

CU30 Project

Structure of the CU30 Sample project

CU30 project contains interface and command creating functionality, used for access to hardware device via USB. The interface to the USB is realized using the **CU30.dll** library functionality. An example Win32 application was prepared, which is illustrating the way of using the hardware features. A Delphi example is available also. An additional project in the same workspace is **CU30Wrap.dll**, which provides a C interface using only basic types (suitable for access from other applications like LabView etc.).

CU30.DLL Interface library

The **CU30.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 connection to it, performing of step or continuous movements in all directions, etc. The CU30.dll exports following functions:

CU30Open ()

Declaration:

<i>C/C++</i>	void _stdcall CU30Open (R256 * errMsg, DeviceRec * pdevRec);
<i>Delphi</i>	function CU30Open(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:

<i>C/C++</i>	<pre>typedef struct delphiDevRec { DWORD USBInstance; DWORD USBVersion; DWORD DevID; DWORD EEID; } DeviceRec; typedef struct r256 { char A[256]; }R256;</pre>
<i>Delphi</i>	<pre>DeviceRec = record USBInstance,USBVersion,DevID,EEID:dword; end;</pre>

Parameters:

pDevRec - a pointer to a DeviceRec structure, which contains the settings.
 USBInstance - returned by the routine [0..15]
 EEID - identifies the device to be addressed [0..15]
 USBVersion - already preset for the device CU30 (value will be overwritten).
 DevID - already preset for the device CU30 (value will be overwritten).

Output: Error message

errMessage (C++) - A null-terminated string, representing the error if an error occurs or an empty string if the connection was opened successfully.

Returns:

(Delphi) A null-terminated string, representing the error if an error occurs or an empty string if the connection was opened successfully.

Example:

C++

```

DeviceRec ddr;
ddr.USBInstance = 0;
ddr.USBVersion = 1;
ddr.DevID = 1;
ddr.EEID = 0;
R256 tempR;
((pCU30Open)ifCU30Open.ptrToFunc)(&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:=1;
USB01.EEID:=0;

sc:=CU30Open(USB01);
s:=string(sc);

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

CU30Close()**Declaration:**

C/C++

```
void _stdcall CU30Close (DeviceRec devRec );
```

Delphi

```
procedure CU30Close(devRec:DeviceRec); stdcall
```

The function closes a connection to the hardware through the selected USB port, opened with previous call of CU30Open() function. All the settings for the connection are collected in DeviceRec structure, initialized during

CU30Open() call.

Parameters:

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

Returns:

Example:

```
C++
| DeviceRec ddr;
| ((pCU30Close)ifCU30Close.ptrToFunc) (ddr);
```

```
Delphi
| USB01: DeviceRec;
| CU30Close(USB01);
```

CU30Stop()

Declaration:

```
C/C++
| void _stdcall CU30Stop (DeviceRec devRec );
Delphi
| procedure CU30PiezoStop(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 CU30Open() call.

Parameters:

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

Returns:

Example:

```
C++
| DeviceRec ddr;
| ((pCU30Stop)ifCU30Stop.ptrToFunc) (ddr);
```

```
Delphi
| USB01: DeviceRec;
| CU30PiezoStop(USB01);
```

CU30DCDCon()

Declaration:

```
C/C++
| void _stdcall CU30DCDCon (DeviceRec devRec );
Delphi
| procedure CU30DCDCon (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 CU30Open() call.

Parameters:

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

Returns:**Example:**

```
C++
DeviceRec ddr;
((pCU30DCDCon)ifCU30DCDCon.ptrToFunc) (ddr);
```

```
Delphi
USB01: DeviceRec;
CU30DCDCon(USB01);
```

CU30DCDCoff()**Declaration:**

```
C/C++
void _stdcall CU30DCDCoff (DeviceRec devRec );

Delphi
procedure CU30DCDCoff (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 CU30Open() call.

Parameters:

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

Returns:**Example:**

```
C++
DeviceRec ddr;
((pCU30DCDCoff)ifCU30DCDCoff.ptrToFunc) (ddr);
```

```
Delphi
USB01: DeviceRec;
CU30DCDCoff(USB01);
```

CU30Move()**Declaration:**

```
C/C++
void _stdcall CU30Move(DeviceRec ddr, long Axis, long Speed, long
Timeout);

Delphi
procedure CU30Move(ddr:DeviceRec; Axis:long, Speed:long;
Timeout:long); stdcall
```

The function performs movement along one of the axes. The movement could be stopped using CU30PiezoStop() or adjusting the Timeout parameter. All the settings for the connection are collected in DeviceRec structure, initialized during CU30Open() call.

If continuous movement is required, the command has to be resend with a period higher than the timeout period (Timeout=2..254). Alternatively the command has to be send once with Timeout=255

Parameters:

ddr – DeviceRec structure, which contains the settings of the opened connection.
 Speed – Velocity of the movement; Speed = [-1000...-1, 1...+1000] , Speed = 0 => Speed = 1;
 Axis – Determines the axis of the movement (1 = X-Axis, 2 = Y-Axis, 3 = Z-Axis)
 Timeout – Determines the movement duration. Timeout = [2..255]
 If Timeout < 2, – the duration of the movement will be: Duration = 2 * 0.016 sec.
 If Timeout = [2...254], – the duration of the movement will be: Duration = Timeout * 0.016 sec.
 If Timeout > 254, – the timeout will be disabled.

Returns:

Example:

C++

```
DeviceRec ddr;
((pCU30Sweep)ifCU30Sweep.ptrToFunc)(ddr,1,200,255);
```

Delphi

```
USB01: DeviceRec;
CU30Sweep(USB01,1,200,255);
```

CU30Step()

Declaration:

C/C++

```
void _stdcall CU30 Step(DeviceRec ddr, long Axis, long Vel, long Steps);
```

Delphi

```
procedure CU30Step(ddr:DeviceRec; Axis:long; Vel:long; Steps:long); stdcall
```

The function performs movement along one of the axes with a defined number of steps. The movement could be stopped using the subsequent call of CU30Stop(). All the settings for the connection are collected in DeviceRec structure, initialized during CU30Open() call.

Parameters:

ddr – DeviceRec structure, which contains the settings of the opened connection.
 Axis – Determines the axis of the movement (1 = X-Axis, 2 = Y-Axis, 3 = Z-Axis)
 Vel – Velocity of the movement; Vel = [-1000...-1, +1...+1000], Vel = 0 => Vel = 1;
 Steps – Determines the number of steps for the movement, Steps = [1... 1.000.000]
 If Steps < 1, – the number of steps for the movement will be: 1
 If Steps > 1.000.000 – the number of steps for the movement will be: 1.000.000

Returns:

Example:

C++

```
DeviceRec ddr;
((pCU30Step)ifCU30Step.ptrToFunc)(ddr,1,200,100);
```

Delphi

```
USB01: DeviceRec;
CU30Step(USB01,1,200,100);
```

CU30GetUSBEEPromInfo()**Declaration:****C/C++**

```
void _stdcall GetUSBEEPromInfo(USBEEProm * epromInfo, 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;
    char Company[32];
    char Date[32];
    char ProductStr[32];
    char Customer[32];
    char CustomerStr[32];
} 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;

type PChar16 = array[0..31] of char;
PChar32 = array[0..31] of char;
```

Parameters:

DevRec	- DeviceRec structure, which contains the settings.
epromInfo (C++)	- A USBEEProm structure, containing the data from the board.

Returns:

(Delphi)	- A USBEEProm structure, containing the data from the board.
----------	--

Example:

C++	DeviceRec ddr; USBEEProm usbEEProm; ((pGetUSBEEPromInfo)ifGetUSBEEPromInfo.ptrToFunc)(& usbEEProm, ddr);
Delphi	USB01: DeviceRec; EP:=GetUSBEEPromInfo(USB01);

Echo()**Declaration:**

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 CU30Open() 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);

CU30TestReady ()**Declaration:**

C/C++		DWORD _stdcall CU30TestReady(DeviceRec add);
Delphi		function CU30TestReady(MUSBDeviceID:DeviceRec):dword; stdcall

The function returns 0 if the previous command is still running or 1 if the command is already finished. All the settings for the connection are collected in DeviceRec structure, initialized during CU30Open() call.

Parameters:

return value	- (0 - previous command is still running)
	- (1 - previous command is finished)
ddr	- DeviceRec structure, which contains the settings of the opened connection.

Returns:**Example:**

C++		DeviceRec ddr;
		DWORD X = CU30TestReady(ddr);
Delphi		USB01: DeviceRec;
		X := CU30TestReady(USB01);

CU30WRAP.DLL Wrapper library

The **CU30Wrap.dll** is an interface dynamic linked library, which encapsulates the interface to **CU30.dll**. It exports functions, which works only with basic types (int, DWORD, etc, i.e. without using any structures or records. The calling conventions of exportable functions are defined as `__stdcall`. The *extern "C"* attribute on a C++ function turns off name mangling so that the external name becomes compatible with the C language. The CU30Wrap.dll is suitable when it is not possible to use structures, that are needed to interface the CU30.dll 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 CU30Wrap.dll exports following functions:

CU30WOpen ()

Declaration:

```
C/C++
extern "C" CU30WRAP_API char * __stdcall CU30WOpen(  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;

CU30WRAP_API: __declspec(dllexport) or __declspec(dllimport)
```

Parameters:

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

Returns:

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

Example:

```
C/C++
USBVersion = 1;
USBInstance = 0;
DevID = 1;
EEID = 0;
char * tempR = CU30WOpen(&USBInstance, &USBVersion, &DevID, &
EEID);
if ((tempR!=NULL) && ( tempR[0]!='\0'))
{
    MessageBox(hDlg, tempR,"USB Error",MB_OK);
}
```

CU30WClose()**Declaration:**

```

C/C++
extern "C" CU30WRAP_API void __stdcall CU30WClose(    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 CU30WOpen() function. All the settings for the connection are collected in DeviceRec structure members, initialized during CU30WOpen() 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++
CU30WClose(USBInstance, USBVersion, DevID, EEID);

```

CU30WStop()**Declaration:**

```

C/C++
extern "C" CU30WRAP_API void __stdcall CU30WStop(    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 CU30WOpen() 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++
CU30WStop(USBInstance, USBVersion, DevID, EEID);

```

CU30WDCDCon ()**Declaration:**

```

C/C++
extern "C" CU30WRAP_API void __stdcall CU30WDCDCon(    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 CU30WOpen() 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++
| CU30WDCDCon(USBInstance, USBVersion, DevID, EEID);
```

CU30WDCDCoff ()

Declaration:

```
C/C++
| extern "C" CU30WRAP_API void __stdcall CU30WDCDCoff(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 CU30Open() 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++
| CU30WDCDoff(USBInstance, USBVersion, DevID, EEID);
```

CU30WSweep()

Declaration:

```
C/C++
| extern "C" CU30WRAP_API void __stdcall CU30WSweep(    DWORD
| USBInstance, DWORD USBVersion, DWORD DevID, DWORD EEID, int Vel,
| int Axis, int Timeout);
```

The function performs continuous movement along one of the axes. The movement could be stopped using CU30WStop() or adjusting the Timeout parameter. All the settings for the connection are collected in DeviceRec structure members, initialized during CU30WOpen() call.

Parameters:

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

Vel – Velocity of the movement; Vel = [-1000...-1, 1...+1000] , Vel = 0 => Vel = 1;

Axis – Determines the axis of the movement (1 = X-Axis, 2 = Y-Axis, 3 = Z-Axis)

Timeout – Determines the movement duration. Timeout = [2..255]

If Timeout < 0, - the duration of the movement will be: Duration = 2 * 0.016 sec.

If Timeout = 0, - the timeout will be disabled.

If Timeout = 1, - the duration of the movement will be: Duration = 2 * 0.016 sec.

If Timeout = [2...254], - the duration of the movement will be: Duration = Timeout * 0.016 sec.

If Timeout > 254, - the timeout will be disabled.

Returns:**Example:**

```
C++
| CU30WSweep (USBInstance, USBVersion, DevID, EEID, 200, 1, 0);
```

CU30WStep()**Declaration:**

```
C/C++
| extern "C" CU30WRAP_API void __stdcall CU30WStep(DWORD
| USBInstance, DWORD USBVersion, DWORD DevID, DWORD EEID, int Axis,
| int n, int Vel);
```

The function performs movement along one of the axes with a defined number of steps. The movement could be stopped using the subsequent call of CU30WStop(). All the settings for the connection are collected in DeviceRec structure members, initialized during CU30WOpen() 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 (1 = X-Axis, 2 = Y-Axis, 3 = Z-Axis)

n – Determines the number of steps for the movement, n = [1... 1.000.000]

If n < 1, – the number of steps for the movement will be: 1

If n > 1.000.000 – the number of steps for the movement will be: 1.000.000

Vel – Velocity of the movement; Vel = [-1000...-1, +1...+1000], Vel = 0 => Vel = 1;

Returns:**Example:**

```
C++
| CU30WStep (USBInstance, USBVersion, DevID, 1, 200, 100);
```

CU30WGetEEpromInfo ()**Declaration:**

```
C/C++
| extern "C" CU30WRAP_API int __stdcall CU30WGetEEpromInfo(
| DWORD USBInstance, DWORD USBVersion, DWORD DevID, DWORD EEID,
| DWORD * pUSBVendorID,
| DWORD * pUSBProductID,
| DWORD * pUSBDeviceID,
| DWORD * pDeviceID,
| DWORD * pEEPromID,
| DWORD * pVersion,
| DWORD * pSerialNumber,
| DWORD * pCustomerID,
| char * pCompany,
| char * pDate,
| char * pProductStr,
| char * pCustomer,
```

```
| char * pCustomerStr);
```

The function collects information about EEPROM and returns it in the output parameters. All the settings for the connection are collected in DeviceRec structure members, initialized during CU30WOpen() call.

Parameters:

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

pUSBVendorID - pointer to a DWORD variable, where the vendor ID will be returned.

pUSBProductID - pointer to a DWORD variable, where the product ID will be returned.

pUSBDeviceID - pointer to a DWORD variable, where the USB device ID will be returned.

pDeviceID - pointer to a DWORD variable, where the device ID will be returned.

pEEPROMID - pointer to a DWORD variable, where the EEPROM id will be returned.

pVersion - pointer to a DWORD variable, where the version will be returned.

pSerialNumber - pointer to a DWORD variable, where the serial number will be returned.

pCustomerID - pointer to a DWORD variable, where the customer ID will be returned.

pCompany - pointer to a string variable, where the company name will be returned. The size of the buffer must be at least 32 characters.

pDate - pointer to a string variable, where the date will be returned. The size of the buffer must be at least 32 characters.

pProductStr - pointer to a string variable, where the product name will be returned. The size of the buffer must be at least 32 characters.

pCustomer - pointer to a string variable, where the customer name will be returned. The size of the buffer must be at least 32 characters.

pCustomerStr - pointer to a string variable, where the customer string will be returned. The size of the buffer must be at least 32 characters.

Returns:

Example:

C++

```
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];
int val = CU30WGetEEPROMInfo(USBInstance, USBVersion, DevID, EEID, &
USBVendorID, & USBProductID, & USBDeviceID,
& DeviceID, & EEPROMID, & Version, & SerialNumber, CustomerID,
Company, Date, ProductStr, Customer, CustomerStr);
```

CU30WrapperInit ()

Declaration:

C/C++

```
| extern "C" CU30WRAP_API void __stdcall CU30WrapperInit();
```

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++  
| CU30WrapperInit();
```

CU30WrapperDispose ()

Declaration:

```
C/C++  
| extern "C" CU30WRAP_API void __stdcall CU30WrapperDispose();
```

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:

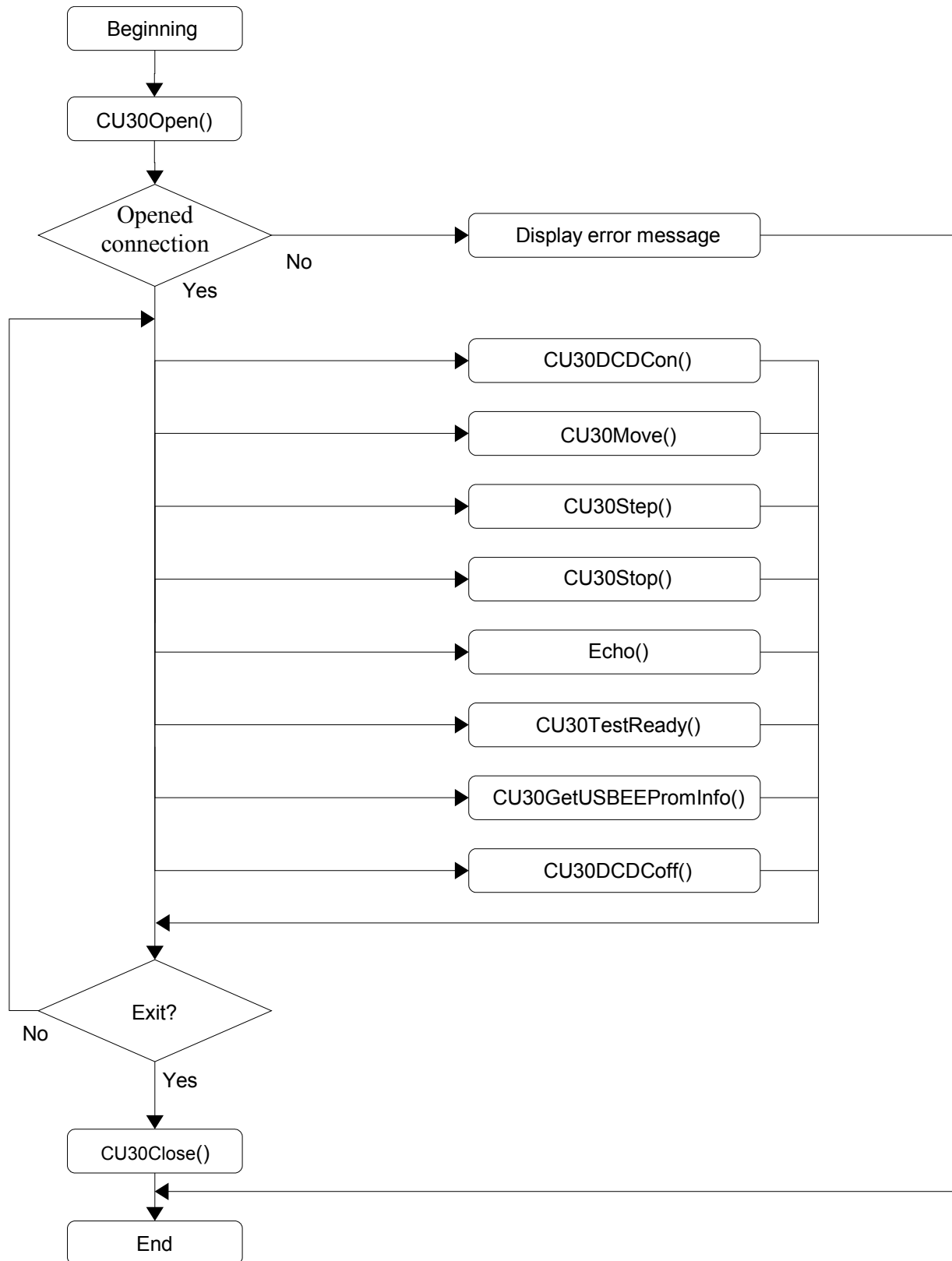
```
C++  
| CU30WrapperDispose();
```

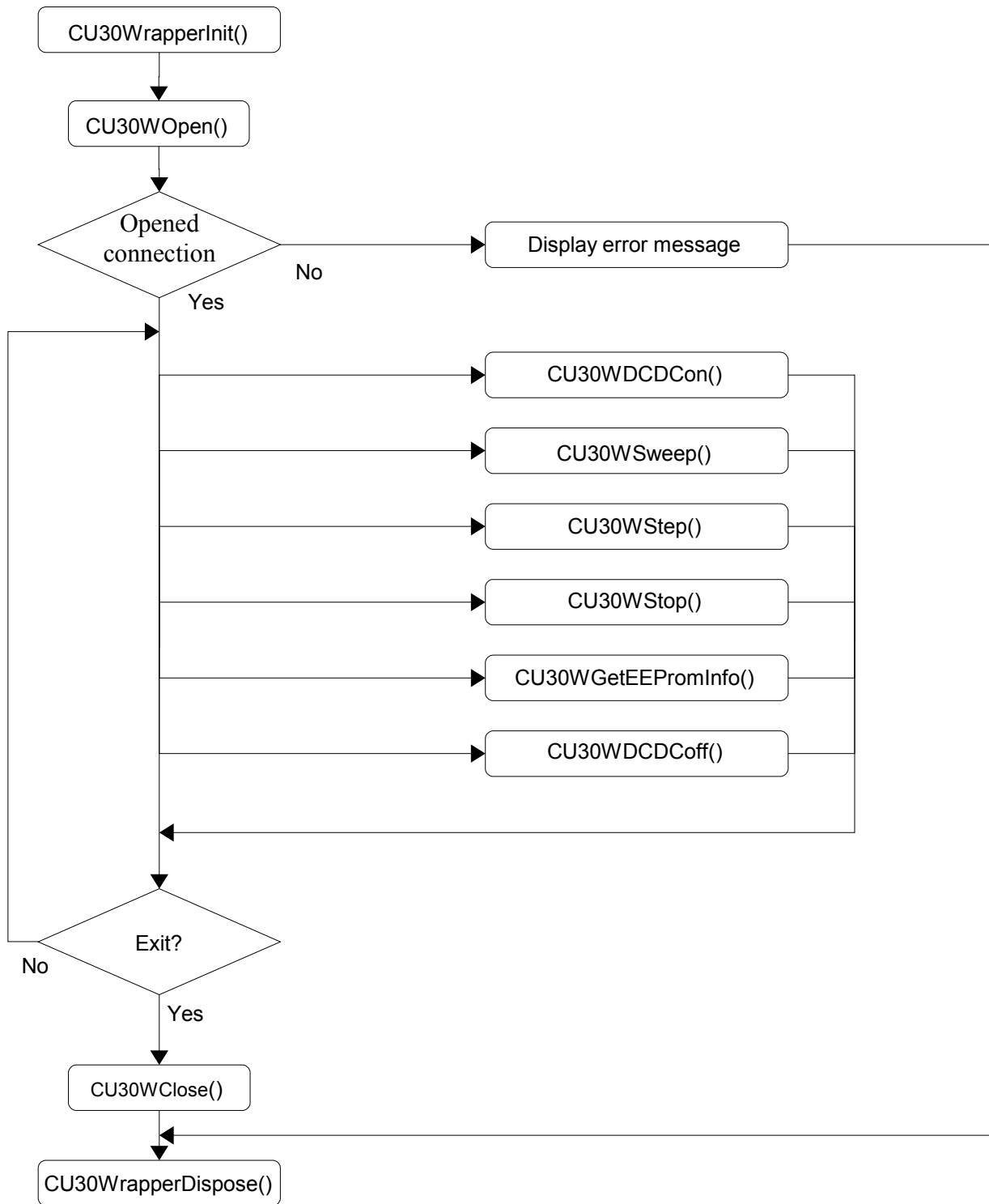

Appendixes

Appendix 1 Structure of the project

The workspace consists of the following directories:

- .\\bin -The output directory, where the binaries are placed. The CU30.dll should be present in this folder.
- .\\Docs -Contains this document.
- .\\CU30_App -Contains the sources of the sample C++ Win32 application. For more details see Appendix 4.
- .\\CU30Wrap -Contains the sources of the Wrapper dll.

Appendix 2 Block diagram of using the CU30 interface library

Appendix 3 Block diagram of using the CU30Wrap interface library

Appendix 4 Structure of the CU30_APP sample application

The CU30_App subdirectory contains the project file and sources for the C++ Win32 sample application. The following files are included in this directory:

CU30_App.cpp

- Defines the entry point for the application. The main window callback is defined here.

CU30_mainScreen.cpp

- The main dialog box callback function and the utility functions are placed in this file.

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 CU30.dll is defined here. This class contains the methods, encapsulating the CU30.dll functions, which are directly accessed from CU30_mainScreen code.

resource.h

- contains resource definitions

resource.rc

- contains definitions for dialog boxes, strings, etc.

Appendix 5 Structure of the CU30Wrap interface dll

The CU30Wrap subdirectory contains the project file and sources for the C++ Win32 dynamic linked library. The following files are included in this directory:

CU30Wrap.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 CU30.dll is defined here. This class contains the methods, encapsulating the CU30.dll functions, which are directly accessed from CU30_mainScreen code.

CU30Wrap.h

- The header file contains the definitions of the exportable functions.

CU30Wrap.lib

- Static library, containing the function prototypes

CU30Wrap.dll

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

CU30 PROJECT.....	1
STRUCTURE OF THE CU30 SAMPLE PROJECT.....	1
CU30.DLL Interface library	1
CU30Open ().....	1
CU30Close().....	2
CU30Stop().....	3
CU30DCDCon().....	3
CU30DCDCoff().....	4
CU30Move().....	4
CU30Step().....	5
CU30GetUSBEEPromInfo().....	6
Echo().....	8
CU30TestReady ().....	8
CU30WRAP.DLL Wrapper library	9
CU30WOpen ().....	9
CU30WClose().....	10
CU30WStop().....	10
CU30WDCDCon ().....	10
CU30WDCDCoff ().....	11
CU30WSweep().....	11
CU30WStep().....	12
CU30WGetEEPromInfo ().....	12
CU30WrapperInit ().....	12
CU30WrapperDispose ().....	12
Win32 sample application	15
APPENDICES.....	16
APPENDIX 1 STRUCTURE OF THE PROJECT.....	16
APPENDIX 2 BLOCK DIAGRAM OF USING THE CU30 INTERFACE LIBRARY.....	17
APPENDIX 3 BLOCK DIAGRAM OF USING THE CU30WRAP INTERFACE LIBRARY.....	18
APPENDIX 4 STRUCTURE OF THE CU30_APP SAMPLE APPLICATION.....	19
APPENDIX 5 STRUCTURE OF THE CU30WRAP INTERFACE DLL.....	20