

SERASY32.DLL - Dokumentation
Version 1.0

(c) 1998

Josef Bernhardt
Enzianstraße 16
D-93342 Saal / Donau

email: Josef@Bernhardt.de

1. Allgemein	3
2. Interface	3
2.1. Öffnen eines seriellen Kanals.....	3
Name.....	3
Beschreibung	3
Parameter	3
Rückgabe.....	3
2.2. Schliessen eines seriellen Kanals	4
Name.....	4
Beschreibung	4
Parameter	4
Rückgabe.....	4
2.3. Initialisieren eines seriellen Kanals.....	4
Name.....	4
Beschreibung	4
Parameter	4
Rückgabe.....	6
2.4. Senden.....	6
Name.....	6
Beschreibung	6
Parameter	6
Rückgabe.....	7
2.5. Empfangen	7
Name.....	7
Beschreibung	7
Parameter	7
Rückgabe.....	7
2.6. Sendestatus.....	8
Name.....	8
Beschreibung	8
Parameter	8
Rückgabe.....	8
2.7. Empfangsstatus.....	8
Name.....	8
Beschreibung	8
Parameter	8
Rückgabe.....	8
3. Beispielprogramme	9
4. Kontaktadresse	9
Anhang A C++ Beispiel	9
Anhang B Visual Basic Beispiel.....	12

1. Allgemein

SERASY32.DLL wurde für eine einfache Programmierung der seriellen Schnittstellen eines PC geschrieben. Die DLL ist in C++ (unter Borland C++ 5.0) geschrieben. Sie ist eine 32 Bit DLL und ist unter Windows95, Windows98 und Windows NT lauffähig.

SERASY32 kann von jeder Programmiersprache aufgerufen werden. Die folgende Dokumentation enthält eine Beschreibung der Schnittstellen für die Sprachen C und Visual Basic. Da die Microsoft Office Programme inzwischen Visual Basic als Macro Sprache implementiert haben, kann die Interface Beschreibung auch dort angewandt werden.

SERASY32 setzt auf der API des Win32 auf. Das Interface umfaßt 7 Funktionen:

- Öffnen eines seriellen Kanals
- Schliessen eines seriellen Kanals
- Initialisieren eines seriellen Kanals
- asynchrones Senden über einen Kanal
- asynchrones Empfangen über einen Kanal
- Abfrage ob Senden erledigt
- Abfrage ob Empfangen erledigt

Das Senden und Empfangen geschieht asynchron zum aufrufendem Programm. Das aufrufende Programm braucht nach Aufruf des Sendebefehls bzw. des Empfangsbefehls nicht auf Beendigung der seriellen Übertragung zu warten, sondern kann zwischenzeitlich weiterarbeiten.

Der Send- und Empfangspuffer darf maximal 1024 Zeichen umfassen.

SERASY32 verwaltet bis zu 8 serielle Kanäle (die auch von unterschiedlichen Programmen angesprochen werden können).

SERASY32 reicht für die meisten Kommunikationsaufgaben aus. In schwierigeren Fällen muß man wohl direkt auf die (recht komplexe und schwer verständliche) Win32 API ausweichen.

2. Interface

2.1. Öffnen eines seriellen Kanals

Name

int ComOpen (int Port)

Beschreibung

ComOpen() öffnet (falls dieser noch nicht belegt ist) einen seriellen Kanal und liefert einen "handle" zurück. Dieser handle muß (zur Identifizierung des Kanals) bei allen weiteren Aufrufen als Aufrufparameter benutzt werden. Es können bis zu 8 serielle Kanäle geöffnet werden. SERASY32 kann als dynamische Bibliothek von mehreren Programmen angesprochen werden. Andere Funktionen von SERASY32 dürfen erst nach erfolgreichem Öffnen aufgerufen werden.

Parameter

Parameter	C-Typ	Visual Basic Typ	Länge	Beschreibung
Port	int	ByVal as Long	4 Byte	Gibt an, welcher Kommunikations Port geöffnet werden soll. 1 für Com1, 2 für Com2, usw.

Rückgabe

C-Typ	Visual Basic Typ	Länge	Beschreibung
int	Long	4 Byte	Im Erfolgsfall ist der Rückgabewert ein "handle" (Referenz) auf den geöffneten Kanal. Der "handle" wird bei den folgenden Funktionsaufrufen als Parameter erwartet. Im Fehlerfall (der Kanal ist belegt, die Windows Ressourcen sind erschöpft oder es sind alle 8 von SERASY verwaltbaren Kanäle belegt) wird -1 zurückgegeben.

2.2. Schliessen eines seriellen Kanals

Name

int ComClose (int Handle)

Beschreibung

ComClose schließt den durch "handle" gekennzeichneten Kanal. Nach Beendigung aller Kommunikationsaufgaben durch Ihr Programm, sollten Sie den Kanal schliessen um in bei erneutem Start wieder zur Verfügung zu haben (die DLL wird nicht unbedingt mit Ihrem Programm beendet, sie kann auch weiterhin, wenn sie von einem anderen Programm noch benötigt wird, im Speicher verbleiben).

Parameter

Parameter	C-Typ	Visual Basic Typ	Länge	Beschreibung
Handle	int	ByVal as Long	4 Byte	Gibt an, welcher Kommunikations Port geschlossen werden soll.

Rückgabe

C-Typ	Visual Basic Typ	Länge	Beschreibung
int	Long	4 Byte	Im Erfolgsfall wird 1 zurückgegeben. Im Fehlerfall (referenzierter Kanal war nicht geöffnet) wird -1 zurückgegeben.

2.3. Initialisieren eines seriellen Kanals

Name

int ComInit (int Handle, unsigned int Baudrate, unsigned char DataBits, unsigned char StopBits, unsigned char Parity, unsigned char Handshake, bool SetDTR, bool SetRTS, bool UseCTS, bool UseDSR)

Beschreibung

ComInit() initialisiert die serielle Schnittstelle mit ihren Betriebsparametern (Baudrate, Startbits, Stopbits, Parity). Um eine verlustfreie Kommunikation mit dem Kommunikationspartner zu ermöglichen, kann außerdem ein "Handshake" angegeben werden. Für "No Handshake" und für "XON-XOFF Handshake" kann angegeben werden, welche Statusleitungen zusätzlich benutzt werden sollen.

Mit ComInit() können Sie die Kommunikation zurücksetzen falls ein nicht behebbarer Fehler aufgetreten ist.

Parameter

Parameter	C-Typ	Visual Basic Typ	Länge	Beschreibung
Handle	int	ByVal as Long	4 Byte	Gibt an, welcher Kommunikations Port initialisiert werden soll.
Baudrate	unsigned int	ByVal as Long	4 Byte	einzustellende Baudrate: 110 Baud: 110 300 Baud: 300 600 Baud: 600 1200 Baud: 1200 2400 Baud: 2400 4800 Baud: 4800 9600 Baud: 9600 14400 Baud: 14400 19200 Baud: 19200 38400 Baud: 38400 56000 Baud: 56000 57600 Baud: 57600 115200 Baud: 115200

				<p>128000 Baud: 128000 256000 Baud: 256000</p> <p>in C und C++ können Sie die in windows.h definierten "defines" benutzen: CBR_110, CBR_300, CBR_600, CBR_1200, CBR_2400, CBR_4800, CBR_9600, CBR_14400, CBR_19200, CBR_38400, CBR_56000, CBR_57600, CBR_115200, CBR_128000, CBR_256000</p>
DataBits	unsigned char	ByVal as Byte	1 Byte	<p>Anzahl der Datenbits: 5 Datenbits: 5 6 Datenbits: 6 7 Datenbits: 7 8 Datenbits: 8</p>
StopBits	unsigned char	ByVal as Byte	1 Byte	<p>Anzahl der Stopbits: 1 Stopbit: 0 1 ½Stopbits: 1 2 Stopbits: 2</p> <p>in C und C++ können Sie die in windows.h definierten "defines" benutzen: ONESTOPBIT, ONE5STOPBITS, TWOSTOPBITS</p>
Parity	unsigned char	ByVal as Byte	1 Byte	<p>einzustellende Parity Überwachung no Parity: 0 odd Parity: 1 even Parity: 2 mark Parity: 3 space Parity: 4</p> <p>in C und C++ können Sie die in windows.h definierten "defines" benutzen: EVENPARITY, MARKPARITY, NOPARITY, ODDPARITY</p>
Handshake	unsigned char	ByVal as Byte	1 Byte	<p>einzustellender Handshake: kein Handshake: 0 Hardware Handshake: 1 XON-XOFF Handshake: 2</p> <p>Ohne Handshake kann DTR und DSR auf einen bestimmten Pegel gesetzt werden (z. Bsp. um den Kommunikationspartner einzuschalten) CTS und DSR werden ignoriert. Bei Hardware Handshake werden alle 4 Handshakeleitungen grundsätzlich benutzt. DTR und RTS wechseln nach "low" wenn der Empfangspuffer mindstens zu ¾voll ist. Sie wechseln nach "high" wenn der Empfangspuffer weniger als ½voll ist. Im XON-XOFF Modus wird der Handshake mittels der Zeichen XON und XOFF durchgeführt. Alle 4 Handshakeleitungen können zusätzlich benutzt werden.</p>
SetDTR	bool	ByVal as Long	4 Byte	<p>0: DTR wird bei "no handshake" und bei "XON-XOFF" handshake ausgeschalten. 1: DTR wird bei "no handshake" und bei "XON-XOFF" handshake eingeschalten</p>

SetRTS	bool	ByVal as Long	4 Byte	0: DSR wird bei "no handshake" und bei "XON-XOFF" handshake ausgeschalten. 1: DSR wird bei "no handshake" und bei "XON-XOFF" handshake eingeschalten.
UseCTS	bool	ByVal as Long	4 Byte	0: CTS wird bei "XON-XOFF" handshake nicht benutzt. 1: CTS wird bei "XON-XOFF" handshake benutzt.
UseDSR	bool	ByVal as Long	4 Byte	0: DSR wird bei "XON-XOFF" handshake nicht benutzt. 1: DSR wird bei "XON-XOFF" handshake benutzt.

Rückgabe

C-Typ	Visual Basic Typ	Länge	Beschreibung
int	Long	4 Byte	Im Erfolgsfall wird 0 zurückgegeben. Im Fehlerfall (referenzierter Kanal war nicht geöffnet, Fehler in Parameterübergabe) wird -1 zurückgegeben.

2.4. Senden

Name

int ComTransmitData (int Handle, char *Buffer, int Length, int Timeout)

Beschreibung

ComTransmitData() übergibt einen zu sendenden Datenblock an den ausgewählten Kommunikationskanal. Der Datenblock wird während der Laufzeit von ComTransmitData() nicht umkopiert (Parameterübergabe als Referenz und nicht als Wert). D. h. Sie müssen sicherstellen, daß der Datenblock bis zum Abschluß der Übertragung (kann mit ComIsTransmitterEmpty() überprüft werden) nicht verändert wird.

Die Funktion kehrt sofort zurück. Während die Daten übertragen werden können Sie in Ihrem Programm weiterarbeiten. Das Ende der Übertragung muß mit ComIsTransmitterEmpty() abgefragt werden. Um Deadlocks zu vermeiden können Sie einen Timeout für die Übertragung angeben, nach dem das Senden spätestens abgebrochen wird. Den minimalen Timeout können Sie aus Ihren Kommunikationsparametern, Ihrer Telegrammlänge und der Verarbeitungsgeschwindigkeit Ihres Kommunikationspartners ermitteln. Bsp.: Datenübertragung von 100 Bytes mit 9600 Baud, 8 Datenbits, 1 Stopbit, even Parity an einen Kommunikationspartner, der alle Daten ohne Pause verarbeiten kann:

$$\text{Datengeschwindigkeit} [s / \text{Bit}] = \frac{1}{\text{Baudrate}}$$

$$\text{Anzahl_Datenbits} [\text{Bit}] = \text{Datenbits} + \text{Stopbits} + \text{Paritybit} \cdot \text{Anzahl_Bytes}$$

$$\text{Timeout}_{\min} = \text{Anzahl_Datenbits} \cdot \text{Datengeschwindigkeit}$$

Im oben angeführtem Beispiel führt dies zu folgendem Ergebnis:

$$\text{Datengeschwindigkeit} = \frac{1}{9600} s / \text{Bit}$$

$$\text{Anzahl_Datenbits} = 8 + 1 + 1 \cdot 100 \text{Bit}$$

$$\text{Timeout}_{\min} = 1000 \text{Bit} \cdot \frac{1}{9600} s / \text{Bit} \approx 0,1s$$

Der an ComTransmitData() übergebene Timeout sollte um einiges größer gewählt werden als der minimale Timeout.

Parameter

Parameter	C-Typ	Visual Basic Typ	Länge	Beschreibung
-----------	-------	------------------	-------	--------------

Handle	int	ByVal as Long	4 Byte	Gibt an, welcher Kommunikations Port übertragen soll.
Buffer	char*	ByRev as Byte	4 Byte (Adresse)	Adresse des zu übertragenden Datenfelds (Referenz des Datenfelds). Das Datenfeld darf bis zum Abschluß der Übertragung nicht geändert werden
Length	int	ByVal as Long	4 Byte	Anzahl der zu übertragenden Bytes. Die Anzahl darf 1024 nicht überschreiten.
Timeout	int	ByVal as Long	4 Byte	Maximale Zeit für die Übertragung, nach dieser Zeit wird die Übertragung beendet. 0 = kein Timeout

Rückgabe

C-Typ	Visual Basic Typ	Länge	Beschreibung
int	Long	4 Byte	Im Erfolgsfall wird die Anzahl der während der Aufrufzeit bereits übertragenen Bytes zurückgegeben (0...1024). Im Fehlerfall (referenzierter Kanal war nicht geöffnet, Fehler im Kommunikationstreiber) wird -1 zurückgegeben.

2.5. Empfangen

Name

int ComReceiveData (int Handle, char *Buffer, int Length, int Timeout)

Beschreibung

ComReceiveData() teilt dem Kommunikationskanal mit, daß er bis zu 1024 Byte aus dem "Receive-Buffer" des Kommunikationstreibers entnehmen darf (daß können bereits eingetroffene oder erst vom Kommunikationspartner kommende Daten sein). ComReceiveData() benötigt eine Referenz auf einen ausreichend großen Speicherbereich für die Ablage der empfangenen Daten. Dieser Speicherbereich darf bis zum Abschluß der Übertragung nicht verändert werden.

Die Funktion wartet nicht auf das Ende der Übertragung sonder kehrt sofort zurück.

Der Datenblock wird während der Laufzeit von ComTransmitData() nicht umkopiert (Parameterübergabe als Referenz und nicht als Wert). D. h. Sie müssen sicherstellen, daß der Datenblock bis zum Abschluß der Übertragung (kann mit ComIsTransmitterEmpty() nicht verändert wird.

Die Funktion kehrt sofort zurück. Während die Daten übertragen werden, können Sie in Ihrem Programm weiterarbeiten. Um Deadlocks zu vermeiden können Sie einen Timeout für die Übertragung angeben, nach dem das Einlesen der Daten spätestens abgebrochen wird. Das Ende der Übertragung muß mit ComIsReceiverReady() abgefragt werden. Die Bestimmung des Timeouts finden Sie bei "Senden".

Parameter

Parameter	C-Typ	Visual Basic Typ	Länge	Beschreibung
Handle	int	ByVal as Long	4 Byte	Gibt an, welcher Kommunikations Port Daten empfangen soll.
Buffer	char*	ByRev as Byte	4 Byte (Adresse)	Adresse des Datenfelds für die zu empfangenden Daten (Referenz des Datenfelds). Das Datenfeld muß bis zum Abschluß der Übertragung verfügbar sein
Length	int	ByVal as Long	4 Byte	Anzahl der zu empfangenden Bytes. Die Anzahl darf 1024 nicht überschreiten.
Timeout	int	ByVal as Long	4 Byte	Maximale Zeit für die Übertragung, nach dieser Zeit wird das Empfangen beendet. 0 = kein Timeout

Rückgabe

C-Typ	Visual Basic Typ	Länge	Beschreibung
-------	------------------	-------	--------------

int	Long	4 Byte	Wird der Empfangsauftrag bereits während der Aufrufzeit abgearbeitet wird die Anzahl der empfangenen Daten zurückgegeben, ansonsten 0. Im Fehlerfall (referenzierter Kanal war nicht geöffnet, Fehler im Kommunikationstreiber) wird -1 zurückgegeben.
-----	------	--------	---

2.6. Sendestatus

Name

int ComIsTransmitterEmpty (int Handle)

Beschreibung

Mit ComIsTransmitterEmpty() kann man überprüfen ob die letzte Datenübertragung abgeschlossen ist (Ausgabe aller Daten oder Timeout). Die Anzahl der übertragenen Daten wird zurückgegeben.

Parameter

Parameter	C-Typ	Visual Basic Typ	Länge	Beschreibung
Handle	int	ByVal as Long	4 Byte	Gibt an, welcher Kommunikations Port abgefragt werden soll.

Rückgabe

C-Typ	Visual Basic Typ	Länge	Beschreibung
int	Long	4 Byte	Nach Abschluß der Übertragung (Ausgabe aller Daten oder Timeout) wird die Anzahl der übertragenen Daten zurückgegeben. Ist der Sendeauftrag noch nicht abgeschlossen wird -1 zurückgegeben. Der Rückgabewert bleibt gleich bis ein neuer Sendeauftrag ansteht.

2.7. Empfangsstatus

Name

int ComIsReceiverReady (int Handle)

Beschreibung

Mit ComIsReceiverReady() kann man überprüfen ob der letzte Leseauftrag abgeschlossen ist (Einlesen der gewünschten Anzahl von Daten oder Timeout). Die Anzahl der eingelesenen Daten wird zurückgegeben

Parameter

Parameter	C-Typ	Visual Basic Typ	Länge	Beschreibung
Handle	int	ByVal as Long	4 Byte	Gibt an, welcher Kommunikations Port abgefragt werden soll.

Rückgabe

C-Typ	Visual Basic Typ	Länge	Beschreibung
int	Long	4 Byte	Nach Abschluß des Leseauftrags (Einlesen aller Daten oder Timeout) wird die Anzahl der übertragenen Daten zurückgegeben. Ist der Leseauftrag noch nicht abgeschlossen wird -1 zurückgegeben. Der Rückgabewert bleibt gleich bis ein neuer Leseauftrag ansteht.

3. Beispielprogramme

Es sind jeweils ein Beispielprogramm für C++ und für Visual Basic beigelegt. Die Demo installiert einen seriellen Kanal und sendet einen eingegebenen Text auf Kommando ab. Gleichzeitig erwartet die Demo den Empfang des gleichen Textes auf dem gleichen Kanal. Der empfangene Text wird angezeigt.

Für den korrekten Ablauf der Demo müssen sie Pin 2 und Pin 3 einer seriellen Schnittstelle miteinander verbinden.

Das C++ Programm benutzt Com 1 als seriellen Kanal, im Visual Basic Programm lässt sich die Kommunikationsschnittstelle von Com 1 bis Com 4 einstellen.

4. Kontaktadresse

Wir hoffen SERASY32 stellt Sie zufrieden. Sollten Sie Probleme mit SERASY32 haben oder sollte Ihre Aufgabenstellung sich mit SERASY32 nicht abdecken lassen, sprechen Sie uns bitte an.

Kontaktadresse:

Bernhardt PC-Systeme GmbH
 93342 Saal / Donau
 Enzianstraße 16
 Tel.: (0 94 41) 8 18 59
 Fax.: (0 94 41) 8 15 41
 Internet: <http://www.bernhardt.de>

Anhang A C++ Beispiel

Das folgende Beispiel wurde mit Borland C++ 5.0 und OWL erstellt:

C-File:

```
//-----
// Project serasytest
//
// Copyright © 1996. Alle Rechte vorbehalten.
//
// SUBSYSTEM: serasytest Application
// FILE: serasytestdlgclient.cpp
// AUTHOR:
//
// OVERVIEW
// ~~~~~
// Source file for implementation of TserasytestDlgClient (TDialog).
//
//-----

#include <owl/pch.h>

#include "serasytestapp.h"
#include "serasytestdlgclient.h"
#include "serasy.h"

//
// Build a response table for all messages/commands handled by the application.
//
DEFINE_RESPONSE_TABLE1(TserasytestDlgClient, TDialog)
//{{TserasytestDlgClientRSP_TBL_BEGIN}}
EV_BN_CLICKED(IDC_SENDEN, BNClicked),
EV_WM_TIMER,
```

```

//{{ TserasytestDlgClientRSP_TBL_END}}
END_RESPONSE_TABLE;

//{{ TserasytestDlgClient Implementation}}

//-----
// TserasytestDlgClient
// ~~~~~
// Construction/Destruction handling.
//
static TserasytestDlgClientXfer TserasytestDlgClientData;

TserasytestDlgClient::TserasytestDlgClient(TWindow* parent, TResId resId, TModule* module)
:
  TDialog(parent, resId, module)
{
//{{ TserasytestDlgClientXFER_USE}}
  FeldEmpfangen = new TEdit(this, IDC_EMPFANGEN_TEXT, 255);
  FeldSenden = new TEdit(this, IDC_SENDEN_TEXT, 255);
  ButtonSenden = new TButton(this, IDC_SENDEN);

  SetTransferBuffer(&TserasytestDlgClientData);
//{{ TserasytestDlgClientXFER_USE_END}}

  // INSERT>> Your constructor code here.
  int Ret;

  bReceive=false;
  ComHandle=-1;
  ComHandle=ComOpen(1);
  if(ComHandle>=0)
    Ret=ComInit(ComHandle,CBR_600,8,ONESTOPBIT,EVENPARITY,NO_HANDSHAKE,0,0,0,0);
}

TserasytestDlgClient::~TserasytestDlgClient()
{
  Destroy();

  // INSERT>> Your destructor code here.
  KillTimer(1);
  ComClose(ComHandle);
}

void TserasytestDlgClient::BNClicked()
{
  // INSERT>> Your code here.

  FeldSenden->GetText(SendBuffer,256);
  if(ComIsTransmitterEmpty(ComHandle)<0)
  {
    ComTransmitData(ComHandle,SendBuffer,strlen(SendBuffer),1000);
    ComReceiveData(ComHandle,ReceiveBuffer,strlen(SendBuffer),1000);
    bReceive=true;
    ctr=0;
  }
}

```

```

void TserasytestDlgClient::EvTimer(uint timerId)
{
    TDialog::EvTimer(timerId);

    // INSERT>> Your code here.
    if(bReceive)
    {
        ctr++;
        if(ComIsReceiverReady(ComHandle)==strlen(SendBuffer))
        {
            bReceive=false;
            FeldEmpfangen->SetText(ReceiveBuffer);
        }
    }
}

```

```

void TserasytestDlgClient::SetupWindow()
{
    TDialog::SetupWindow();

    // INSERT>> Your code here.
    SetTimer(1,250,NULL);
}

```

h-File:

```

//-----
// Project serasytest
//
// Copyright © 1996. Alle Rechte vorbehalten.
//
// SUBSYSTEM: serasytest Application
// FILE: serasytestdlgclient.h
// AUTHOR:
//
// OVERVIEW
// ~~~~~
// Class definition for TserasytestDlgClient (TDialog).
//
//-----
#if !defined(serasytestdlgclient_h) // Sentry, use file only if it's not already included.
#define serasytestdlgclient_h

#include "serasytestapp.rh" // Definition of all resources.

#include <owl/commctrl.h>

#include <owl/button.h>
#include <owl/edit.h>
//{{ TDialog = TserasytestDlgClient }}
struct TserasytestDlgClientXfer {
//{{ TserasytestDlgClientXFER_DATA }}
    char FeldEmpfangen[ 255 ];
    char FeldSenden[ 255 ];
}

```

```

//{{ TserasytestDlgClientXFER_DATA_END }}
};

class TserasytestDlgClient : public TDialog {
public:
    TserasytestDlgClient(TWindow* parent, TResId resId = IDD_CLIENT, TModule* module = 0);
    virtual ~TserasytestDlgClient();

protected:
    int ComHandle;
    char ReceiveBuffer[256];
    char SendBuffer[256];
    bool bReceive;
    int ctr;
//{{ TserasytestDlgClientXFER_DEF }}
protected:
    TEdit* FeldEmpfangen;
    TEdit* FeldSenden;
    TButton* ButtonSenden;

//{{ TserasytestDlgClientXFER_DEF_END }}

//{{ TserasytestDlgClientVIRTUAL_BEGIN }}
public:
    virtual void SetupWindow();
//{{ TserasytestDlgClientVIRTUAL_END }}

//{{ TserasytestDlgClientRSP_TBL_BEGIN }}
protected:
    void BNClicked();
    void EvTimer(uint timerId);
//{{ TserasytestDlgClientRSP_TBL_END }}
DECLARE_RESPONSE_TABLE(TserasytestDlgClient);
}; //{{ TserasytestDlgClient }}

#endif // serasytestdlgclient_h sentry.

```

Anhang B Visual Basic Beispiel

```

Private Declare Function ComOpen Lib "serasy32.dll" (ByVal Port As Long) As Long
Private Declare Function ComClose Lib "serasy32.dll" (ByVal Handle As Long) As Long
Private Declare Function ComInit Lib "serasy32.dll" (ByVal Handle As Long, _
    ByVal Baudrate As Long, _
    ByVal DataBits As Byte, _
    ByVal StopBits As Byte, _
    ByVal Parity As Byte, _
    ByVal Handshake As Byte, _
    ByVal SetDTR As Long, _
    ByVal SetRTS As Long, _
    ByVal UseCTS As Long, _
    ByVal UseDSR As Long) As Long

Private Declare Function ComTransmitData Lib "serasy32.dll" (ByVal Handle As Long, _
    ByVal Buffer As Byte, _
    ByVal Length As Long, _
    ByVal Timeout As Long) As Long
Private Declare Function ComReceiveData Lib "serasy32.dll" (ByVal Handle As Long, _

```

```
ByRef Buffer As Byte, _  
ByVal Length As Long, _  
ByVal Timeout As Long) As Long
```

```
Private Declare Function ComIsTransmitterEmpty Lib "serasy32.dll" (ByVal Handle As Long) As Long  
Private Declare Function ComIsReceiverReady Lib "serasy32.dll" (ByVal Handle As Long) As Long
```

```
Dim InBuffer(128) As Byte  
Dim OutBuffer(128) As Byte  
Dim Length As Long  
Dim bReceive As Boolean  
Dim bInitOk As Boolean  
Dim ComHandle As Long
```

```
Const VB_BR_110 As Long = 110  
Const VB_BR_300 As Long = 300  
Const VB_BR_600 As Long = 600  
Const VB_BR_1200 As Long = 1200  
Const VB_BR_2400 As Long = 2400  
Const VB_BR_4800 As Long = 4800  
Const VB_BR_9600 As Long = 9600  
Const VB_BR_14400 As Long = 14400  
Const VB_BR_19200 As Long = 19200  
Const VB_BR_38400 As Long = 38400  
Const VB_BR_56000 As Long = 56000  
Const VB_BR_57600 As Long = 57600  
Const VB_BR_115200 As Long = 115200  
Const VB_BR_128000 As Long = 128000  
Const VB_BR_256000 As Long = 256000
```

```
Const VB_5_DATABITS As Byte = 5  
Const VB_6_DATABITS As Byte = 6  
Const VB_7_DATABITS As Byte = 7  
Const VB_8_DATABITS As Byte = 8
```

```
Const VB_ONE_STOPBIT As Byte = 0  
Const VB_ONE5_STOPBITS As Byte = 1  
Const VB_TWO_STOPBITS As Byte = 2
```

```
Const VB_NO_PARITY As Byte = 0  
Const VB_ODD_PARITY As Byte = 1  
Const VB_EVEN_PARITY As Byte = 2  
Const VB_MARK_PARITY As Byte = 3  
Const VB_SPACE_PARITY As Byte = 4
```

```
Const VB_NO_HANDSHAKE As Byte = 0  
Const VB_HARDWARE_HANDSHAKE As Byte = 1  
Const VB_XON_XOFF_HANDSHAKE As Byte = 2
```

```
Private Sub Form_Load()  
    ComHandle = -1  
    bReceive = False  
    bInitOk = False  
    InitCom (1)  
    Timer1.Interval = 100 ' Set Timer interval to 100ms  
End Sub
```

```
Private Sub Form_Terminate()  
    result = ComClose(ComHandle)
```

End Sub

```
Private Sub Option1_Click()  
If Option1.Value = True Then  
    InitCom (1)  
End If  
End Sub
```

```
Private Sub Option2_Click()  
If Option2.Value = True Then  
    InitCom (2)  
End If  
End Sub
```

```
Private Sub Option3_Click()  
If Option3.Value = True Then  
    InitCom (3)  
End If  
End Sub
```

```
Private Sub Option4_Click()  
If Option4.Value = True Then  
    InitCom (4)  
End If  
End Sub
```

```
Private Sub Command1_Click()  
    Length = Len(Text1.Text)  
    For i = 1 To Length  
        OutBuffer(i - 1) = Asc(Mid(Text1.Text, i, 1))  
    Next i  
    If bInitOk = True Then  
        result = ComTransmitData(ComHandle, OutBuffer(0), Length, 10000)  
        result = ComReceiveData(ComHandle, InBuffer(0), Length, 10000)  
        bReceive = True  
    End If  
End Sub
```

```
Private Sub Timer1_Timer()  
If bReceive = True Then  
    result = ComIsReceiverReady(ComHandle)  
    If result >= 0 Then  
        bReceive = False  
        Text2.Text = ""  
        For i = 1 To Length  
            Text2.Text = Text2.Text + Chr(InBuffer(i - 1))  
        Next i  
    End If  
End If  
End Sub
```

```
Private Sub InitCom(PortNo)  
    If ComHandle >= 0 Then  
        ComClose (ComHandle)  
    End If  
    ComHandle = ComOpen(PortNo)  
    If (ComHandle < 0) Then  
        result = MsgBox("Serielle Schnittstelle nicht vorhanden oder wird bereits benutzt!", vbOKOnly +  
vbCritical, "Fehler", 0, 0)
```

```
    bInitOk = False
Else
    result = ComInit(ComHandle, VB_BR_9600, VB_8_DATABITS, VB_ONE_STOPBIT,
VB_EVEN_PARITY, VB_NO_HANDSHAKE, 0, 0, 0, 0)
    If result >= 0 Then
        bInitOk = True
    End If
End If
End Sub
```