USBVersion, &DevID, &
EEID);
if((tempR!=NULL)&&( tempR[0]!='\0'))
{
   MessageBox(hDlg, tempR,"USB Error",MB_OK);
}
```

CU80WClose ()

Definition:

C / C++

```
extern "C" CU80WRAP_API void __stdcall CU80WClose(DWORD
USBInstance, DWORD USBVersion, DWORD DevID, DWORD EEID);
```

The function closes a connection to the hardware through the selected USB port, opened with previous call of CU80WOpen() function. All the settings for the connection are collected in DeviceRec structure members,

initialized during CU80WOpen() call.

Parameters:

USBInstance, USBVersion, DevID, EEID – represent the corresponding fields of a DeviceRec structure, which contains the settings of the opened connection.

Returns:

Example:

C / C++

```
CU80WClose (USBInstance, USBVersion, DevID, EEID);
```

CU80WPiezoStop ()

Definition:

C / C++

```
extern "C" CU80WRAP_API void __stdcall CU80WPiezoStop(DWORD
USBInstance, DWORD USBVersion, DWORD DevID, DWORD EEID);
```

The function instantly terminates any movement of the selected hardware. All the settings for the connection are collected in DeviceRec structure members, initialized during CU80WOpen() call.

Parameters:

USBInstance, USBVersion, DevID, EEID – represent the corresponding fields of a DeviceRec structure, which contains the settings of the opened connection.

Returns:

Example:

C / C++

```
CU80WPiezoStop (USBInstance, USBVersion, DevID, EEID);
```

CU80WDCDCon ()

Definition:

C / C++

```
extern "C" CU80WRAP_API void __stdcall CU80WDCDCon(DWORD
USBInstance, DWORD USBVersion, DWORD DevID, DWORD EEID);
```

The function switches DCDC-converter on for the selected hardware. All the settings for the connection are collected in DeviceRec structure members, initialized during CU80WOpen() call.

Parameters:

USBInstance, USBVersion, DevID, EEID – represent the corresponding fields of a DeviceRec structure, which contains the settings of the opened connection.

Returns:

Example:

C / C++

```
CU80WDCDCon (USBInstance, USBVersion, DevID, EEID);
```

CU80WDCDCoff ()

Definition:

C / C++

```
extern "C" CU80WRAP_API void __stdcall CU80WDCDCoff (DWORD
USBInstance, DWORD USBVersion, DWORD DevID, DWORD EEID);
```

The function switches DCDC-converter off for the selected hardware. All the settings for the connection are collected in DeviceRec structure, initialized during CU80WOpen () call.

Parameters:

USBInstance, USBVersion, DevID, EEID – represent the corresponding fields of a DeviceRec structure, which contains the settings of the opened connection.

Returns:

Example:

C / C++

```
CU80WDCDCoff (USBInstance, USBVersion, DevID, EEID);
```

CU80WEcho ()

Definition:

C / C++

```
extern "C" CU80WRAP_API DWORD __stdcall CU80WEcho(DWORD
USBInstance, DWORD USBVersion, DWORD DevID, DWORD EEID, DWORD
w);
```

The function sends a number to the selected connection and performs listening on the connection. All the settings for the connection are collected in DeviceRec structure members, initialized during CU80WOpen() call.

Parameters:

USBInstance, USBVersion, DevID, EEID – represent the corresponding fields of a DeviceRec structure, which contains the settings of the opened connection.
w - DWORD value.

Returns:

The DWORD value, received as result from CU80WEcho() function.

Example:

C++

```
Servo3WEcho(USBInstance, USBVersion, DevID, EEID, 0);
```

CU80WPiezoMove ()

Definition:

C / C++

```
extern "C" CU80WRAP_API void __stdcall CU80WPiezoMove(DWORD
USBInstance, DWORD USBVersion, DWORD DevID, DWORD EEID, INT32
Axis, INT32 Vel, INT32 Volt);
```

The function performs movement along one of the axes. The movement could be stopped using subsequental call of CU80WPiezoStop (). All the settings for the connection are collected in DeviceRec structure, initialized during CU80WOpen() call.

Parameters:

USBInstance, USBVersion, DevID, EEID – represent the corresponding fields of a DeviceRec structure, which contains the settings of the opened connection.

Axis – Determines the axis of the movement (Axis = 1 – X Axis, Axis = 2 – Y Axis, Axis = 3 – Z Axis)
Vel – Velocity in X-, Y-, Z- direction (-63 .. 63) (0 = stop) (±1 = single step)
Volt – Voltage in X-, Y-, Z- direction (22 …82)

Returns:

Example:

C++

```
CU80WPiezoMove (USBInstance, USBVersion, DevID, EEID,1,-
63,40);
```

CU80WGetEEPROMInfo ()

Definition:

C / C++

```
extern "C" CU80WRAP_API void __stdcall
CU80WGetEEPROMInfo(
DWORD USBInstance, DWORD USBVersion, DWORD DevID, DWORD EEID,
DWORD* USBVendorID,
DWORD* USBProductID,
DWORD* USBDeviceID,
DWORD* DeviceID,
DWORD* EEPromID,
DWORD* Version,
DWORD* SerialNumber,
DWORD* CustomerID,
char*  Company,     //[32]
char*  Date,        //[32]
char*  ProductStr,  //[32]
char*  Customer,    //[32]
char*  CustomerStr);//[32]
```

The function returns a set of collected information from EEPROM. All the settings for the connection are collected in DeviceRec structure members, initialized during Servo3WAxUSB2Open() call.

Parameters:

USBInstance, USBVersion, DevID, EEID – represent the corresponding fields of a DeviceRec structure, which contains the settings of the opened connection.
USBVendorID,
USBProductID,
USBDeviceID,
DeviceID,
EEPromID,
Version,
SerialNumber,
CustomerID,
Company,
Date,
ProductStr,
Customer,
CustomerStr

– The parameters correspond to the members of USBEEProm structure, which is used in interface between the wrapper dll and the CU80USB1.dll.
Returns:

Example:

C++

```
CU80WGetEEPROMInfo(USBInstance, USBVersion, DevID, EEID,
  & USBVendorID,
  & USBProductID,
  & USBDeviceID,
  & DeviceID,
  & EEPromID,
  & Version,
  & SerialNumber,
  & CustomerID,
  Company,
  Date,
  ProductStr,
  Customer,
  CustomerStr);
```

CU80WrapperInit ()

Definition:

C / C++

```
extern "C" CU80WRAP_API void __stdcall CU80WrapperInit();
```

The function performs initialization of the wrapper dll. It is either called automatically during the loading of the DLL into the memory or manually.

Parameters:

Returns:

Example:

C++

```
CU80WrapperInit();
```

CU80WrapperDispose ()

Definition:

C / C++

```
extern "C" CU80WRAP_API void __stdcall CU80WrapperDispose();
```

The function releases all the memory and resources, allocated during work of the wrapper dll. It is called automatically when the dll is being removed from memory or it can be accessed manually.
Parameters:

Returns:

Example:

C++

```
CU80WrapperDispose();
```

Win32 sample application

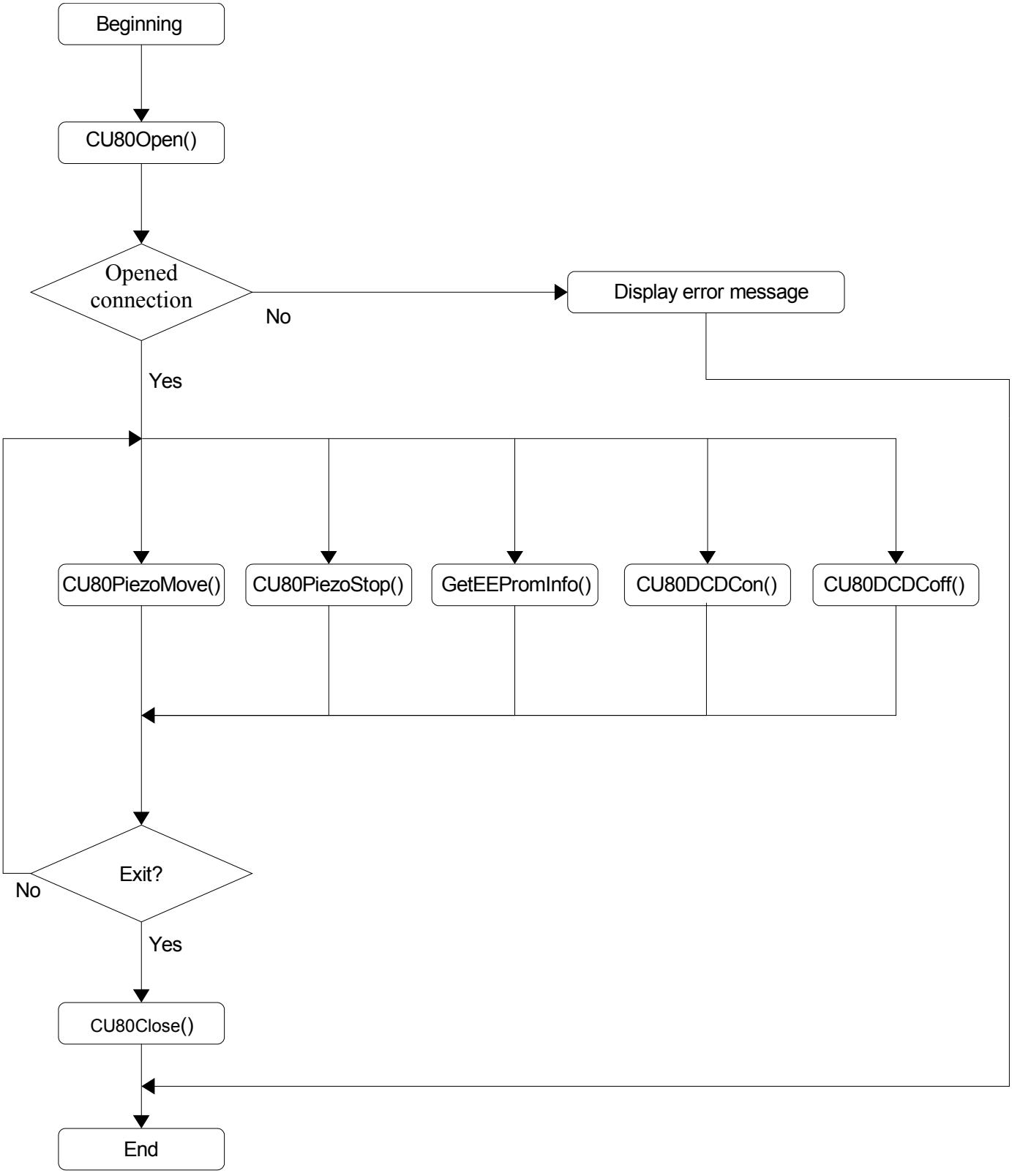
The **Cu17App** subdirectory contains a Win32 C++ application, which can be used as a sample for using the CU17/CU80 Wrapper DLL. Currently it is used only for test and debug purposes, it has no graphical user interface (GUI).

Appendixes

Appendix 1 Structure of the project

The workspace consists of the following directories:
.\bin The output directory, where the binaries are placed. The CU80USB1.DLL should be present in this folder.
.\Docs Contains this document.
.\Cu17APP Contains the sources of the sample C++ Win32 application.
.\Cu17Wrap Contains the wrapper project and source files.

Appendix 2 Block diagram of using the CU80USB1 interface library



Appendix 3 Structure of the Cu17Wrap interface dll

The Cu17Wrap subdirectory contains the project file and sources for the C++ Win32 dynamic linked library. The following files are included in this directory:

- **Cu17Wrap.cpp**
Defines the entry points for the application. All the exportable functions are included here.

- **dllLoader.cpp, dllLoader.h**
An utility class for dynamic loading of the dll is defined here.

- **USBdllLoader.cpp, USBdllLoader.h**
A superstructure of the dllLoader class, specific for the CU80USB1.dll is defined here. This class contains the methods, encapsulating the Cu80USB1.dll functions.

- **Cu17Wrap.h**
The header file contains the definitions of the exportable functions.

- **Cu17Wrap.lib**
Static library, containing the function prototypes

- **Cu17Wrap.dll**
The dynamic linked library, containing the code of the functions.