Dokumentation
ModBus Master Connector

S. Rothenbacher GmbH

Automation Industrieelektronik GmbH

Zeppelinstraße 16

D-89160 Dornstadt, Germany

Phone +49-7348-201208

Fax +49-7348-201382

web www.Rothenbacher-GmbH.de

e-mail Info@Rothenbacher-GmbH.de

Stand 0910.2014

Inhaltsverzeichnis

1 Allgemeines

Unser Softwarepaket ModbusConneector besteht aus einer einzigen .Net Dll (Bibliothek).

Die dll ist für sämtliche Windows Plattformen entwickelt worden.Durch das Linux Mono Projekt kann die dll auch unter Linux verwendet werden.

1.1 Support

Haben Sie Fragen oder Probleme mit der Installation oder der Anwendung des Modbus Conectors, so wenden Sie sich bitte an unseren Support. Sie erreichen ihn entweder telefonisch unter (07348) 20 12 08 oder per E-Mail über support@rothenbacher-gmbh.de. Schicken Sie uns Ihre Fragen oder die Problembeschreibung mit der von ihnen verwendeten .net version und Betriebssystemversion.

1.2 Projektunterstützung

Kontaktieren Sie uns, wenn Sie einen personellen Engpass oder einfach Bedarf an kompetenter Projektunterstützung haben. Gerne realisieren wir für Sie Projekte zum Festpreis. Setzen Sie sich im Bedarfsfalle einfach mit uns in Verbindung, wir erstellen Ihnen gerne ein unverbindliches Angebot.

1.3 Lieferumfang

Der Modbus Connector wird als Zip Datei geliefert. In dieser Zip Datei sind alle Basisprogramme, Beispiele und die Demoversion der DLL enthalten.

ModBusMaster Connectro	

1.4 Einschränkung der Demo Version

In der Demoversion stehen Ihnen sämtliche Funktionen der Vollversion zur Verfügung. Es erscheint nur beim Starten ein kleiner Hinweise, dass es sich um eine Demoversion des ModbusConnectors handelt.

1.5 Lizenzvereinbarungen

EULA / Endbenutzerlizenzvereinbarung

Diese Endbenutzerlizenzvereinbarung (englisch:End User License Aggreement - im folgenden als EULA abgekürzt) enthält die Bedingungen und Konditionen bezüglich der Verwendung dieser SOFTWARE (wie unten definiert). Diese EULA enthält Beschränkungen Ihrer Rechte in Bezug auf diese SOFTWARE. Sie sollten diese EULA sorgfältig lesen und als wichtiges Merkmal dieser SOFTWARE behandeln.

1. Vereinbarung zwischen Ihnen und S.Rothenbacher GmbH

Diese EULA ist eine gesetzlich bindende Vereinbarung zwischen Ihnen und der Firma S.Rothenbacher GmbH. Sie beabsichtigen, gesetzlich an diese EULA in demselben Umfang gebunden zu werden, als ob Sie und die Firma S.Rothenbacher GmbH diese EULA physisch unterschreiben würden. Indem Sie diese SOFTWARE installieren, kopieren oder anderweitig nutzen, erklären Sie sich einverstanden, an die in dieser EULA enthaltenen Bedingungen und Konditionen gebunden zu sein. Wenn Sie nicht mit allen Bedingungen und Konditionen dieser EULA einverstanden sind, dürfen Sie die SOFTWARE nicht installieren oder benutzen.

2. Definition von "SOFTWARE"

Diese EULA regelt die Verwendung von Software-Produkten der Firma S.Rothenbacher GmbH. (beigefügt oder anderweitig verfügbar) durch Sie - einzeln und kollektiv als "SOFTWARE" bezeichnet. Der Begriff "SOFTWARE" beinhaltet, neben dem von der Firma S.Rothenbacher GmbH. gelieferten Umfang:

1) Alle Revisionen, UPDATES und/oder UPGRADES hierzu

2) Alle Daten, Bilder, ausführbare Dateien, Datenbanken, Datenbanksysteme, Computersoftware oder ähnliche Elemente, die normalerweise mit Computersoftware-Produkten verteilt oder verwendet werden

3) Alle damit in Verbindung stehenden Datenträgern, Dokumentationen (beinhaltet physische, elektronische und online verfügbare Dokumente) und gedruckte Materialien.

3. Copyright

Die SOFTWARE gehört S.Rothenbacher GmbH. und/oder ihren Lizenzgebern und wird von Copyrightgesetzen und internationalen Verträgen geschützt. Sie dürfen den Copyright-Hinweis aus keiner Kopie der SOFTWARE entfernen.

4. Einräumung einer Lizenz

Die SOFTWARE wird Ihnen nicht verkauft. Stattdessen wird die SOFTWARE auf einer nicht exklusiven Grundlage an Sie - und nur an Sie - zum Gebrauch unter den Bestimmungen dieser Vereinbarung lizenziert. S.Rothenbacher GmbH. behält alle Titel und Eigentumsrechte an der SOFTWARE sowie alle Rechte, die Ihnen nicht ausdrücklich gewährt werden. Solange Sie keine Lizenz für die SOFTWARE erhalten und installiert haben, läuft die SOFTWARE im Demomodus und ist in ihrer Funktionalität eingeschränkt oder funktioniert eines Tages nach der Installation nicht mehr. Für Details zum Demomodus lesen Sie bitte die der SOFTWARE beigefügten Dokumentation.

5. Verwendung nur an einem Einzelarbeitsplatz

Die SOFTWARE darf nur an einem einzigen Arbeitsplatz genutzt werden. Ein Arbeitsplatz ist definiert durch die Kombination eines physischen oder virtuellen Computers (oder einer Session auf einem Terminal-Server) und einer Person. Sie dürfen die SOFTWARE auf jedem Computer eines Arbeitsplatzes installieren (z.B. Arbeitsstation und Notebook), wenn sichergestellt ist, dass die SOFTWARE zu keiner Zeit von mehr als einer Person verwendet wird.

6. Eine Archivierungskopie

Sie dürfen nur eine einzige Sicherungskopie anfertigen, die ausschließlich zu Archivierungszwecken genutzt werden darf. Diese darf nicht an Dritte weiter gegeben werden.

7. Dekompilierung, Entassemblierung oder Zurückentwicklung

Sie erkennen an, dass die SOFTWARE Betriebsgeheimnisse und andere Eigentumsinformationen der Firma S.Rothenbacher GmbH. und/oder ihren Lizenzgebern enthält. Sie dürfen die SOFTWARE nicht dekompilieren, entassemblieren oder auf irgendeine Art und Weise zurückentwickeln (Reverse Engineering) - noch sich mit anderen Aktivitäten beschäftigen, um die zugrunde liegenden Informationen zu erhalten, die für den Benutzer während der normalen Verwendung der SOFTWARE nicht sichtbar sind.

8. Aktualisierungen (Updates und Upgrades)

Diese Lizenz räumt Ihnen kein Recht auf irgendwelche Erweiterungen, Fehlerbeseitigungen, Programmkorrekturen oder Aktualisierungen der SOFTWARE ein - noch irgendwelche Support-Dienstleistungen. UPDATES (Aktualisierungen) sind definiert als neue Versionen der SOFTWARE, in denen sich die Hauptversionsnummer nicht geändert hat (die Hauptversionsnummer ist die erste Zahl der Versionskennzeichnung des Produktes. Beispiel: Die Hauptversionsnummer von "1.2.3.4" ist 1). Wird die Hauptversionsnummer geändert, dann wird diese neue Version der SOFTWARE als UPGRADE definiert. Verfügbare UPDATES werden kostenlos von der Firma S.Rothenbacher GmbH. über die jeweilige Produkt-Website oder auf der Hauptwebsite (www.rothenbacher-gmbh.de.de) zum Download zur Verfügung gestellt (Kosten für Ihre Internet-Verbindung und/oder Kosten des Transfers selber sowie jegliche Kosten, die in Verbindung mit dem Erhalt des UPDATES stehen, müssen von Ihnen übernommen werden).

9. Beendigung

Die Ihnen erteilte Lizenz ist bis zur Beendigung gültig. Die Beendigung kann zu jeder Zeit durch die Rückgabe der SOFTWARE (inkl. aller Kopien davon) an die Firma S.Rothenbacher GmbH. stattfinden. Außerdem wird die Gültigkeit Ihrer Lizenz automatisch beendet (auch ohne einen Hinweis von der Firma S.Rothenbacher GmbH.), wenn Sie einer Bedingung oder Kondition dieser Vereinbarung zuwiderhandeln. Sie stimmen bei einer solchen Beendigung zu, sämtliche Bestandteile der SOFTWARE (inkl. Kopien davon) an die Firma S.Rothenbacher GmbH. zurückzugeben. Bei Beendigung kann die Firma S.Rothenbacher GmbH. die ihr per Gesetz zustehenden Rechte durchsetzen. Die Bedingungen und Konditionen dieser Vereinbarung, die die Eigentumsrechte der Firma S.Rothenbacher GmbH. schützen, bleiben auch nach Beendigung in Kraft.

10. Haftungsausschluss

Die Firma S.Rothenbacher GmbH. garantiert nicht, dass die in der SOFTWARE enthaltenen Funktionen Ihre Anforderungen erfüllt und/oder dass der Betrieb der SOFTWARE ununterbrochen, fehlerfrei oder frei von arglistigem Code ist ("arglistiger Code" bezeichnet jeglichen Programm-Code, der entwickelt wurde, um andere Computer-Programme und/oder Computer-Daten zu kontaminieren, Computerbetriebsmittel zu verbrauchen, Daten zu ändern / zu löschen / aufzuzeichnen oder zu übermitteln - oder in irgendeiner anderen Art den Normalbetrieb von Computern, Rechnersystemen oder Computernetzwerken zu stören, einschließlich Viren, trojanischen Pferden, Droppern, Würmern, Logik-Bomben und Ähnliches).

Diese SOFTWARE wird WIE SIE IST geliefert – ohne irgendwelche Garantien. S.Rothenbacher GmbH ist nicht verpflichtet, UPDATES, UPGRADES oder technischen Support für diese SOFTWARE bereitzustellen. Die Firma S.Rothenbacher GmbH übernimmt außerdem keine Haftung für die Genauigkeit sämtlicher von S.Rothenbacher GmbH oder von Dritten zur Verfügung gestellten Informationen oder für irgendwelche Schäden, die direkt oder indirekt durch Aktionen oder Unterlassungen aufgrund dieser Informationen entstehen.

Sie übernehmen die volle Verantwortung für die Auswahl der SOFTWARE, um Ihre zukünftigen Ergebnisse zu erreichen, sowie für die Installation, Benutzung und die Ergebnisse, die Sie von der SOFTWARE erhalten. Weiterhin übernehmen Sie das volle Risiko in Bezug auf Qualität und Leistung der SOFTWARE.

Sollte sich die SOFTWARE als fehlerhaft erweisen, übernehmen Sie (und nicht die Firma S.Rothenbacher GmbH oder ihre Distributoren oder Händler) die gesamten Kosten für alle notwendigen Service-, Reparatur- und/oder Korrektur-Leistungen.

In keinem Fall haftet die Firma S.Rothenbacher GmbH oder ihre Lizenzgeber für direkte, indirekte, beiläufige oder besondere Schäden noch für Folgeschäden oder irgendwelche Verluste, entgangene Gewinne, entgangene Einnahmen, entgangene Einsparungen oder für entstandene Datenverluste, die durch oder in Verbindung mit dieser SOFTWARE oder dieser Vereinbarung entstehen, selbst wenn die Firma S.Rothenbacher GmbH oder ihre Lizenzgeber über die Möglichkeit solcher Schäden unterrichtet wurde. In jedem Fall ist die Haftung auf den für die SOFTWARE gezahlten Betrag beschränkt, unabhängig von der Art des Schadenfalls.

11. Schlussklausel

Diese Vereinbarung bindet Sie wie auch Ihre Angestellten, Arbeitgeber, Auftragnehmer und Agenten sowie alle Nachfolger und Bevollmächtigten. Diese Vereinbarung ist die gesamte Vereinbarung zwischen uns und hat Vorrang vor allen anderen Absprachen und Vereinbarungen. Falls eine Bedingung oder Kondition dieser Vereinbarung ungültig ist oder wird, bleiben alle anderen Bedingungen und Konditionen dieser Vereinbarung davon unberührt und behalten ihre Gültigkeit. In einem solchen Fall verpflichten sich beide Parteien, die ungültig Bedingung und/oder Kondition durch eine gültige Bedingung und/oder Kondition zu ersetzten, die in ihrer rechtlichen, wirtschaftlichen und technischen Bedeutung möglichst gleichkommt. Gerichtsstand ist Deutschland.

1.6 Kompatibilität

1.6.1 Modbus TCP/IP / RTU RS232/R485

Bit access	Physical Discret Input	Read Discrete Inputs	Function Code:02
	Internal Bits or Physical Bits	Read Coils	Function Code:01
		Write Singel Coil	Function Code:05
		Write Multiple Coil	Function Code:15
16 Bits Access	Physical Input Register	Read Input Register	Function Code:04
	Internel Register	Read Holding Register	Function Code:03
		Write Singel Register	Function Code:06
		Write Multipel Register	Function Code:16

1.7 Kompatibilität Betriebssystem

1.7.1 Windows X86 und X64 /ARM / MIPS

		32 Bit	64 Bit
Windows 95	.Net Framwork / Mono Framwork	-	-
Windows Me	.Net Framwork / Mono Framwork	-	-
Windows 98	.Net Framwork / Mono Framwork	X	-
Windows NT Workstation 4.0 (alle Versionen - Service Pack 6a erforderlich)	.Net Framwork / Mono Framwork	X	-
Windows 2000 Professional	.Net Framwork / Mono Framwork	X	-
Windows XP Home Edition	.Net Framwork / Mono Framwork	X	X
Windows XP Professional	.Net Framwork / Mono Framwork	X	X
Windows Vista	.Net Framwork / Mono Framwork	X	X
Windows 7	.Net Framwork / Mono Framwork	X	X
Windows 8	.Net Framwork / Mono Framwork	X	X
Windows 2003 Server	.Net Framwork / Mono Framwork	X	X
Windows 2008 Server	.Net Framwork / Mono Framwork	X	X
Windows CE .NET 4.0 – 4.2	.NET Compact Framework	X	-
Windows CE 5.0	.NET Compact Framework	X	-
Windows Embedded CE 6.0	.NET Compact Framework	X	-

1.7.2 Linux / MAC / ARM

		32 Bit	64 Bit
Ubuntu	Mono Framwork	X	X
Red Hat	Mono Framwork	X	X
OpenSuse	Mono Framwork	X	X
Debian	Mono Framwork	X	X
LinuxMint	Mono Framwork	X	X
Knoppix	Mono Framwork	X	X
Raspberry PI - Raspbian	Mono Framwork	X	-
CubieBoard 2	Mono Framwork	X	-
Apple OSX	Mono Framwork	X	X

Und sicher viele mehr ….

2 Grundlegendes zur Anwendung

2.1 Der Namespace

Alle im folgenden verwendete Funktionen sowie Variablen des Modbus-Connectors befinden sich im Namespace: „Modbus”.

Die Dokumentation bezieht sich darauf das der Namespace dem Projekt bekannt gegeben wurde.
Beispiel für C#

```
using Modbus;
```

2.2 Die Basisklasse

Die Basisklasse „S7.Base“ stellt den Ausgangspunkt der Projektierung dar.
Sie beinhaltet alle Funktionsaufrufe um z.B. Daten aus einem DB zu lesen bzw. zu schreiben.
Hierfür wird eine Instanz der Basisklasse benötigt.
Beispiel für C#

```
private Modbus.Master MBmaster;
```

Der Zugriff auf die Funktionen erfolgt nun über diese Instanz:

```
MBmaster;
```

2.3 Die Überladung der Basisklasse

Diese Basisklasse wird durch den Verbindungsaufbau über eine der abgeleiteten Klassen „überschrieben“ und nutzt dadurch die physikalischen und logischen Verbindungsparameter, die bei der Verbindung angegeben werden.

Mögliche Verbindungsklassen zur Überladung sind:

- `MBmaster = new Master(TB_IP_Address_TCP.Text, 502);`
- `MBmaster = new Master(CB_Port_RTU.Text, Convert.ToInt32(TB_Baud_RTU.Text), Convert.ToInt32(TB_Databits_RTU.Text), RTU_Selected_Parity, RTU_Selected_Stopbits,false);`

Eine genaue Anleitung zu den einzelnen Verbindungsarten finden sie unter „3 Verbindungsaufbau“

3 Verbindungsaufbau

Bevor Sie auf Daten über Modbus zugreifen können muss die Kommunikation initialisiert werden.Stellen Sie die gewünschte Kommunikation über den Aufruf der Verbindungsaufbau- Master() ein. Die Fehlerbehandlung der einzelne Funktionen können Sie über die try{ }Code... catch (Exception E) abfangen. Eine Liste der möglichen Fehler und ihrer Ursachen entnehmen Sie dem Kapitel (5).

3.1 Modbus TCP-IP

	Beschreibung
Position	`public Master(string ip, ushort port)`

Codeauszug C#	`Try`
	`{`
	`// Create new modbus master`
	`Master.Licence = "Demo";`
	`MBmaster = new Master(TB_IP_Address_TCP.Text, 502);`
	`}`
	
	`//Troubleshooting, Fehlerbehandlung`
	`catch (SystemException error)`
	`{`
	`MessageBox.Show(error.Message);`
	`}`

Beschreibung	`Create master instance with parameters for Modbus TCP`
Parameter	`"ip">IP adress of modbus slave.` `"port">Port number of modbus slave. Usually port 502 is used.`

3.2 Modbus RTU

	Beschreibung
Position	`public Master(string portName, int baudRate, int databits, Parity parity, StopBits stopBits,bool Rs485_echo)`

Codeauszug C#	`try`
	`{`
	`// Create new RTU modbus master`
	`Master.Licence = "Demo";`
	`MBmaster = new Master(CB_Port_RTU.Text, Convert.ToInt32(TB_Baud_RTU.Text), Convert.ToInt32(TB_Databits_RTU.Text), RTU_Selected_Parity, RTU_Selected_Stopbits,false);`
	
	`}`
	`catch (SystemException error)`
	`{`
	`MessageBox.Show(error.Message);`
	`}`

Beschreibung	`Create master instance with parameters for Modbus RTU`

Parameter	`"portName">Gets the port for communications from or sets`
	`"baudRate">1200 - 115200`
	`"databits">8,7`
	`"parity">N,E,O`
	`"stopBits">1,2`
	`"Rs485_echo">true False`

4 Modbus Funktionen

4.1 Read Coils Function Code:01

This function code is used to read contiguous status of coils in a remote
device. The Request of the function specifies the starting address, i.e. the address of the first coil. specified, the number of coils.and the Coils are addressed starting. Therefore coils numbered 1-16 are addressed as 0-15.

Diese Funktion wird benötigt, um mehrere zusammenhängende Zustände von Ausgängen eines Gerätes zu lesen. Die Rückgabe der Funktion ist ein Abbild der Ausgänge ab der Start-Adresse.
Übergeben wird Start und die Anzahl der zu lesenden Ausgänge. Die Ausgänge 1-16,werden als 0-15 angesprochen.

	Beschreibung
Position	`public void ReadCoils(ushort Slave, ushort startAddress, ushort numcoils ,ref byte[] values)`

Codeauszug C#	`try`
	`{`
	`ushort Slave = Convert.ToUInt16(TB_Address_RTU.Text); //important for rtu`
	`ushort Length = Convert.ToUInt16(TB_Count_Read.Text); //number of Bits`
	
	`byte[] byteresult = new byte[1]; // Start of Bits`
	`ushort StartAddress = Convert.ToUInt16(TB_StartAddress_Read.Text);`
	
	`//Modbus Read`
	`MBmaster.ReadCoils(Slave, StartAddress, Length,ref byteresult);`
	
	`//conversion, Wandlung`
	`object Result = MBmaster.bytetodata(byteresult,Master.DataTypes.BOOL);`
	
	`LB_Result.Items.Clear(); //Result, Ergebnis`
	`for (int i = 0; i < (Result as Array).Length; i++)`
	`LB_Result.Items.Add("VAL" + i.ToString("0000") + " = " +`
	`(Result as Array).GetValue(i).ToString());`
	`}`
	
	`//Troubleshooting, Fehlerbehandlung`
	`catch (Exception E)`
	`{`
	`LB_Errors.Items.Add(E.ToString());`
	`}`

Beschreibung	`Read coils from slave synchronous.`

Parameter	`"Slave">ID Modbus Slave`
	`"startAddress">Address from where the data read begins in Bits.`
	`"numcoils">Length of data. in Bits`
	`"values">Contains the result of function.`

4.2 Read Discrete Inputs Function Code:02

This function code is used to read contiguous status of inputs in a remote device. The Request of the function specifies the starting address, i.e. the address of the first input. specified, the number of inputs.and the inputs are addressed starting. Therefore inputs numbered 1-16 are addressed as 0-15.

Diese Funktion wird benötigt, um mehrere zusammenhängende Zustände von Eingängen eines Gerätes zu lesen. Die Rückgabe der Funktion ist ein Abbild der Eingänge ab der Start-Adresse.
Übergeben wird Start und die Anzahl der zu lesenden Eingänge. Die Eingange 1-16,werden als 0-15 angesprochen.

	Beschreibung
Position	`public void ReadDiscreteInputs(ushort Slave, ushort startAddress, ushort numInputs, ref byte[] values)`

Codeauszug C#	`try`
	`{`
	`ushort Slave = Convert.ToUInt16(TB_Address_RTU.Text);`
	`ushort Length = Convert.ToUInt16(TB_Count_Read.Text);`
	`byte[] byteresult = new byte[1];`
	`ushort StartAddress = Convert.ToUInt16(TB_StartAddress_Read.Text);`
	
	`MBmaster.ReadHoldingRegister(Slave, StartAddress, Length, ref byteresult);`
	
	`//conversion, Wandlung`
	`object Result = Mbmaster.bytetodata(byteresult,Master.DataTypes.BOOL);`
	
	`//Result, Ergebnis`
	`LB_Result.Items.Clear();`
	`for (int i = 0; i < (Result as Array).Length; i++)`
	`LB_Result.Items.Add("VAL" + i.ToString("0000") + " = " + (Result as Array).GetValue(i).ToString());`
	
	`}`
	`//Troubleshooting, Fehlerbehandlung`
	`catch (Exception E)`
	`{`
	`LB_Errors.Items.Add(E.ToString());`
	`}`

Beschreibung	`Read discrete inputs from slave synchronous.`

Parameter	`"Slave">ID Modbus Slave`
	`"startAddress">Address from where the data read begins.`
	`"numInputs">Length of data.`
	`"values">Contains the result of function.`

4.3 Read Holding Register Function Code:03

This function code is used to read the contents of a contiguous block of holding registers in a remote device. The Request specifies the starting register address and the number of registers.

Diese Funktion wird benötigt, um das Block Register von einem Gerät zu lesen. Die Rückgabe der Funktion ist ein Abbild der Register ab der Start-Adresse.
Übergeben wird Start und die Anzahl der zu lesenden Register.

	Beschreibung
Position	`public void ReadHoldingRegister(ushort Slave, ushort startAddress, ushort numInputs, ref byte[] values)`

Codeauszug C#	`try`
	`{`
	`ushort Slave = Convert.ToUInt16(TB_Address_RTU.Text); //important for rtu`
	`ushort Length = Convert.ToUInt16(TB_Count_Read.Text); //number word`
	`byte[] byteresult = new byte[1];`
	`ushort StartAddress = Convert.ToUInt16(TB_StartAddress_Read.Text); //Start`
	
	`//Modbus Read`
	`MBmaster.ReadHoldingRegister(Slave, StartAddress, Length, ref byteresult);`
	
	`//conversion, Wandlung`
	`object Result = MBmaster.bytetodata(byteresult,Read_Selected_DataType);`
	
	`//Result, Ergebnis`
	`LB_Result.Items.Clear();`
	`for (int i = 0; i < (Result as Array).Length; i++)`
	`LB_Result.Items.Add("VAL" + i.ToString("0000") + " = " + (Result as Array).GetValue(i).ToString());`
	`}`
	`//Troubleshooting, Fehlerbehandlung`
	`catch (Exception E)`
	`{`
	`LB_Errors.Items.Add(E.ToString());`
	`}`

Beschreibung	`Read holding registers from slave synchronous`

Parameter	`"Slave">ID Modbus Slave`
	`"startAddress">Address from where the data read begins.`
	`"numInputs">Length of data.`
	`"values">Contains the result of function.`

4.4 Read Input Register Function Code:04

This function code is used to read contiguous input registers in a remote device.
The Request specifies the starting register address and the number of input registers.

Diese Funktion wird benötigt, um das Input Register von einem Gerät zu lesen.
Die Rückgabe der Funktion ist ein Abbild der Input Register ab der Start-Adresse.
Übergeben wird Start und die Anzahl der zu lesenden Input Register.

	Beschreibung
Position	`public void ReadInputRegister(ushort Slave, ushort startAddress, ushort numInputs, ref byte[] values)`

Codeauszug C#	`Try`
	`{`
	`ushort Slave = Convert.ToUInt16(TB_Address_RTU.Text);//important for rtu`
	`ushort Length = Convert.ToUInt16(TB_Count_Read.Text);//number word`
	`byte[] byteresult = new byte[1];`
	`ushort StartAddress = Convert.ToUInt16(TB_StartAddress_Read.Text);//Start`
	
	`// Modbus Read`
	`MBmaster.ReadInputRegister(Slave, StartAddress, Length, ref byteresult);`
	
	`//conversion, Wandlung`
	`object Result = MBmaster.bytetodata(byteresult, Read_Selected_DataType);`
	
	`//Result`
	`LB_Result.Items.Clear();`
	`for (int i = 0; i < (Result as Array).Length; i++)`
	`LB_Result.Items.Add("VAL" + i.ToString("0000") + " = " + (Result as Array).GetValue(i).ToString());`
	`}`
	`//Troubleshooting, Fehlerbehandlung`
	`catch (Exception E)`
	`{`
	`LB_Errors.Items.Add(E.ToString());`
	`}`

Beschreibung	`Read input registers from slave synchronous.`

Parameter	`"Slave">ID Modbus Slave`
	`"startAddress">Address from where the data read begins.`
	`"numInputs">Length of data.`
	`"values">Contains the result of function.>`

4.5 Write Singel Coil Function Code:05

This function code is used to write a single output to either ON or OFF in a remote device.The requested ON/OFF state is specified by a constant in the request data field. A value of true requests the output to be ON. A value of false requests it to be OFF. All other values are illegal and will not affect the output.The specifi es the address of the coil to be forced. Coils are addressed starting at zero. Therefore coil numbered 1 is addressed as 0. The requested ON/OFF state is specified by a constant in the Coil Value field.

Diese Funktion wird benötigt, um einen einzelnen Ausgang in einem Gerät zu beschreiben. Dabei wird in dem Partnergeräte ein Ausgang je nach Parameter auf true oder false geschaltet. Alle anderen Parameter als true, bzw. false werden zurückgewiesen.
Übergeben wird Start und der zu schreibende Wert des Ausgangs. Die Ausgänge 1-16,werden als 0-15 angesprochen.

	Beschreibung
Position	`public void WriteSingleCoils(ushort Slave, ushort startAddress, bool OnOff, ref byte[] result)`

Codeauszug C#	`try`
	`{`
	`ushort Slave = Convert.ToUInt16(TB_Address_RTU.Text);//important for rtu`
	`byte[] byteresult = new byte[1];`
	`ushort StartAddress = Convert.ToUInt16(TB_StartAddress_Read.Text);//number of Bits`
	
	`//Modbus Write Coil ON`
	`MBmaster.WriteSingleCoils(Slave, StartAddress, true, ref byteresult);`
	
	`//Troubleshooting, Fehlerbehandlung`
	`}`
	`catch (Exception E)`
	`{`
	`LB_Errors.Items.Add(E.ToString());`
	`}`

Beschreibung	`Write single coil in slave synchronous.`

Parameter	`"Slave">ID Modbus Slave`
	`"startAddress">Address from where the data read begins.`
	`"OnOff">Specifys if the coil should be switched on or off.`
	`"result">Contains the result of the synchronous write.`

4.6 Write Singel Register Function Code:06

This function code is used to write a single holding register in a remote device.
The Request PDU specifies the address of the register to be written. Registers are addressed starting at zero. Therefore register numbered 1 is addressed as 0.
The normal response is an echo of the request, returned after the register contents have been written.

Diese Funktion wird benötigt, um ein einzelnes Block Register zu beschreiben.
Übergeben wird Start und der zu schreibende Wert des Block Register.Das Register 1 wird als 0 angesprochen.

	Beschreibung
Position	`public void WriteSingleRegister(ushort Slave, ushort startAddress, byte[] values, ref byte[] result)`

Codeauszug C#	`try`
	`{`
	`ushort Slave = Convert.ToUInt16(TB_Address_RTU.Text);//important for rtu`
	`byte[] byteresult = new byte[4];`
	`ushort StartAddress = Convert.ToUInt16(TB_StartAddress_Read.Text);//Start`
	
	`//conversion, Wandlung`
	`byteresult = MBmaster.ValueToBytes(Convert.ToInt16(TB_Value.Text), 2);`
	
	`//Modbus Write`
	`MBmaster.WriteSingleRegister(Slave, StartAddress, byteresult, ref byteresult);`
	`}`
	`//Troubleshooting, Fehlerbehandlung`
	`catch (Exception E)`
	`{`
	`LB_Errors.Items.Add(E.ToString());`
	`}`

Beschreibung	`Write single register in slave synchronous.`

Parameter	`"Slave">ID Modbus Slave`
	`"startAddress">Address to where the data is written.`
	`"values">Contains the register information.`
	`"result">Contains the result of the synchronous write.`

4.7 Write Multiple Coil Function Code:15

This function code is used to force each coil in a sequence of coils
to either ON or OFF in a remote device. The Request specifies the coil references to be forced. Coils are addressed starting at zero. Therefore coil numbered 1 is addressed as 0.The requested ON/OFF states are specified by contents of the request data field. A logical '1'
in a bit position of the field requests the corresponding output to be ON. A logical '0' requests
it to be OFF.The normal response returns the function code, starting address, and quantity of coils forced

Diese Funktion wird benötigt, um mehrere Ausgang in einem Gerät zu beschreiben. Dabei wird in dem Partnergeräte die Ausgang je nach Parameter auf true oder false geschaltet. Die Ausgänge 1-16,werden als 0-15 angesprochen.Übergeben wird Start und die zu schreibende Wert der Ausgangs.

	Beschreibung
Position	`public void WriteMultipleCoils(ushort Slave, ushort startAddress, ushort numBits, byte[] values, ref byte[] result)`

Codeauszug C#	`Try`
	`{`
	`ushort Slave = Convert.ToUInt16(TB_Address_RTU.Text);//important for rtu`
	`byte[] byteresult = new byte[2];`
	`ushort StartAddress = Convert.ToUInt16(TB_StartAddress_Read.Text); //Start`
	
	`// all coil on`
	`byteresult[0] = 255;`
	`byteresult[1] = 255;`
	
	`//Modbus Write`
	`MBmaster.WriteMultipleCoils(Slave,`
	`StartAddress,Convert.ToByte(byteresult.Length * 8), byteresult, ref byteresult);`
	
	`}`
	`//Troubleshooting, Fehlerbehandlung`
	`catch (Exception E)`
	`{`
	`LB_Errors.Items.Add(E.ToString());`
	`}`

Beschreibung	`Write multiple coils in slave synchronous.`

Parameter	`"Slave">ID Modbus Slave`
	`"startAddress">Address from where the data read begins.`
	`"numBits">Specifys number of bits.`
	`"values">Contains the bit information in byte format.`
	`"result">Contains the result of the synchronous write.`

4.8 Write Multiple Register Function Code:16

This function code is used to write a block of contiguous registers in a remote device. The requested written values are specified in the request data field. Data is packed as two bytes per register.The normal response returns the function code, starting address, and quantity of registers written.

Diese Funktion wird benötigt, um mehrere Block Register zu beschreiben.
Übergeben wird Start und die zu schreibende Wert des Block Register.Das Register 1 wird als 0 angesprochen.Das zu schreibende Register wird als zwei bytes an die methode übergeben.mehrere Register werden als byte arry an die methode übergenben.

	Beschreibung
Position	`public void WriteMultipleRegister(ushort Slave, ushort startAddress, byte[] values, ref byte[] result)`

Codeauszug C#	`try`
	`{`
	`ushort Slave = Convert.ToUInt16(TB_Address_RTU.Text);//important for rtu`
	`byte[] byteresult;`
	`ushort StartAddress = Convert.ToUInt16(TB_StartAddress_Read.Text);//Start`
	
	`//conversion, Wandlung`
	`byteresult = MBmaster.ValueToBytes(Convert.ToInt16(TB_Value.Text), 2);`
	
	`//Modbus Write`
	`MBmaster.WriteMultipleRegister(Slave, StartAddress, byteresult, ref byteresult);`
	`}`
	
	`//Troubleshooting, Fehlerbehandlung`
	`catch (Exception E)`
	`{`
	`LB_Errors.Items.Add(E.ToString());`
	`}`

Beschreibung	`Write multiple registers in slave synchronous.`

Parameter	`"Slave">ID Modbus Slave`
	`"startAddress">Address to where the data is written.`
	`"values">Contains the register information.`
	`"result">Contains the result of the synchronous write.`

4.9 Datentypen

An die Schreibe oder Lese Methoden werden (bis auf die Methoden die einzelne bits beschreiben) immer Bytes übergeben beziehungsweise zurückgegeben.
Für die Wandlung der Empfangene und zusendender daten sthen zwei Methoden zur Verfügung:
bytetodata und ***ValuetoByte***

```
//conversion, Wandlung
object Result = MBmaster.bytetodata(byteresult,Read_Selected_DataType);

//conversion, Wandlung
byteresult = MBmaster.ValueToBytes(Convert.ToInt16(TB_Value.Text), 2);
```

Jeder elementare Datentyp verfügt über einen zugeordneten Speicherplatz mit fester Länge. Der Datentyp BOOL zum Beispiel hat nur ein Bit, ein Byte (BYTE) besteht aus 8 Bits, ein Wort (WORD) sind 2 Bytes (bzw. 16 Bits), ein Doppelwort (DWORD) hat 4 Bytes (bzw. 32 Bits). Die folgende Tabelle zeigt alle vorhandenen elementaren Datentypen:

S7 Datentype	C# Datentype	Beschreibung	Bits	Wertebereichs
BOOL	bool	Einzelnes Bit	1	TRUE FALSE
BYTE	byte	Festpunktzahl	8	0 bis 255
WORD	uint16	Festpunktzahl	16	0 bis 65535
DWORD	uint32	Festpunktzahl	32	
INT	int16	Festpunktzahl	16	-32768 bis 32767
DINT	int32	Festpunktzahl	32	-2147483648 bis 2147483647
REAL	float	Gleitpunktzahl	32	1.0 1.238 1.2E-2 -5E12 100.0 -1000.0 -123E-18
S5TIME	S5Time	Zeitwert Simatic	16	S5T#0ms bis S5TIME#2h46m30s
TIME	TimeIEC32	Zeitwert IEC 32	32	-24d20h31m23s647ms bis 24d20h31m23s647ms
CHAR	char	ASCII-Zeichen	8	
DATE			16	
TIME_OF_DAY			32	
COUNTER				

5 Fehlerbehandlung

5.1 Exception Handling

Kommt es währen der Kommunikation oder durch die Kommunikation zu einem Programm Fehler so können diese mittels try{ }Code... catch (Exception E) abgefangen werden.

Die Kommunikation wird über CRC (Cyclical Redundancy Check) kontrolliert (nur bei RTU Verbindung). Bei Auftreten eines Kommunikationsfehlers antwortet der angesprochene Slave mit einem Fehler Protokoll, oder ModBusConnector meldet Timeout.
Wird eine Anfrage nicht ausgeführt, so antwortet der ModBusConnector mit einer AUSNAHME-ANTWORT, wie in der folgenden Tabelle aufgeführt.

5.2 Error List

ILLEGAL FUNCTION	The function code received in the query is not an allowable action for the server. This may be because the function code is only applicable to newer devices, and was not implemented in the unit selected. It could also indicate that the server is in the wrong state to process a request of this type, for example because it is unconfigured and is being asked to return register values.	
ILLEGAL DATA ADDRESS	The data address received in the query is not an allowable address for the server. More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, the PDU addresses the first register as 0, and the last one as 99. If a request is submitted with a starting register address of 96 and a quantity of registers of 4, then this request will successfully operate (address-wise at least) on registers 96, 97, 98, 99. If a request is submitted with a starting register address of 96 and a quantity of registers of 5, then this request will fail with Exception Code 0x02 “Illegal Data Address” since it attempts to operate on registers 96, 97, 98, 99 and 100, and there is no register with address 100.	
ILLEGAL DATA VALUE	A value contained in the query data field is not an allowable value for server. This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does NOT mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the MODBUS protocol is unaware of the significance of any particular value of any particular register.	
SERVER DEVICE FAILURE	An unrecoverable error occurred while the server was attempting to perform the requested action.	

ACKNOWLEDGE	Specialized use in conjunction with programming commands. The server has accepted the request and is processing it, but a long duration of time will be required to do so. This response is returned to prevent a timeout error from occurring in the client. The client can next issue a Poll Program Complete message to determine if processing is completed.	
SERVER DEVICE BUSY	Specialized use in conjunction with programming commands. The server is engaged in processing a long–duration program command. The client should retransmit the message later when the server is free.	
MEMORY PARITY ERROR	Specialized use in conjunction with function codes 20 and 21 and reference type 6, to indicate that the extended file area failed to pass a consistency check. The server attempted to read record file, but detected a parity error in the memory. The client can retry the request, but service may be required	
GATEWAY PATH UNAVAILABLE	Specialized use in conjunction with gateways, indicates that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. Usually means that the gateway is misconfigured or overloaded.	
GATEWAY TARGET DEVICE FAILED TO RESPOND	Specialized use in conjunction with gateways, indicates that no response was obtained from the target device. Usually means that the device is not present on the network.	
TIMEOUT		
CRC CHECKSUM ERROR		
TCP CONNECTION LOST		
TCP RESPONSE TIMEOUT		
TCP WRONG OFFSET		
TCP SEND FAILT		