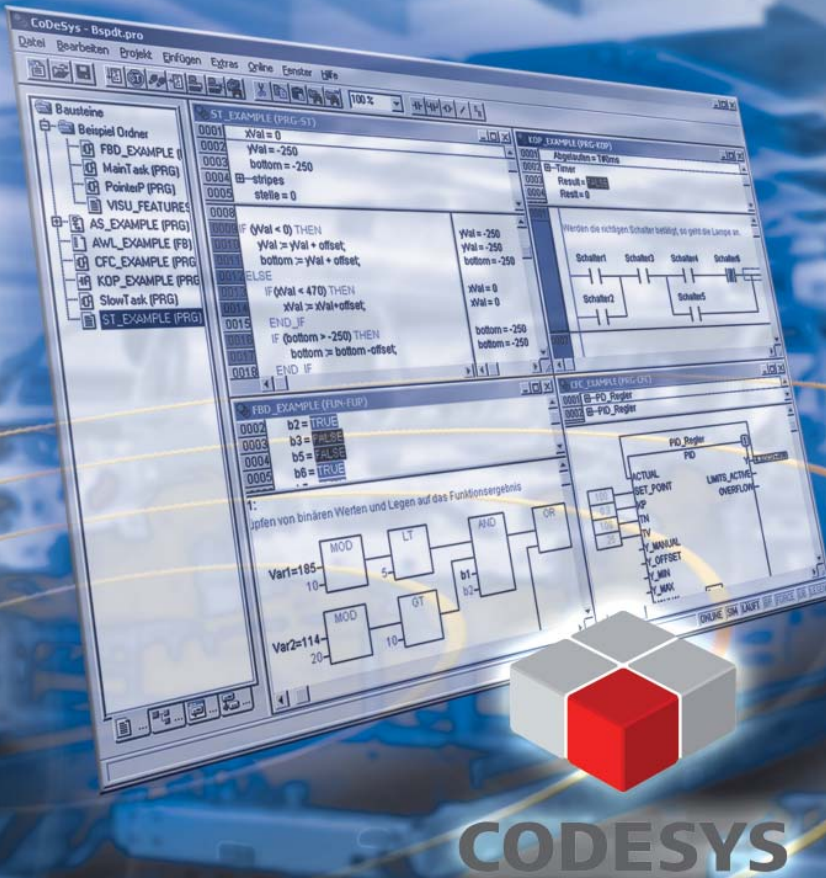


TURCK

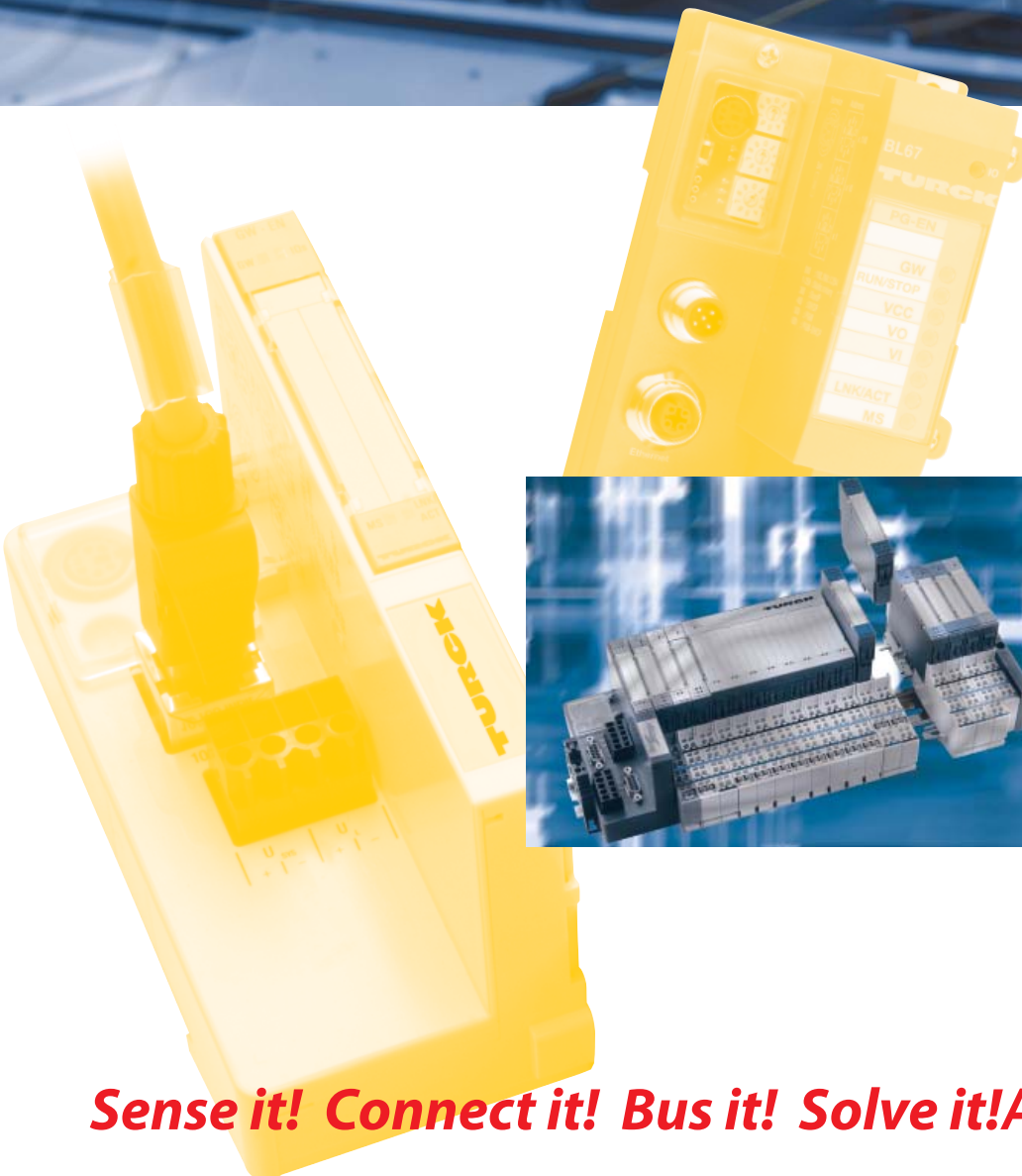
Industrielle Automation

APPLICATION NOTE

BLxx- FUNKTIONS- BAUSTEINE FÜR CODESYS



CODESYS



Sense it! Connect it! Bus it! Solve it! APPLICATION

Alle Marken- und Produktnamen sind Warenzeichen oder eingetragene Warenzeichen der jeweiligen Titelfalter.

Ausgabe 10/2012

© Hans Turck GmbH, Mülheim an der Ruhr

Alle Rechte, auch die der Übersetzung, vorbehalten.

Kein Teil dieses Handbuches darf in irgendeiner Form (Druck, Fotokopie, Mikrofilm oder einem anderen Verfahren) ohne schriftliche Zustimmung der Firma Hans Turck GmbH & Co. KG, Mülheim an der Ruhr reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Änderungen vorbehalten.

Inhaltsverzeichnis

1	CoDeSys - Funktionsbausteine für programmierbare Gateways	
1.1	Installation	1-2
1.2	Allgemeines	1-3
1.3	BL20-Funktionsbausteine.....	1-4
1.3.1	BL20-1CNT-24VDC-Modul.....	1-4
1.4	BL20/BL67-Funktionsbausteine.....	1-8
1.4.1	BLxx-1RS232- und BLxx-1RS485/422-Modul.....	1-8
1.4.2	BLxx-1SSI-Modul.....	1-11
2	Anwendungsbeispiel des BLxx_1RSxxx_FB mit HyperTerminal	
2.1	Allgemeines	2-2
2.1.1	Windows HyperTerminal.....	2-3
2.2	Einstellen der Kommunikationsparameter.....	2-4
2.2.1	Einstellen der Modulparameter in CoDeSys.....	2-4
2.2.2	Einstellen der Eigenschaften im HyperTerminal.....	2-5
2.3	CoDeSys - Aufruf des FBs und Variablendeklaration	2-6
2.4	Das Senden von Daten (Modul → HyperTerminal)	2-8
2.5	Das Empfangen von Daten (HyperTerminal → Modul)	2-9

1 CoDeSys - Funktionsbausteine für programmierbare Gateways

1.1	Installation.....	2
1.2	Allgemeines	3
1.3	BL20-Funktionsbausteine.....	4
1.3.1	BL20-1CNT-24VDC-Modul.....	4
	– Beschreibung der Variablen.....	4
	– Beispiel für eine Steuerungskonfiguration des Moduls BL20-1CNT-24VDC.....	7
1.4	BL20/BL67-Funktionsbausteine	8
1.4.1	BLxx-1RS232- und BLxx-1RS485/422-Modul.....	8
	– Aufbau des Funktionsbausteins	8
	– Beschreibung der Variablen.....	8
	– Beispiel für eine Steuerungskonfiguration des Moduls BLxx-1RSxxx.....	10
1.4.2	BLxx-1SSI-Modul.....	11
	– Aufbau des Funktionsbausteins	11
	– Beschreibung der Variablen.....	11
	– Beispiel für eine Steuerungskonfiguration des Moduls BLxx-1SSI.....	14

1.1 Installation

Die CoDeSys Funktionsbausteine für die programmierbaren Gateway sind in der TURCK-spezifischen CoDeSys-Bibliothek „BLxx_PG_FB.lib“ enthalten.

Diese *.lib ist Bestandteil der BLxx-Target-Dateien und wird bei deren Installation automatisch mit installiert.

Zu finden ist die Datei dann, je nach Installationspfad der CoDeSys unter:

x:\Programme\Gemeinsame Dateien\CAA-Targets\Turck\BLxx

Sollte die Datei nicht in diesem Verzeichnis zu finden sein, kann sie von der TURCK Homepage www.turck.com heruntergeladen und manuell nachinstalliert werden.

1.2 Allgemeines

Die Gateways BL20-PG-xx und BL67-PG-xx unterstützen am lokalen Modulbus Technologiemodule, die über die Prozessdaten Steuer- und Kontrollbits für den Datenaustausch bereit stellen.

Um Funktionen bzw. Aktionen dieser Module nutzen zu können, müssen dazu in den Anwenderprogrammen sogenannte „Handshake“-Mechanismen ausprogrammiert werden.

Nachfolgend werden Funktionen beschrieben die diese Hantierung übernehmen. Diese sind in der Bibliothek BLxx_PG_FB.lib enthalten. Diese Bibliothek ist in zwei Unterverzeichnisse unterteilt:

- 1** BL20_PG_FB
mit Funktionsbaustein BL20_1CNT_FB für das Modul BL20-1CNT
- 2** BLxx_PG_FB
mit Funktionsbausteine BLxx_1RSxx_FB für die Module BLxx-1RS232 bzw. BLxx-1RS485/422 und BLxx_1SSI_FB für das Modul BLxx-1SSI

Anmerkung:

Den Variablennamen wird ein Präfix hinzugefügt. Dieses Präfix wird entsprechend der Empfehlungen der Spezifikation IEC 61131 und der Fa. 3S – Smart Software Solutions GmbH verwendet.

Anhand dieses Präfixes kann der Anwender den Datentyp einer Variablen erkennen.

Beispiel:

xVarName = Datentyp BOOLEAN

bVarName = Datentyp BYTE usw.

1.3 BL20-Funktionsbausteine

In diesem Kapitel werden Funktionsbausteine beschrieben, die nur für BL20-Module gültig sind.

1.3.1 BL20-1CNT-24VDC-Modul

Der Baustein BL20_1CNT_FB wird zum Hantieren der Daten des Moduls BL20-1CNT-24VDC in der Zähl- oder Messbetriebsart des Moduls eingesetzt.

Dazu benötigt der Funktionsbaustein die Anfangsadressen der Prozessein- und Prozessausgangsdaten des Zählermoduls. Damit wird zum einen der aktuelle Zähl-/Messwert angezeigt und zum anderen können die Modulfunktionen, die im Handbuch „BL20- I/O-Module“ ([D300716](#)) beschrieben sind, gesteuert werden.

Abbildung 1-1:
BL20_1CNT_FB

BL20_1CNT_FB	
ptCNTInput : POINTER TO ARRAY [0..7] OF BYTE	diEncoderValue : DINT
ptCNTOutput : POINTER TO ARRAY [0..7] OF BYTE	xERR_24VDC : BOOL
xCountOrMeasure : BOOL	xERR_DO : BOOL
xSW_GATE : BOOL	xERR_PARA : BOOL
xCTRL_SYN : BOOL	xERR_LOAD : BOOL
xCTRL_D01 : BOOL	xRES_STS_A : BOOL
xSET_D01 : BOOL	xSTS_LOAD : BOOL
xCTRL_D02 : BOOL	xSTS_GATE : BOOL
xSET_D02 : BOOL	xSTS_DI : BOOL
xRES_STS : BOOL	xSTS_D01 : BOOL
xEXTF_ACK : BOOL	xSTS_D02 : BOOL
diLOAD_VAL : DINT	xSTS_C_UP : BOOL
xLOAD_VAL : BOOL	xSTS_C_DN : BOOL
diLOAD_PREPARE : DINT	xSTS_SYN : BOOL
xLOAD_PREPARE : BOOL	xSTS_CMP1 : BOOL
diCMP_VAL1 : DINT	xSTS_CMP2 : BOOL
xLOAD_CMP_VAL1 : BOOL	xSTS_OFLW : BOOL
diCMP_VAL2 : DINT	xSTS_UFLW : BOOL
xLOAD_CMP_VAL2 : BOOL	xSTS_ND : BOOL
udiVAL_INTTIME : UDINT	wRetVal : WORD
xLOAD_INTTIME : BOOL	
udiVAL_LOLIMIT : UDINT	
xLOAD_LOLIMIT : BOOL	
udiVAL_HILIMIT : UDINT	
xLOAD_HILIMIT : BOOL	
udiVAL_DO_PARAM : UDINT	
xLOAD_DO_PARAM : BOOL	

Beschreibung der Variablen

Für interne Zwecke werden die Prozesseingangs- bzw. -ausgangsdaten in den Datentyp BYTE umgewandelt. Deshalb ist die Variable „ptCNTInput“ bzw. „ptCNTOutput“ vom Datentyp POINTER.

Tabelle 1-1:
Variablenbe-
schreibung
BL20_1CNT_FB

Variable	Typ	Bedeutung
ptCNTInput	POINTER TO ARRAY [0..7] OF BYTE	Zeiger auf die Prozesseingangsworte des Zählermoduls z.B. ADR(%IW4) oder ADR(CNT_IN) → Beispiel für eine Steuerungskonfiguration des Moduls BL20-1CNT-24VDC (Seite 1-7) .

Tabelle 1-1:
Variablenbe-
schreibung
BL20_1CNT_FB

Variable	Typ	Bedeutung
ptCNTOutput	POINTER TO ARRAY [0..7] OF BYTE	Zeiger auf die Prozessausgangsworte des Zählermoduls z.B. ADR(%QW4) oder ADR(CNT_OUT). → Beispiel für eine Steuerungskonfiguration des Moduls BL20-1CNT-24VDC (Seite 1-7) .
xCountOrMeasure	BOOL	Betriebsart des Moduls, wie in der Steuerungskonfiguration gewählt wurde: 0 = Zählbetriebsart, 1 = Messbetriebsart
xSW_GATE	BOOL	Softwarefreigabe Zählen bzw. Messen
xCTRL_SYN	BOOL	Freigabe Synchronisation
xCTRL_DO1	BOOL	Freigabe Ausgang DO1
xSET_DO1	BOOL	Steuerbit Ausgang DO1
xCTRL_DO2	BOOL	Freigabe Ausgang DO2
xSET_DO2	BOOL	Steuerbit Ausgang DO2
xRES_STS	BOOL	Rücksetzen Statusbits: 0 → 1 Start Rücksetzen
xEXTF_ACK	BOOL	Quittierung Diagnosefehler
diLOAD_VAL	DINT	für Zählbetrieb: Wert für „Ladewert direkt“
xLOAD_VAL	BOOL	für Zählbetrieb: „Ladewert direkt“ laden
diLOAD_PREPARE	DINT	für Zählbetrieb: Wert für „Ladewert vorbereitend“
xLOAD_PREPARE	BOOL	für Zählbetrieb: „Ladewert vorbereitend“ laden
diCMP_VAL1	DINT	für Zählbetrieb: Wert für „Vergleichswert 1“
xLOAD_CMP_VAL1	BOOL	für Zählbetrieb: „Vergleichswert 1“ laden
diCMP_VAL2	DINT	für Zählbetrieb: Wert für „Vergleichswert 2“
xLOAD_CMP_VAL2	BOOL	für Zählbetrieb: „Vergleichswert 2“ laden
udiVAL_INTTIME	UDINT	für Messbetrieb: Wert für „Integrationszeit“
xLOAD_INTTIME	BOOL	für Messbetrieb: „Integrationszeit“ laden
udiVAL_LOLIMIT	UDINT	für Messbetrieb: Wert für „untere Grenze“
xLOAD_LOLIMIT	BOOL	für Messbetrieb: „untere Grenze“ laden
udiVAL_HILIMIT	UDINT	für Messbetrieb: Wert für „obere Grenze“
xLOAD_HILIMIT	BOOL	für Messbetrieb: „obere Grenze“ laden
udiVAL_DO_PARAM	UDINT	Funktion und Verhalten von Ausgang DO1 und DO2

*Tabelle 1-1:
Variablenbe-
schreibung
BL20_1CNT_FB*

Variable	Typ	Bedeutung
xLOAD_DO_PARAM	BOOL	Funktion und Verhalten von Ausgang DO1 und DO2 ändern
xSTS_LOAD	BOOL	Funktion und Verhalten von Ausgang DO1 und DO2 laden
diEncoderValue	DINT	Zählwert
xERR_24VDC	BOOL	Fehlerbit Kurzschluss Geber oder Fehler Spannungsversorgung
xERR_DO	BOOL	Fehlerbit Kurzschluss Ausgang DO1
xERR_PARA	BOOL	Fehlerbit Parametrierung
xERR_LOAD	BOOL	Fehlerbit Ladevorgang
xRES_STS_A	BOOL	Rücksetzen Statusbits läuft
xSTS_LOAD	BOOL	Ladevorgang läuft
xSTS_GATE	BOOL	Status Freigabe Zählermodul
xSTS_DI	BOOL	Status Hardwareeingang
xSTS_DO1	BOOL	Status Hardwareausgang DO1
xSTS_DO2	BOOL	Status Softwareausgang DO2
xSTS_C_UP	BOOL	Status Zählrichtung vorwärts
xSTS_C_DN	BOOL	Status Zählrichtung rückwärts
xSTS_SYN	BOOL	Status Synchronisation
xSTS_CMP1	BOOL	Status Vergleich 1
xSTS_CMP2	BOOL	Status Vergleich 2
xSTS_OFLW	BOOL	Status Überlauf
xSTS_UFLW	BOOL	Status Unterlauf
xSTS_ND	BOOL	Status Nulldurchgang
wRetVal	WORD	Rückgabewert: Wert > 8000h → Fehler
	– 0x8101:	Größe Array der Eingangsdaten ≠ 8 Bytes → Abbruch des FBs
	– 0x8103	Größe Array der Ausgangsdaten ≠ 8 Bytes → Abbruch des FBs

Beispiel für eine Steuerungskonfiguration des Moduls BL20-1CNT-24VDC

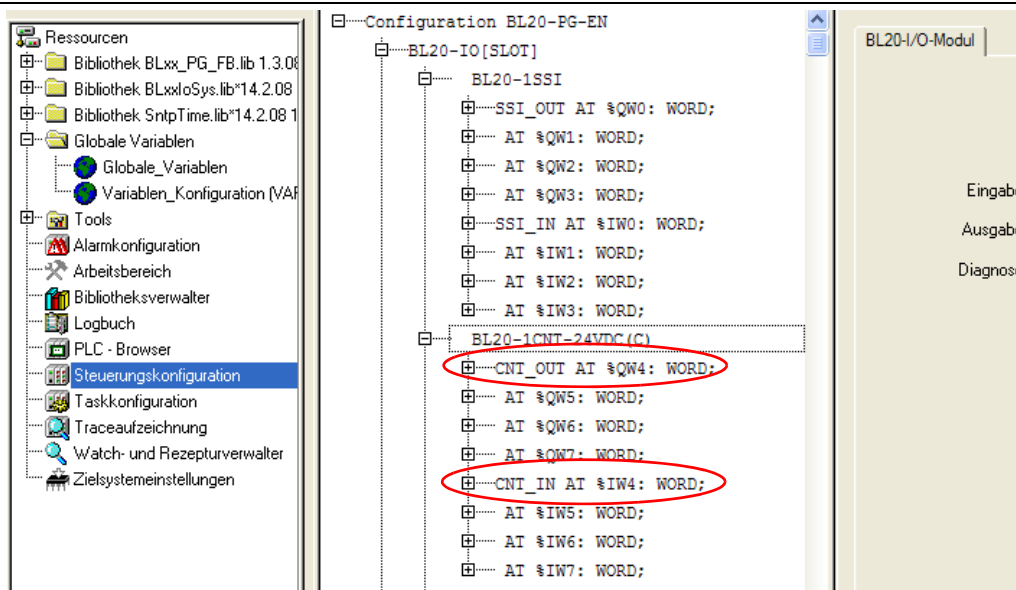
In diesem Beispiel können die Prozesseingangsdaten der Variablen „ptCNTInput“ auf unterschiedliche Weisen zugeordnet werden:

- 1 über ADR(CNT_IN), wenn für die Eingangsadresse ein symbolischer Name vergeben wurde,
- 2 oder direkt als ADR(%IW4)

Dies gilt auch für die Prozessausgangsdaten der Variable „ptCNTOutput“:

- 1 über ADR(CNT_OUT), wenn für die Ausgangsadresse ein symbolischer Name vergeben wurde
- 2 oder direkt als ADR(%QW4).

Abbildung 1-2:
Beispiel



1.4 BL20/BL67-Funktionsbausteine

In diesem Abschnitt werden Funktionsbausteine beschrieben, die sowohl für BL20 - Module als auch BL67 - Module verwendet werden können.

1.4.1 BLxx-1RS232- und BLxx-1RS485/422-Modul

Der Baustein BLxx_1RSxxx_FB kann zum Hantieren der Daten der Schnittstellen-Module (BL20-1RS232, BL20-1RS485/422, BL67-1RS232 und BL67-1RS485/422) eingesetzt werden.

Er unterstützt das gleichzeitige Senden und Empfangen von Daten, d.h. ein Vollduplex - Betrieb, wie ihn z.B. das BLxx-1RS232 ermöglicht, ist damit möglich. Dieser Funktionsbaustein kann unabhängig vom Schnittstellentyp verwendet werden, da nur die Eingangs- bzw. Ausgangsdaten ausgewertet werden.

Die empfangenen Daten werden über die Prozesseingangsdaten (ptRxData) des Moduls vom Funktionsbaustein „abgeholt“ und im Datenablagepuffer (ptRxBuffer) abgelegt. Größe und Ablageort des Ablagepuffers werden vom Anwender bestimmt. Des Weiteren kann der Anwender individuell die Anzahl (uiMaxRxBuffer) der Telegrammbytes bestimmen.

Gleiches gilt für die zu sendenden Daten.

Aufbau des Funktionsbausteins

Abbildung 1-3:

Aufbau des

Bausteins

BLxx_1RSxxx_FB

BLXX_1RSXXX_FB	
ptRxData : POINTER TO ARRAY [0..7] OF BYTE	xBusyRx : BOOL
ptTxData : POINTER TO ARRAY [0..7] OF BYTE	uiReceivedBytes : UINT
xEnableRx : BOOL	xBusyTx : BOOL
xEnableTx : BOOL	uiSentBytes : UINT
xQuit : BOOL	xSendBufNotEmpty : BOOL
xClr_Buf_Rx : BOOL	wRetVal : WORD
xClr_Buf_Tx : BOOL	
xDisableTxBuffer : BOOL	
ptRxBuffer : POINTER TO BYTE	
uiMaxRxBytes : UINT	
ptTxBuffer : POINTER TO BYTE	
uiMaxTxBytes : UINT	

Beschreibung der Variablen

Für interne Zwecke werden die Prozesseingangs- bzw. -ausgangsdaten in den Datentyp BYTE gewandelt. Deshalb ist die Variable „ptRxData“ bzw. „ptTxData“ vom Datentyp POINTER.

Tabelle 1-2:
Variablenbe-
schreibung
BLxx_1RSxxx_FB

Variable	Typ	Bedeutung
ptRxData	POINTER TO ARRAY [0..7] OF BYTE	Zeiger auf die Prozesseingangsworte des RSxxx Moduls z.B. ADR(%IW8) oder ADR(RS232_RX) → Beispiel für eine Steuerungskonfiguration des Moduls BLxx-1RSxxx (Seite 1-10).
ptTxData	POINTER TO ARRAY [0..7] OF BYTE	Zeiger auf die Prozessausgangsworte des RSxxx Moduls z.B. ADR(%QW8) oder ADR(RS232_TX) → Beispiel für eine Steuerungskonfiguration des Moduls BLxx-1RSxxx (Seite 1-10).
xEnableRx	BOOL	Freigabe für Datenempfang
xEnableTx	BOOL	Freigabe zum Datensenden

Tabelle 1-2:
 Variablenbe-
 schreibung
 BLxx_1RSxxx_FB

Variable	Typ	Bedeutung
xQuit	BOOL	Quittierung von Fehlern
xClr_Buf_Rx	BOOL	Löschen des Empfangspuffer: 0 → 1 und Quitt = 1
xClr_Buf_Tx	BOOL	Löschen des Sendepuffers: 0 → 1 und Quitt = 1
xDisableTxBuffer	BOOL	Sperren des Sendepuffers: 0 = freigegeben; 1 = gesperrt
ptRxBuffer	POINTER TO BYTE	Adresse der Empfangsdatenablage in der SPS. Array aus n Elementen vom Datentyp BYTE
uiMaxRxBytes	UINT	Maximale Anzahl der zu empfangenen Datenbytes eines Telegramms. Kann je nach erwarteter Telegrammlänge vor einem neuen Job geändert werden. Hinweis: Muss > 0 sein, sonst werden keine Daten empfangen.
ptTxBuffer	POINTER TO BYTE	Adresse der Sendedatenablage im programmierbaren Gateway. Array aus n Elementen vom Datentyp BYTE
uiMaxTxBytes	UINT	Maximale Anzahl der zu sendenden Datenbytes eines Telegramms. Kann je nach erwarteter Telegrammlänge vor einem neuen Job geändert werden. Hinweis: Muss > 0 sein, sonst werden keine Daten gesendet.
xBusyRx	BOOL	Anzeige Empfang von Daten aktiv
uiReceivedBytes	UINT	Zähler der empfangenen Datenbytes
xBusyTx	BOOL	Anzeige Senden von Daten aktiv
uiSentBytes	UINT	Zähler der gesendeten Datenbytes
xSendBufNotEmpty	BOOL	
wRetVal	WORD	Rückgabewert: Wert > 8000h → Fehler
– Fehler der Größe-Variablen	0x8101	Größe des Empfangsfachs > max. zu empfangenen Bytes → Abbruch
	0x8103	Größe Array der Eingangsdaten ≠ 8 Bytes → Abbruch
	0x8201	Größe des Sendefachs > max. zu sendenden Bytes → Abbruch
	0x8203	Größe Array der Ausgangsdaten ≠ 8 Bytes → Abbruch

Tabelle 1-2:
Variablenbe-
schreibung
BLxx_1RSxxx_FB

Variable	Typ	Bedeutung
– Fehler des Moduls	0x8000	Modul für Kommunikation nicht bereit.
	0x8008	Parametrierfehler des Moduls
	0x8010	Hardwarefehler des Moduls
	0x8020	Fehler Datenflusskontrolle
	0x8040	Rahmenfehler
	0x8080	(Empfangs-)Puffer Überlauf

Beispiel für eine Steuerungskonfiguration des Moduls BLxx-1RSxxx

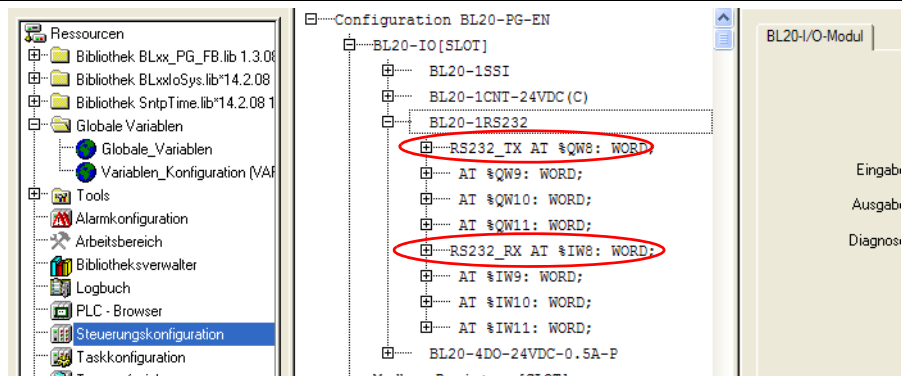
In diesem Beispiel können die Prozesseingangsdaten der Variablen „ptRxData“ auf unterschiedliche Weisen zugeordnet werden:

- 1 über ADR(RS232_RX), wenn für die Eingangsadresse ein symbolischer Name vergeben wurde,
- 2 oder direkt als ADR(%IW8)

Dies gilt auch für die Prozessausgangsdaten der Variable „ptTxData“:

- 1 über ADR(RS232_TX), wenn für die Ausgangsadresse ein symbolischer Name vergeben wurde
- 2 oder direkt als ADR(%QW8).

Abbildung 1-4:
Beispiel



1.4.2 BLxx-1SSI-Modul

Der Baustein BLxx_1SSI_FB wird zum Hantieren der Daten der Module BL20-1SSI und BL67-1SSI eingesetzt.

Aufbau des Funktionsbausteins

Abbildung 1-5:
Aufbau des
Bausteins
BLxx_1SSI_FB

BLXX_1SSI_FB	
ptSSIInput : POINTER TO ARRAY [0..7] OF BYTE	dwRegRdData : DWORD
ptSSIOutput : POINTER TO ARRAY [0..7] OF BYTE	bRegRdAdrStat : BYTE
xStop : BOOL	xRegRdAbort : BOOL
xEnCMP1 : BOOL	xRegWrAckn : BOOL
xClrCMP1 : BOOL	xRegWrAccept : BOOL
xEnCMP2 : BOOL	xStsCMP1 : BOOL
xClrCMP2 : BOOL	xFlagCMP1 : BOOL
diREG_CMP1 : DINT	xRelCMP1 : BOOL
xLOAD_REG_CMP1 : BOOL	xStsCMP2 : BOOL
diREG_CMP2 : DINT	xFlagCMP2 : BOOL
xLOAD_REG_CMP2 : BOOL	xRelCMP2 : BOOL
diREG_LOWER_LIMIT : DINT	xStsDn : BOOL
xLOAD_REG_LOWER_LIMIT : BOOL	xStsUp : BOOL
diREG_UPPER_LIMIT : DINT	xStsOflw : BOOL
xLOAD_REG_UPPER_LIMIT : BOOL	xStsUflw : BOOL
xRegWr : BOOL	xStsStop : BOOL
bRegRdAdr : BYTE	xSSI DIAG : BOOL
bRegWrAdr : BYTE	xSSISts0 : BOOL
diRegWrData : DINT	xSSISts1 : BOOL
	xSSISts2 : BOOL
	xSSISts3 : BOOL
	xErrSSI : BOOL
	xErrPara : BOOL
	wRetVal : WORD

Beschreibung der Variablen

Für interne Zwecke werden die Prozesseingangs- bzw. -ausgangsdaten in den Datentyp BYTE gewandelt. Deshalb ist die Variable „ptSSIInput“ bzw. „ptSSIOutput“ vom Datentyp POINTER.

Tabelle 1-3:
Variablenbe-
schreibung
BLxx_1SSI_FB

Variable	Typ	Bedeutung
ptSSIInput	POINTER TO ARRAY [0..7] OF BYTE	Zeiger auf die Prozesseingangsworte des Moduls BLxx-1SSI z.B. ADR(%IW0) oder ADR(SSI_IN) → Beispiel für eine Steuerungskonfiguration des Moduls BLxx-1SSI (Seite 1-14).
ptSSIOutput	POINTER TO ARRAY [0..7] OF BYTE	Zeiger auf die Prozessausgangsworte des Moduls BLxx-1SSI z.B. ADR(%IW0) oder ADR(SSI_OUT) → Beispiel für eine Steuerungskonfiguration des Moduls BLxx-1SSI (Seite 1-14).
xStop	BOOL	Kommunikationskontrolle: 0 = zyklisches Lesen; 1 = Kommunikation gestoppt
xEnCMP1	BOOL	Freigabe Vergleicher 1
xClrCMP1	BOOL	Vergleichsbit 1 löschen
xEnCMP2	BOOL	Freigabe Vergleicher 2

*Tabelle 1-3:
Variablenbe-
schreibung
BLxx_1SSI_FB*

Variable	Typ	Bedeutung
xClrCMP2	BOOL	Vergleichsbit 2 löschen
diREG_CMP1	DINT	Vergleichwert 1
xLOAD_REG_CMP1	BOOL	Vergleichwert 1 laden
diREG_CMP2	DINT	Vergleichwert 2
xLOAD_REG_CMP2	BOOL	Vergleichwert 2 laden
diREG_LOWER_LIMIT	DINT	Untere Grenze
xLOAD_REG_LOWER_LI MIT	BOOL	Wert Untere Grenze laden
diREG_UPPER_LIMIT	DINT	Obere Grenze
xLOAD_REG_UPPER_ LIMIT	BOOL	Wert Obere Grenze laden
xRegWR	BOOL	Freigabe Schreiben Register: 0 → 1 aktiviert
bRegRdAdr	BYTE	Adresse zum Auslesen des Registers
bRegWrAdr	BYTE	Adresse zum Schreiben in das Register
diRegWrData	DINT	Daten des zu schreibenden Registers
dwRegRdData	DWORD	Daten des ausgelesenen Registers
bRegRdAdrStat	BYTE	Rückmeldung des gelesenen Registers
xRegRdAbort	BOOL	Abbruch Registerlesen
xRegWrAkn	BOOL	Rückmeldung Register schreiben läuft
xRegWrAccept	BOOL	Rückmeldung Register schreiben O.K.
xStsCMP1	BOOL	Statusbit COMP1: 1 = RegSSIPos = RegCMP1; 0 = RegSSIPos ≠ RegCMP1
xFlagCMP1	BOOL	Statusbit COMP1 (latch): 1 = RegSSIPos = RegCMP1; 0 = RegSSIPos ≠ RegCMP1
xRelCMP1	BOOL	Statusbit COMP1: 1 = RegSSIPos ≥ RegCMP1; 0 = RegSSIPos < RegCMP1
xStsCMP2	BOOL	Statusbit COMP2 1 = RegSSIPos = RegCMP2; 0 = RegSSIPos ≠ RegCMP2

Tabelle 1-3:
Variablenbe-
schreibung
BLxx_1SSI_FB

Variable	Typ	Bedeutung
xFlagCMP2	BOOL	Statusbit COMP2 (latch): 1 = RegSSIPos = RegCMP2; 0 = RegSSIPos ≠ RegCMP2
xRelCMP2	BOOL	Statusbit COMP2 1 = RegSSIPos ≥ RegCMP2; 0 = RegSSIPos < RegCMP2
xStsDN	BOOL	Status Zählrichtung rückwärts
xStsOfw	BOOL	Status Überlauf
xStsUflw	BOOL	Status Unterlauf
xStsStop	BOOL	Status Kommunikation
xSSIDdiag	BOOL	Anzeige Diagnose vorhanden
xSSISts0	BOOL	Diagnosebit 0
xSSISts1	BOOL	Diagnosebit 1
xSSISts2	BOOL	Diagnosebit 2
xSSISts3	BOOL	Diagnosebit 3
xERR_SSI	BOOL	Status Encoder Signal: 1 = Fehler (Kabelbruch) 0 = O.K.
xERR_PARA	BOOL	Status Parametrierung: 1 = fehlerhaft 0 = O.K.
wRetVal	WORD	Rückgabewert: Wert > 8000h Fehler
	– 0x8101:	Größe Array der Eingangsdaten ≠ 8 Bytes → Abbruch
	– 0x8103	Größe Array der Ausgangsdaten ≠ 8 Bytes → Abbruch

Beispiel für eine Steuerungskonfiguration des Moduls BLxx-1SSI

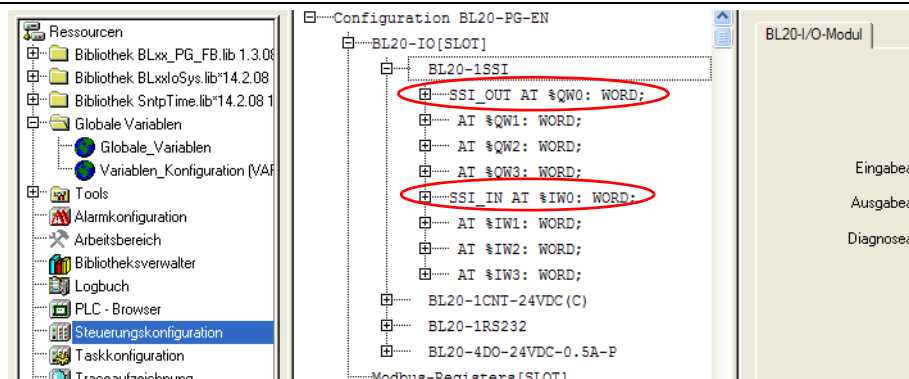
In diesem Beispiel können die Prozesseingangsdaten der Variablen „ptSSIInput“ auf unterschiedliche Weisen zugeordnet werden:

- 1 über ADR(SSI_IN), wenn für die Eingangsadresse ein symbolischer Name vergeben wurde,
- 2 oder direkt als ADR(%IW0)

Dies gilt auch für die Prozessausgangsdaten der Variable „ptSSI_Output“:

- 1 über ADR(SSI_OUT), wenn für die Ausgangsadresse ein symbolischer Name vergeben wurde
- 2 oder direkt als ADR(%QW0).

Abbildung 1-6:
Beispiel



2 Anwendungsbeispiel des BLxx_1RSxxx_FB mit HyperTerminal

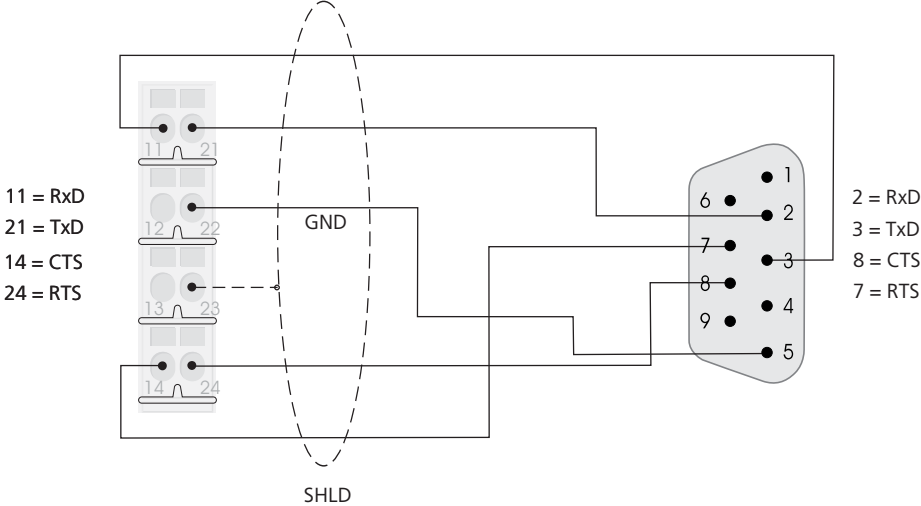
2.1	Allgemeines	2
2.1.1	Windows HyperTerminal.....	3
2.2	Einstellen der Kommunikationsparameter	4
2.2.1	Einstellen der Modulparameter in CoDeSys.....	4
2.2.2	Einstellen der Eigenschaften im HyperTerminal.....	5
2.3	CoDeSys - Aufruf des FBs und Variablendeklaration.....	6
2.4	Das Senden von Daten (Modul → HyperTerminal).....	8
2.5	Das Empfangen von Daten (HyperTerminal → Modul)	9

2.1 Allgemeines

Anhand des folgenden Beispiels wird eine RS232-Kommunikation zwischen einer BL20-Station bestehend aus einem programmierbarem Gateway und u.a. einem RS232-Modul und Windows-HyperTerminal erläutert.

Die Verbindung zwischen RS232-Modul und PC wird dabei über eine COM-Schnittstelle des PCs realisiert:

Abbildung 2-1:
Verbindung
zwischen
BL20-1RS232
und D-Sub-
Stecker



Zuordnung der Signaltypen bei einem 9-poligen Submin-D-Stecker

Tabelle 2-1:
Zuordnung der
Signaltypen RS232

Pin- Nr.	Signalbezeichnung		
1	DCD	Data Carrier Detect	Empfangssignalpegel
2	RxD	Receive Data	Empfangsdaten
3	TxD	Transmit Data	Sendedaten
4	DTR	Data Terminal Ready	Endgerät betriebsbereit
5	GND	Ground	Signalmasse
6	DSR	Data Set Ready	Betriebsbereitschaft
7	RTS	Request To Send	Sendeteil einschalten
8	CTS	Clear To Send	Sendebereitschaft
9	RI	Ring Indicator	Anruf Indikator



Hinweis

Die grau hinterlegten Tabellenreihen kennzeichnen die Signale, die auch an den Klemmen des Basismoduls verfügbar sind.

2.1.1 Windows HyperTerminal

Windows-HyperTerminal wird über „Start → (Alle) Programme → Zubehör → Kommunikation → HyperTerminal“ gestartet.

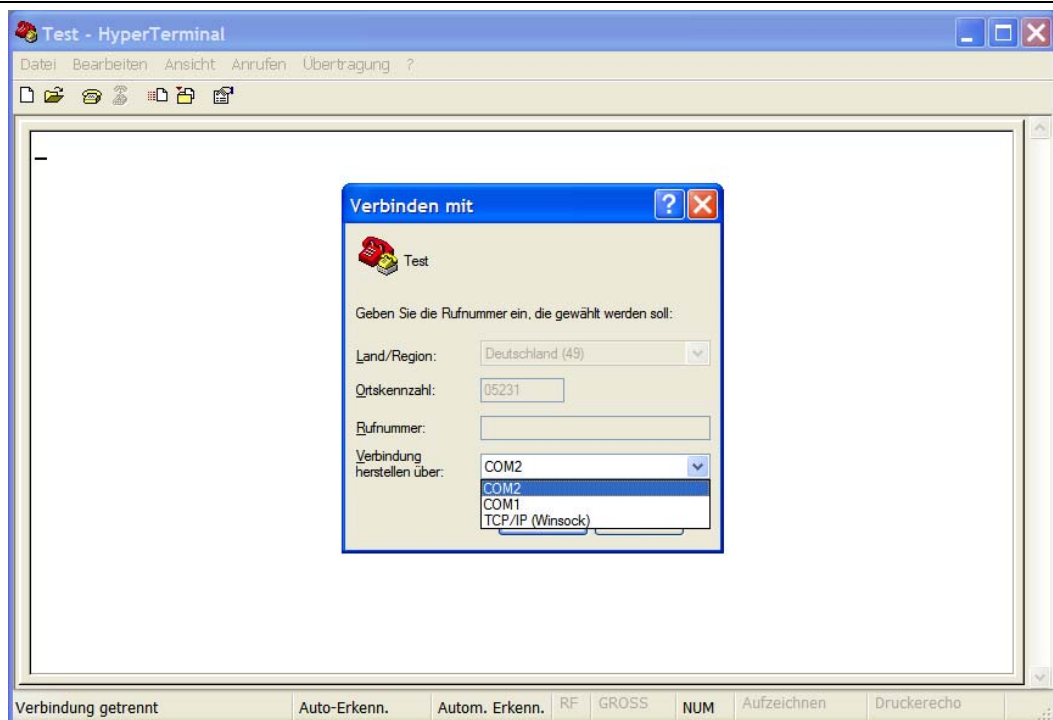


Hinweis

Geben Sie unter „Ortskennzahl“ Ihre Vorwahl an. Die Angabe der Rufnummer ist bei der Kommunikation über eine serielle Schnittstelle am PC nicht erforderlich.

Vergeben Sie im Fenster „Beschreibung der Verbindung“ einen beliebigen Verbindungsnamen und definieren Sie den COM-Port, über den die Verbindung zwischen PC und Modul hergestellt werden soll.

Abbildung 2-2:
Windows
HyperTerminal



2.2 Einstellen der Kommunikationsparameter



Hinweis

Um eine fehlerfreie RS232-Kommunikation zu gewährleisten, müssen die Kommunikationsparameter beider RS232-Teilnehmer (RS232-Modul und HyperTerminal) identisch sein.

2.2.1 Einstellen der Modulparameter in CoDeSys

Das Einstellen der Parameter des RS232-Moduls erfolgt in der Steuerungskonfiguration.

Markieren Sie dazu den Eintrag BLxx-IO [Slot] und markieren Sie dann im Register „Ein-/ Ausgänge“ unter Ausgewählte Module das Modul BL20-1RS232.

Danach öffnen Sie den Parametrier-Dialog „Moduleigenschaften“ über die Schaltfläche „Eigenschaften“.

Abbildung 2-3:
IO-Modul,
Eigenschaften

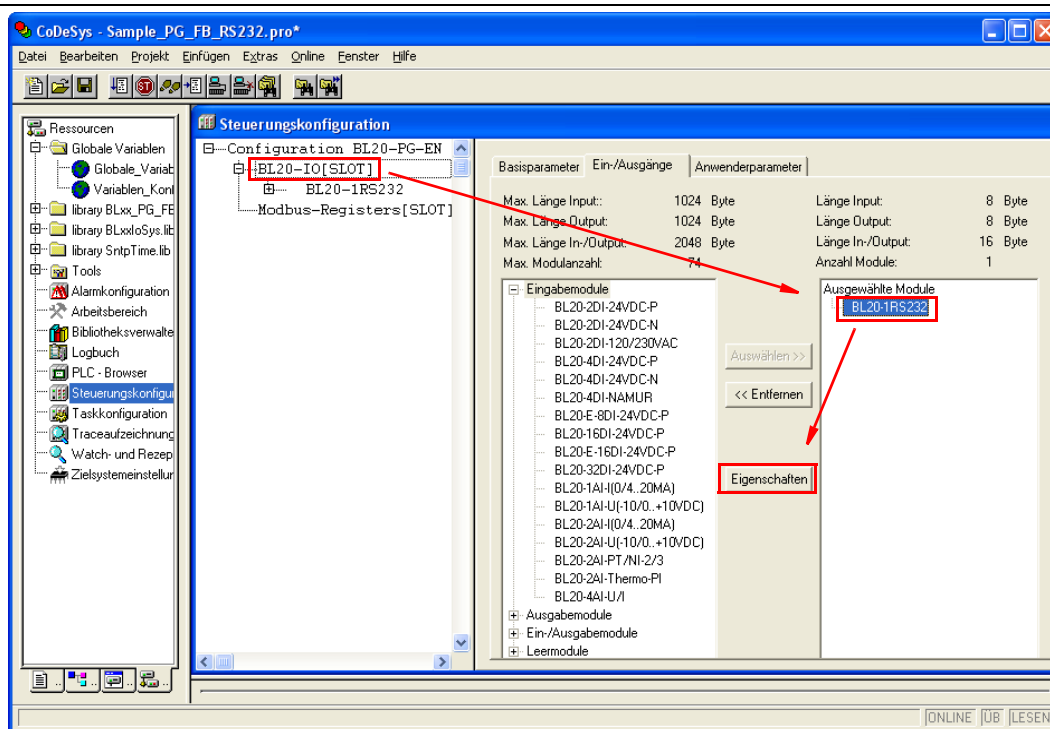
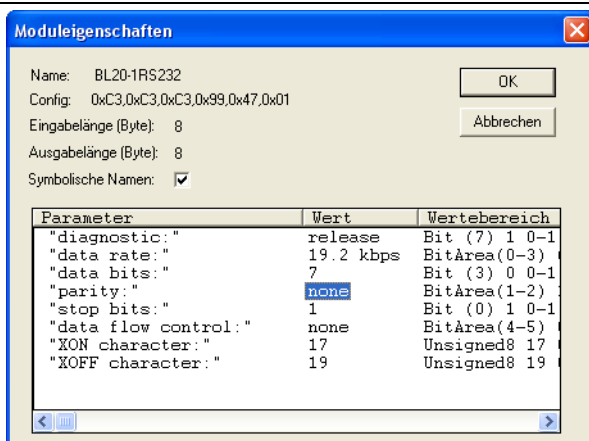


Abbildung 2-4:
Parametrierung
in „Moduleigen-
schaften“



2.2.2 Einstellen der Eigenschaften im HyperTerminal

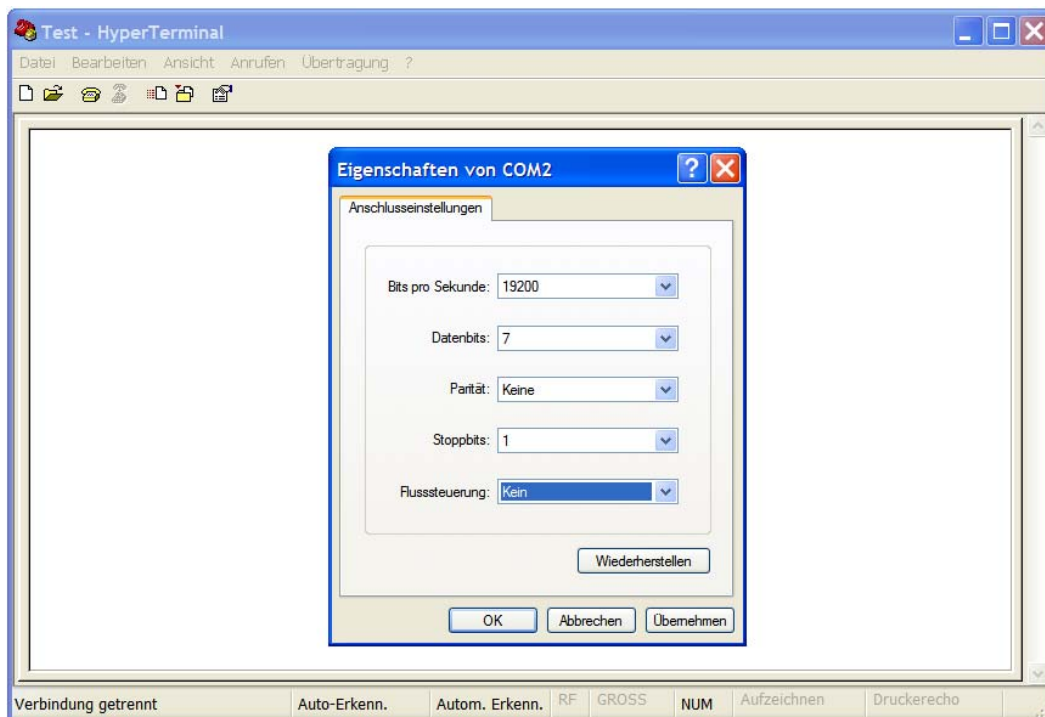
Konfigurieren Sie HyperTerminal je nach Anwendung („Datei → Eigenschaften“). Dies ist nur bei einer getrennten Verbindung möglich, ggf. muss eine bestehende Kommunikation erst über „Anrufen → Trennen“ beendet werden.



Hinweis

Bitte beachten Sie, dass die Konfigurationen des RS232-Moduls und der Verbindung im HyperTerminal identisch sind, da sonst keine fehlerfreie Kommunikation gewährleistet werden kann.

Abbildung 2-5:
Windows
HyperTerminal,
Konfiguration

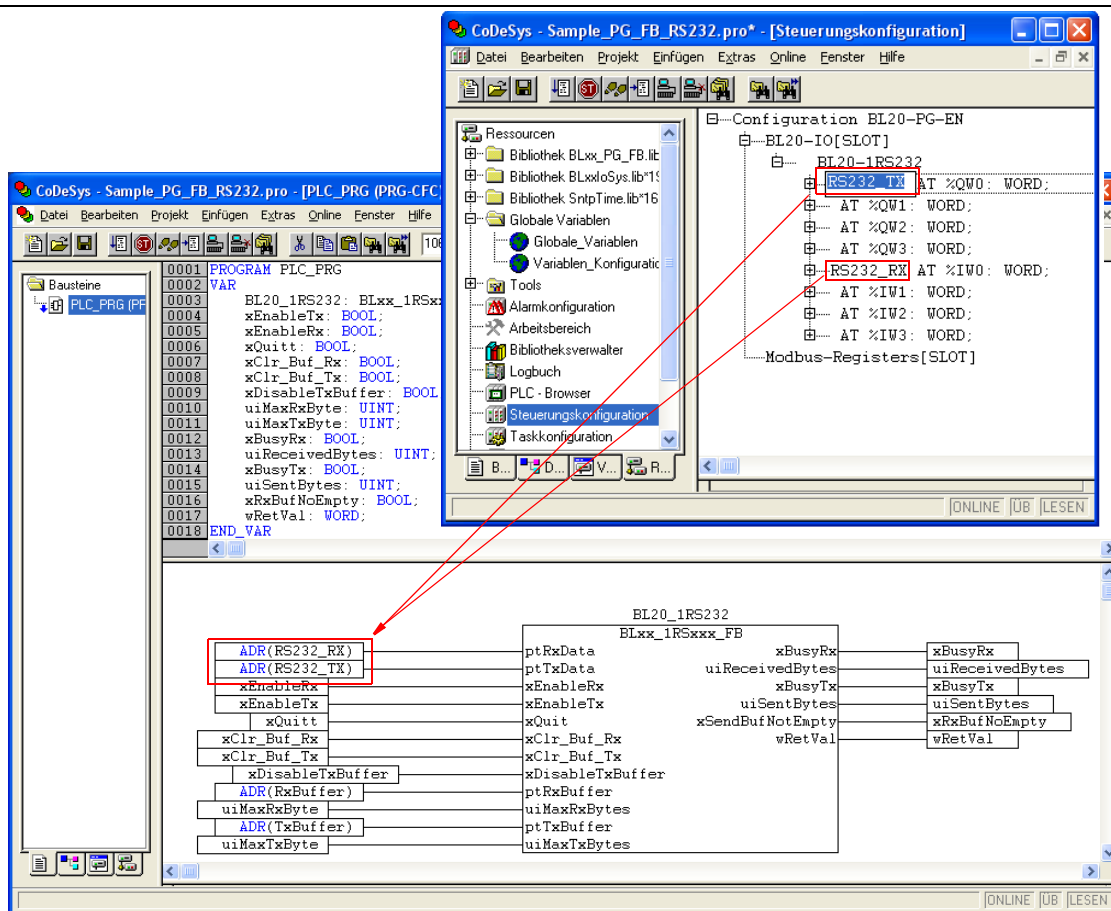


2.3 CoDeSys - Aufruf des FBs und Variablendeklaration

Rufen Sie im PLC_PRG den Funktionsbaustein BLxx_RSxxx_FB für die RSxxx-Kommunikation auf.

Sind in der Steuerungskonfiguration für das Eingangs- und das Ausgangswort des Moduls Variablen definiert worden (hier im Beispiel: „RS232_RX“ und „RS232_TX“), dann müssen diese Variablen im FB den Pointern für den Empfangs- und Sendedatenbereich („ptRxData“ und „ptTxData“, siehe auch Seite 1-8) zugeordnet werden.

Abbildung 2-6:
Variablendeklaration
und
Verwendung im
FB



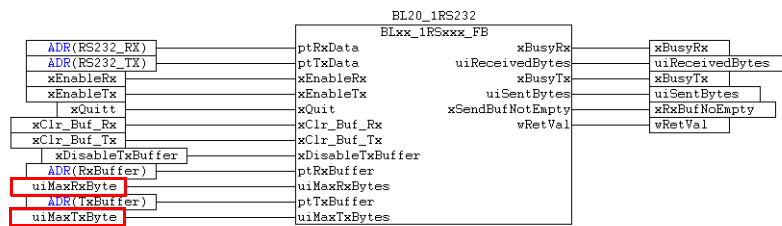
Alle übrigen Variablen sind bereits im lokalen Header (Variablendeklarationsteil) des Bausteins „PLC_PRG“ definiert.



Hinweis

Wichtig ist auch die Eingabe der maximal zu sendenden/ zu empfangenden Bytes in „uiMaxTxByte“ bzw. „uiMaxRxByte“. Ohne diese Eingabe werden keine Daten gesendet.

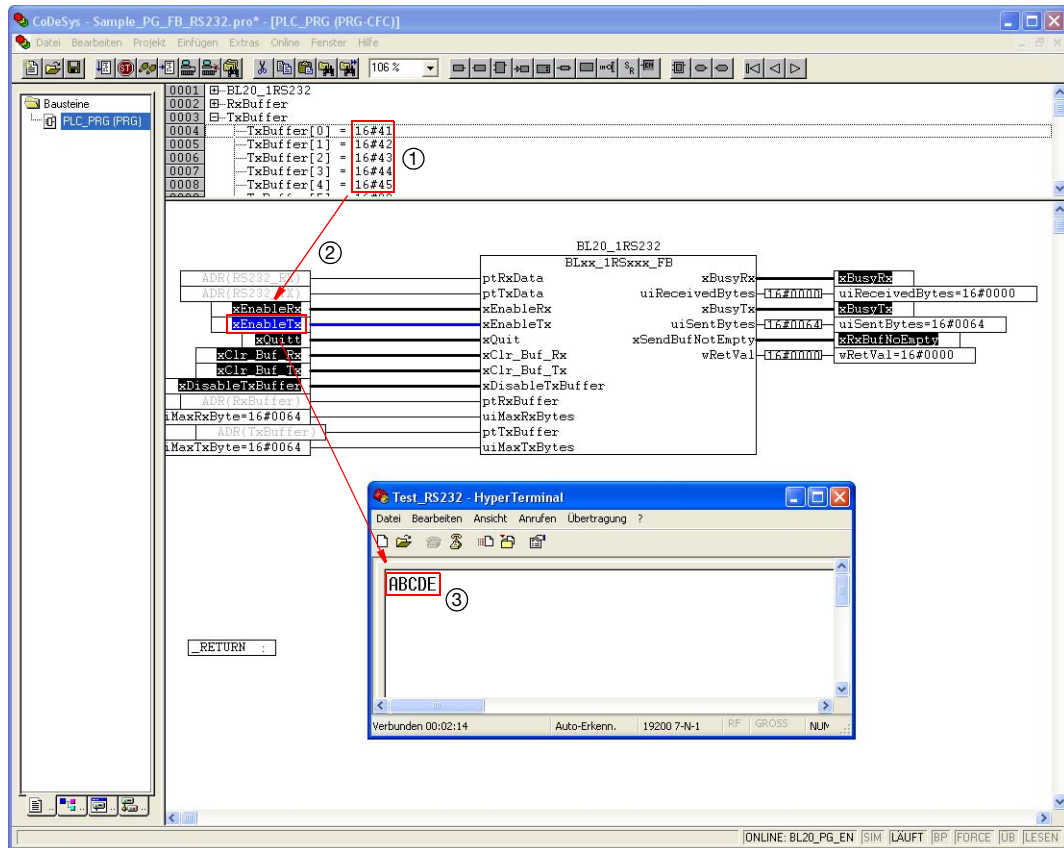
Abbildung 2-7:
Max. Anzahl der
zu sendenden
und zu empfan-
genden Daten



2.4 Das Senden von Daten (Modul → HyperTerminal)

- 1 Die zu sendenden Daten werden in den Sendepuffer „TX_Buffer“ geschrieben.
- 2 Danach muss das Senden zunächst im FB/ Modul freigeschaltet werden. Setzen Sie dazu die Variable „xEnableTx“ auf TRUE.
- 3 Im HyperTerminal werden die empfangenen Daten im ASCII-Code angezeigt.

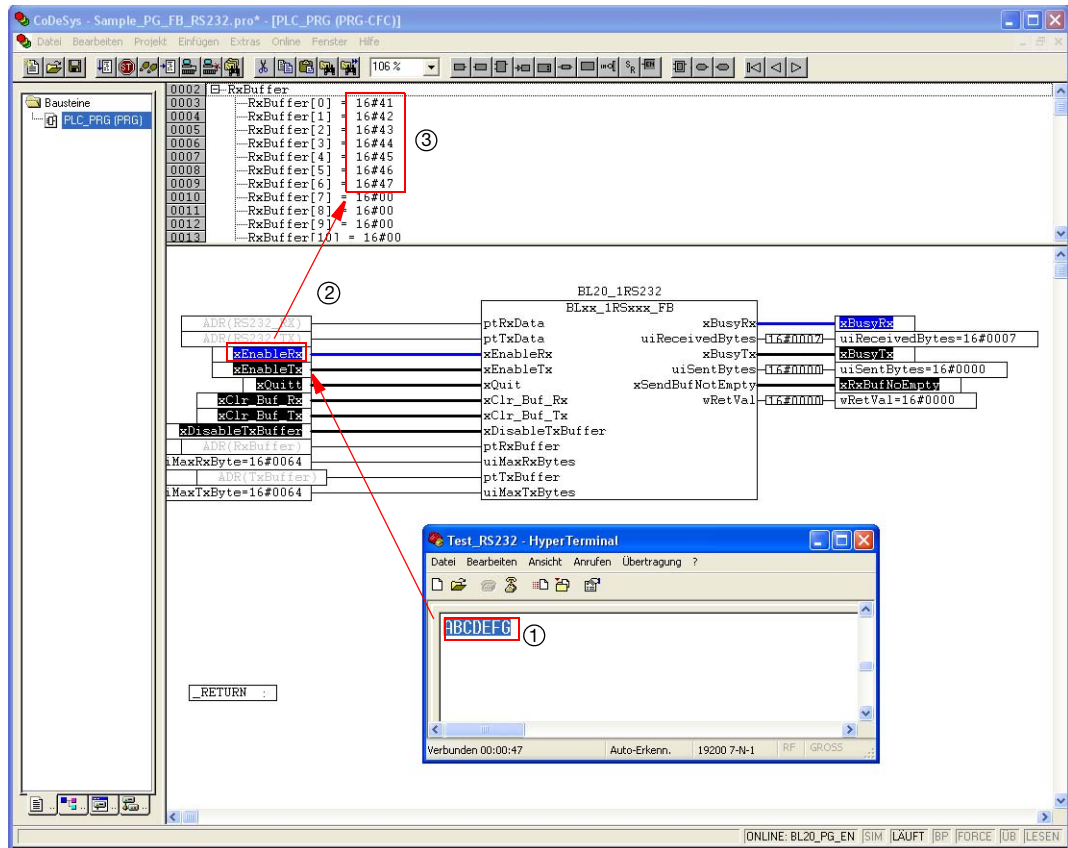
Abbildung 2-8:
Senden



2.5 Das Empfangen von Daten (HyperTerminal → Modul)

- 1 Schreiben Sie die zu sendenden Daten in den HyperTerminal.
- 2 Danach muss zunächst im FB/ Modul das Empfangen der Daten freigeschaltet werden. Setzen Sie dazu die Variable „xEnableRx“ auf TRUE.
- 3 Die empfangenen Daten werden dann im Empfangspuffer „RxBuffer“ angezeigt.

Abbildung 2-9:
Empfangen



TURCK

Industrielle
Automation



www.turck.com

Hans Turck GmbH & Co. KG
45472 Mülheim an der Ruhr
Germany
Witzlebenstraße 7
Tel. +49 (0) 208 4952-0
Fax +49 (0) 208 4952-264
E-Mail more@turck.com
Internet www.turck.com