

## Reference Guide for the CN30 COM-DLL

Release: 2.0.0.0

Version: 16-04-2004

### Content

CN30 COM OBJECT .....	1
STRUCTURE OF THE CN30 COM OBJECT .....	2
<i>Interface Excel sheet</i> .....	2
<i>Win32 sample application</i> .....	3
CN30 COM SERVER .....	4
<i>Transmit interface</i> .....	4
HRESULT iConfigPort(VARIANT* paramBlock, VARIANT* retValue); .....	5
HRESULT iPortInfo(VARIANT* retValue); .....	6
HRESULT iSendByteCommand(long Command, long DelayMS, VARIANT* retValue); .....	7
HRESULT iOpenConnection(VARIANT* retValue); .....	7
HRESULT iCloseConnection(VARIANT* retValue); .....	8
HRESULT iInitReadBackStep(VARIANT* pCallback, VARIANT* retValue); .....	8
HRESULT iExtendedErrorInfo(long Error, VARIANT* retValue); .....	9
HRESULT iChangePositionX(long Direction, long Speed, long Steps, long DelayMS, VARIANT* retValue);	10
HRESULT iChangePositionY(long Direction, long Speed, long Steps, long DelayMS, VARIANT* retValue);	11
HRESULT iChangePositionZ(long Direction, long Speed, long Steps, long DelayMS, VARIANT* retValue);	11
APPENDIXES .....	13
APPENDIX 1 STRUCTURE OF THE PROJECT .....	13
APPENDIX 2 BLOCK DIAGRAM OF USING THE CN30 COM SERVER .....	14
APPENDIX 3 STRUCTURE OF THE CN30_APP SAMPLE APPLICATION .....	15

# CN30 COM object

## Structure of the CN30 COM object

CN30 COM server encapsulates the COM port interface and command creating functionality, used for access to hardware device. The COM interface function receives parameters as long values and arrays of double variables. The readback and error codes are returned to the calling application packed in VARIANT structure. An Excel sheet with according VBA code was created as interface to the COM object features. The COM object is able to handle multiple interfaces to the hardware in the same time.

## Interface Excel sheet

The **CN30\_test\_sheet.xls** Excel sheet is used for input of parameters, COM port configuration and output of the error codes. The sheet is shown on the figure below (**fig.1**):

The following parameters are located on the sheet:

### 1. COM port config

- COM port
- Baud Rate
- Data Bits
- DTR
- RTS
- Parity
- Stop Bits
- Xon/Xoff

### 2. Speed

- Fast
- Medium
- Slow
- Very Slow

### 3. Steps

- Steps

The using of the Excel sheet should be performed in following steps:

#### 1. Adjusting of the configuration of the COM port.

The default settings, used in the COM server are:

- COM port = COM2
- Baud Rate = 9600
- Data Bits = 8
- DTR = FALSE
- RTS = FALSE
- Parity = 0
- Stop Bits = 0
- Xon/Xoff = FALSE

If it is necessary to adjust one or more parameters, it could be done using the Port Settings range on the sheet (B9:B16). In this case only, the customer should use 'Config Port' button to transfer the settings to COM server.

	A	B	C	D	E	F
1	Open Connection					
2	Close Connection					
3						
6	Config Port					
7						
8	<b>Port Settings</b>					
9	COM Port	COM2				
10	Baud Rate	9600				
11	Data Bits	8				
12	DTR (0-False, 1-True)	0				
13	RTS (0-False, 1-True)	0				
14	Parity	0				
15	Stop Bits	0				
16	XonXoff (0-False, 1-True)	0				
17	<b>Speed</b>					
18	<input checked="" type="radio"/> Fast					
19	<input type="radio"/> Medium					
20	<input type="radio"/> Slow		Steps			200
21	<input type="radio"/> Very Slow		Readback			50
22			Status			

Figure 1

## 2. Opening of the connection to the port.

In the sample sheet this could be done using the button 'Open Connection'

## 3. Positioning the device.

The customer could use the direction keys to position the device in the 3 dimensions. This could be done in following 2 modes:

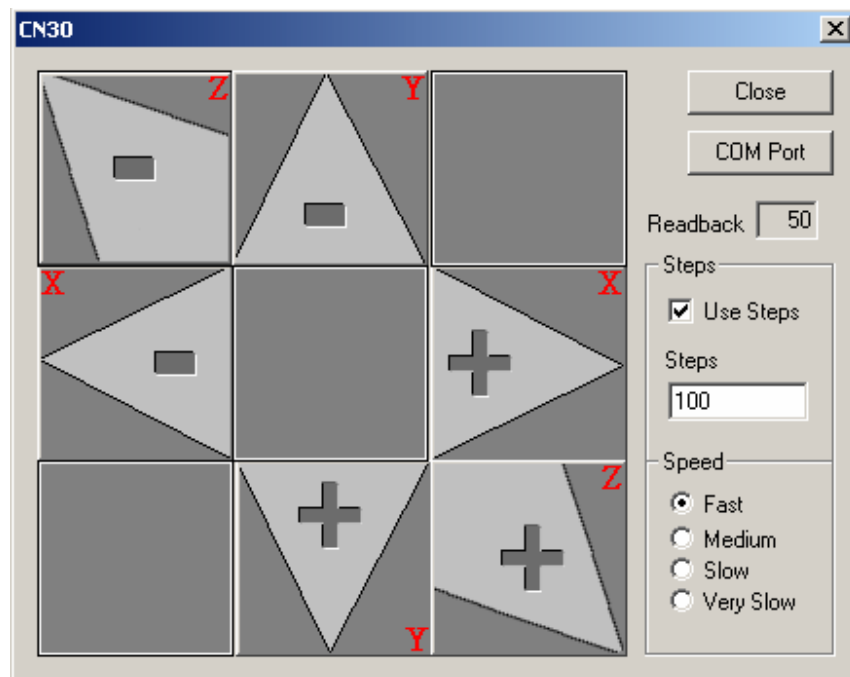
- Step mode: If the value for steps ( E20 ) is greater than zero, the position will be changed with this amount of steps in the direction, selected with pressing the direction key.
- Continuous mode: This is the mode when the number of steps is set to zero. In this case pressing and holding the button, the customer can position the device.

## Win32 sample application

The **CN30\_app** subdirectory contains a Win32 C++ application, which can be used as a sample for using the CN30 server features (see fig.2).

The opening and closing of the connection is done automatically when the dialog box is created and closed. The COM settings could be adjusted using the **COM port** dialog box. There are 6 buttons for movement in the 3 dimensions, which could be used in the same way as ones in the Excel sheet. If the **Steps** check box is not checked or the number of steps is zero, the device will move until the button is being pressed. Otherwise the number of steps will be taken from the edit box. The speed can be adjusted in the same way as in the Excel sheet, the readback is displayed in the read-only field.

For the structure of the project see the Appendix 3.



**Figure 2**

### **CN30 COM server**

The CN30COM.dll is a COM server built as dynamically linked library, which integrates an interface to the COM port communication functions. This is in-process COM server, which should be registered as any COM server and could be accessed from other applications using both C / C++ and VBA.

The COM server contains following interface:

- **Transmit**

**Transmit** interface contains methods for access to COM port transfer functionality. It contains functions for configuration of the COM port, to send a command and to receive a readback. It has functions, which encapsulate the process of creation the necessary commands, thus simplifying the calling application (for example iChangePositionX() function). The return values are exclusively from VARIANT type, which is more convenient when calling from VBA. The last parameter of the functions is always VARIANT \* retValue, where the actual return value is placed. The type of the returned value of the function is HRESULT and contains system value indicating whether the function is called correctly ( S\_OK in most cases ). The parameters, describing the port settings are received as block of double / string elements. As it is described below, the data arrays are expected in VARIANT form. The both string and numerical values are accepted.

### **Transmit interface**

TheTransmit interface contains the following functions:

## HRESULT iConfigPort(VARIANT\* paramBlock, VARIANT\* retValue);

Reconfigures the COM port settings, which will be used during establishing of the next connection. There should be not an open connection to the port already.

### Parameters:

ParamBlock – a pointer to a VARIANT structure, which contains the settings. They are organized as an one-dimensional numerical array, which has following form:

No	Field	Comments
1.	COM port	1 .. 5
2.	Baud Rate	9600, 14400, 19200, ...
3.	Data Bits	0 .. 8
4.	DTR	FALSE (0), TRUE (1)
5.	RTS	FALSE (0), TRUE (1)
6.	Parity	NOPARITY (0), EVENPARITY (1), ODDPARITY (2), MARKPARITY (3), SPACEPARITY (4)
7.	Stop Bits	ONESTOPBIT (0), ONE5STOPBITS (1), TWOSTOPBITS (2)
8.	Xon Xoff	FALSE (0), TRUE (1)

Table 1

### Returns:

- 0 - Success
  - 5 - Invalid type of the input parameter **paramBlock**
  - 4 - Invalid number of parameters in the parameter block
  - 3 - There exists already an opened connection to the port. It must be closed first (using [iCloseConnection\(\)](#))
  - 2 - One of the parameters in the parameter block has invalid value
- Extended error information could be retrieved using [iExtendedErrorInfo\(\)](#) method.

### Example:

#### VBA

```
Dim myObject As Transmit
Dim mmat() As Long
Dim res as Variant

ReDim mmat(7)
Set myObject = CreateObject("CN30COM.Transmit")

Dim i As Integer
For i = 0 To 7
    mmat(i) = Sheets(tsheets).Cells(i, 2)
Next
res = myObject.iConfigPort(mmat)
res = myObject.iExtendedErrorInfo(res)
Sheets(tsheets).Cells(21, 5) = res
```

#### C++

```
#include "com_connection.h"

ITransmit * iTrans;
double * params = new double[8];
int i;
long indexes[2];
VARIANT Var, Var_var, res;
HRESULT hR;

VariantInit ( &Var );

Var.vt=VT_ARRAY | VT_VARIANT;
SAFEARRAYBOUND rgsabound[2];
rgsabound[0].lLbound = 0;
```

```

rgsabound[1].lLbound = 0;
rgsabound[0].cElements = 8;
rgsabound[1].cElements = 1;
Var.parray= SafeArrayCreate(VT_VARIANT, 2, rgsabound);

params[0] = paData->Port;
params[1] = paData->BaudRate;
params[2] = paData->DataBits;
params[3] = paData->DTR ;
params[4] = paData->RTS;
params[5] = paData->Parity;
params[6] = paData->StopBits;
params[7] = paData->XOnXoff;
for(i=0;i<8;i++)
{
    indexes[0]=(long)i;
    indexes[1]=0;
    VariantInit(&Var_var);
    Var_var.vt =VT_R8;
    Var_var.dblVal =params[i];
    hR=SafeArrayPutElement( (Var.parray),indexes,
        (void far *)&(Var_var));
}
delete params;
iTrans->iConfigPort(&Var,&res);
VariantClear( &Var );

```

### **HRESULT iPortInfo(VARIANT\* retValue);**

The iPortInfo() method is used to retrieve the current (default or modified) settings, used during establishing of the connection to COM port. The settings are received in an array with the same size and meaning as the one, used in [iConfigPort\(\)](#).

#### **Returns:**

The function returns a VARIANT with one-dimensional numeric array with the same form as the one, used in iConfigPort() (see Table 1).

#### **Example:**

##### **VBA**

```

Dim myObject As Transmit
Dim res as Variant

Set myObject = CreateObject("CN30COM.Transmit")

Dim i As Integer
res = myObject.iPortInfo()
For i = 0 To 7
    Sheets(tsheet).Cells(i, 2) = res(i)
Next

```

##### **C++**

```

#include "com_connection.h"

ITransmit * iTrans;
int i;
HRESULT hResult;
long Indexes[2];
VARIANT vRes;
iTrans->iPortInfo(&vRes);

double pParams[8];
Indexes[1]=0;
for(i=0;i<8;i++)

```

```

{
    Indexes[0]=i;
    HRESULT= SafeArrayGetElement(vRes.parray,
        Indexes,&(pParams[i]));
    if(FAILED(hResult))
        return;
}
paData->Port      = (int)pParams[0];
paData->BaudRate  = (int)pParams[1];
paData->DataBits  = (int)pParams[2];
paData->DTR       = (int)pParams[3];
paData->RTS       = (int)pParams[4];
paData->Parity    = (int)pParams[5];
paData->StopBits  = (int)pParams[6];
paData->XOnXoff   = (int)pParams[7];
VariantClear(&vRes);

```

### **HRESULT iSendByteCommand(long Command, long DelayMS, VARIANT\* retValue);**

The function could be used for sending of 'low level' commands to the hardware. The command ( as 1 byte value ) is sent as input parameter from the calling application.

#### **Parameters:**

Command - The command to be sent to the hardware as 1 byte hex value.  
 DelayMS - The delay (in milliseconds) to be wait after the execution of the command.

#### **Returns:**

0 - Success  
 1 - There is no established connection. One must be created first (using [iOpenConnection\(\)](#)).

#### **Example:**

##### **VBA**

```

Dim myObject As Transmit
Dim res as Variant

Set myObject = CreateObject("CN30COM.Transmit")

res = myObject.iSendByteCommand(253,20)
res = myObject.iExtendedErrorInfo(res)
Sheets(tsheet).Cells(21, 5) = res

```

##### **C++**

```

#include "com_connection.h"

ITransmit * iTrans;
int i;
HRESULT hResult;
VARIANT vRes;
iTrans-> iSendByteCommand(253,20,&vRes);
VariantClear(&vRes);

```

### **HRESULT iOpenConnection(VARIANT\* retValue);**

The function opens a connection to the selected COM port. The COM port and the settings could be adjusted with a previous call of [iConfigPort\(\)](#) or the defaults will be used instead. To see the current settings the customer can use [iPortInfo\(\)](#). The connection must be closed using [iCloseConnection\(\)](#).

#### **Returns:**

0 - Success  
 4 - There exists already an opened connection to the port. It must be closed first (using [iCloseConnection\(\)](#))

**Example:****VBA**

```
Dim myObject As Transmit
Dim res as Variant

Set myObject = CreateObject("CN30COM.Transmit")

res = myObject.iOpenConnection()
```

**C++**

```
#include "com_connection.h"

ITransmit * iTrans;
int i;
HRESULT hResult;
VARIANT vRes;
iTrans-> iOpenConnectioin(&vRes);
VariantClear(&vRes);
```

**HRESULT iCloseConnection(VARIANT\* retValue);**

The function closes a connection to the selected COM port. The connection must be opened using previous call of [iOpenConnection\(\)](#).

**Returns:**

0 - Success  
6 - There is no opened connection to the port. It must be opened first (using **iOpenConnection()**)

**Example:****VBA**

```
Dim myObject As Transmit
Dim res as Variant

Set myObject = CreateObject("CN30COM.Transmit")

res = myObject.iOpenConnection()
...
res = myObject.iCloseConnection()
```

**C++**

```
#include "com_connection.h"

ITransmit * iTrans;
int i;
HRESULT hResult;
VARIANT vRes;
iTrans-> iCloseConnectioin(&vRes);
VariantClear(&vRes);
```

**HRESULT ilnitReadBackStep(VARIANT\* pCallback, VARIANT\* retValue);**

Using this function, the customer can declare the callback function to be executed when a readback is received from the hardware device. The readback is examined using parallel thread, which collects the data and then calls (if defined) the callback function. The callback function should have the following form:

**VBA**

```
Public Function readBack(ByVal val As Long) As Long
```

**C++**

| long CALLBACK readBack(long value)

The returned value from this function is not used in the COM server.

**Parameters:**

pCallback - The pointer to the callback function to be used when a readback is received. **Note:** In VBA the AddressOf function should be used to reach the pointer of the callback function. But this is possible only in MSOffice 2000 and XP.

**Returns:**

0 - Success

**Example:**

**VBA**

```
Dim myObject As Transmit
Dim res as Variant

Set myObject = CreateObject("CN30COM.Transmit")
res = myObject.iInitReadBackStep(AddressOf readBack)

' callback function, used to receive the readback value

Public Function readBack(ByVal bip As Long) As Long
readBackValue = bip
readBack = bip
End Function
```

**C++**

```
#include "com_connection.h"

ITransmit * iTrans;
int i;
HRESULT hResult;
VARIANT vRes, vX;
VariantInit(&vX);
vX.vt = VT_I4;
vX.lVal = (long)readBack;
iTrans->iInitReadBackStep(&vX,&vRes);
VariantClear(&vRes);

long CALLBACK readBack(long value)
{
    return 0;
}
```

**HRESULT iExtendedErrorInfo(long Error, VARIANT\* retValue);**

Provides extended information for the errors, returned from the other functions in this interface. Returns a string, describing the error code.

**Parameters:**

Error - The error code, returned from one of the other functions.

**Returns:**

The function returns a string description of the error:

0 \_NO\_ERROR: ""  
1 \_NO\_CONNECTION: "No connection established"  
2 \_INVALID\_PARAMS\_VAL: "Invalid parameter value"  
3 \_ALREADY\_CONNECTED: "There already exists a connection"  
4 \_INVALID\_PARAMS\_NO: "Invalid number of parameters"  
5 \_INVALID\_PARAMS : "Invalid parameters"  
6 \_NO\_CONNECTION\_TO\_CLOSE: "No connection to close";

**Example:****VBA**

```
Dim myObject As Transmit
Dim res as Variant

Set myObject = CreateObject("CN30COM.Transmit")

res = myObject.iOpenConnection()
res = myObject.iExtendedErrorInfo(res)
Sheets(tsheet).Cells(21, 5) = res
` if the connection was already opened, in this moment on the sheet will
be outputted "There already exists a connection"
```

**C++**

```
#include "com_connection.h"

ITransmit * iTrans;
int i;
HRESULT hResult;
VARIANT vRes;
iTrans-> iOpenConnection(&vRes);
iTrans-> iExtendedErrorInfo(vRes.lVal,&vRes)
// if the connection was already opened, the vRes.bstrVal
// will contain "There already exists a connection"
VariantClear(&vRes);
```

**HRESULT iChangePositionX(long Direction, long Speed, long Steps, long DelayMS, VARIANT\* retValue);**

'High Level' function for giving the movement in X direction with limited or unlimited number of steps.

**Parameters:**

Direction- The direction of movement ( 0-negative, 1-positive)  
Speed - The speed of the movement (1-Fast, 2-Medium, 3-Slow, 4-Very Slow)  
Steps - Number of steps or 0 for unlimited movement (in this case the movement will continue until the command for stop is not sent using [iSendByteCommand\(\)](#).  
DelayMS - Delay in milliseconds after the execution of the command.

**Returns:**

0 - Success  
1 - There is no established connection. One must be created first (using [iOpenConnection\(\)](#)).

**Example:****VBA**

```
Dim myObject As Transmit
Dim res as Variant

Set myObject = CreateObject("CN30COM.Transmit")

res = myObject.iChangePositionX(0,2,250,20)
```

**C++**

```
#include "com_connection.h"

ITransmit * iTrans;
int i;
HRESULT hResult;
VARIANT vRes;
iTrans-> iChangePositionX(0,2,250,20,&vRes);
VariantClear(&vRes);
```

## **HRESULT iChangePositionY(long Direction, long Speed, long Steps, long DelayMS, VARIANT\* retVal);**

'High Level' function for giving the movement in Y direction with limited or unlimited number of steps.

### **Parameters:**

Direction- The direction of movement ( 0-negative, 1-positive)  
Speed - The speed of the movement (1-Fast, 2-Medium, 3-Slow, 4-Very Slow)  
Steps - Number of steps or 0 for unlimited movement (in this case the movement will continue until the command for stop is not sent using [iSendByteCommand\(\)](#)).  
DelayMS - Delay in milliseconds after the execution of the command.

### **Returns:**

0 - Success  
1 - There is no established connection. One must be created first (using [iOpenConnection\(\)](#)).

### **Example:**

#### **VBA**

```
Dim myObject As Transmit
Dim res as Variant

Set myObject = CreateObject("CN30COM.Transmit")

res = myObject.iChangePositionY(0,2,250,20)
```

#### **C++**

```
#include "com_connection.h"

ITransmit * iTrans;
int i;
HRESULT hResult;
VARIANT vRes;
iTrans-> iChangePositionY(0,2,250,20,&vRes);
VariantClear(&vRes);
```

## **HRESULT iChangePositionZ(long Direction, long Speed, long Steps, long DelayMS, VARIANT\* retVal);**

'High Level' function for giving the movement in Z direction with limited or unlimited number of steps.

### **Parameters:**

Direction- The direction of movement ( 0-negative, 1-positive)  
Speed - The speed of the movement (1-Fast, 2-Medium, 3-Slow, 4-Very Slow)  
Steps - Number of steps or 0 for unlimited movement (in this case the movement will continue until the command for stop is not sent using [iSendByteCommand\(\)](#)).  
DelayMS - Delay in milliseconds after the execution of the command.

### **Returns:**

0 - Success  
1 - There is no established connection. One must be created first (using [iOpenConnection\(\)](#)).

### **Example:**

#### **VBA**

```
Dim myObject As Transmit
Dim res as Variant

Set myObject = CreateObject("CN30COM.Transmit")

res = myObject.iChangePositionZ(0,2,250,20)
```

#### **C++**

```
#include "com_connection.h"

ITransmit * iTrans;
```

```
int i;  
HRESULT hResult;  
VARIANT vRes;  
iTrans-> iChangePositionZ(0,2,250,20,&vRes);  
VariantClear(&vRes);
```

## **Appendixes**

### **Appendix 1 Structure of the project**

The workspace consists of the following directories:

.\BIN The output directory, where the binaries are placed. The CN30COM.DLL is generated in this directory. The CN30\_test\_sheet.xls is located here also.

.\DOCS Contains this document.

.\CN30COM Contains the COM server project.

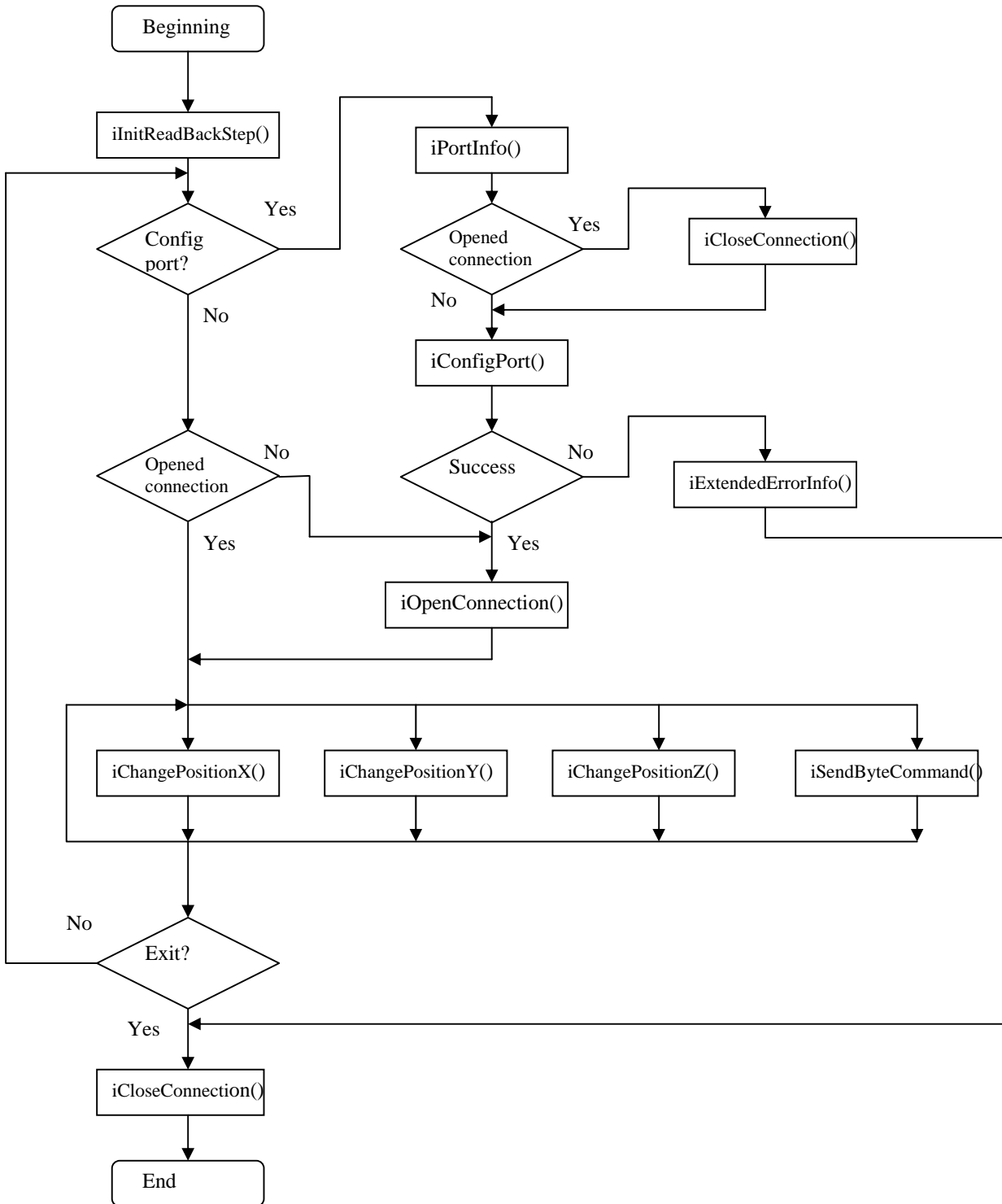
.\CN30COM\HEADERS The headers directory. Contains the headers of the project

.\CN30COM\LIB The library directory. In this directory the VTools.LIB is placed.

.\CN30COM\SOURCES The sources directory. The COM-specific sources and classes are included in this directory.

.\CN30\_APP Contains the sources of the sample C++ Win32 application. For more details see Appendix 3.

**Appendix 2 Block diagram of using the CN30 COM server**



### **Appendix 3 Structure of the CN30\_APP sample application**

The CN30\_APP subdirectory contains the project file and sources for the C++ Win32 sample application. The following files are included in this directory:

CN30\_App.cpp - Defines the entry point for the application. The main window callback is defined here.  
CN30\_com.cpp - The main dialog box callback function and the utility functions are placed in this file.  
COMSettings.cpp - The COM port settings dialog box callback function is defined in this source file.  
COM\_Connection.cpp - COM\_Connection.cpp contains the connection routines, used to establish and release connection to COM server as long as the functions, providing the access to its features.  
CN30COM\_i.c - The system source file, containing the definition of CN30 COM server.  
StdAfx.cpp – Utility file, used for precompiled headers.

CN30COM.h – COM header file, containing definition of iTransmit interface.  
StdAfx.h – used for precompiled headers  
com\_connection.h – Contains definitions of the connection routines in COM\_Connection.cpp  
resource.h – contains resource definitions  
resource.rc – contains definitions for dialog boxes, strings, etc.