Alpenstraße 34
87700 Memmingen

Modbus-Master-Treiber

1. Einleitung

MODBUS ist ein offenes serielles Kommunikationsprotokoll, das auf einer Master/Slave Architektur basiert. Dabei greift der MODBUS-Master (Touch Panel PC) auf die fest definierten Speicherbereiche eines MODBUS-Slaven (z.B. Mess- und Regelsysteme, IO-Systeme) zu. Die tatsächliche Aufteilung welche Daten in welchem Bereich stehen, ist nicht vorgeschrieben, dies wird gerätespezifisch definiert. Die konkrete Definition für ein Gerät ist das MODBUS-Profil des Gerätes.
Es bestehen die Möglichkeiten über die serielle Schnittstelle (RS232/RS485) als auch über Ethernet die Verbindung zu mehreren Slaven aufzubauen. Die Kommunikation geht dabei immer vom Master (Client) aus. Slaven (Server) kommunizieren niemals untereinander und beginnen auch keine Verbindung mit dem Master.

Diese Anleitung dient ausschließlich den notwendigen Einstellungen zur fehlerfreien Kommunikation und kann keine allgemeine Einführung in MODBUS geben. Hierzu wird auf die genaue Beschreibung des Protokolls im Internet unter der Adresse http://www.modbus.org verwiesen.

1.1 Genereller Aufbau

Der Modbus hat folgende grundlegende Struktur:

Connection ➔ Slave ➔ Register ➔ Variable.

Dies bedeutet, dass zuerst eine Connection erstellt wird, an welcher ein oder mehrere Slave(s) hängen. Diese bekommen dann je nach Profil ein oder mehrere Register, wobei jedes definierte Register einen Speicherbereich festgelegt bekommt, der während eines Zyklus bearbeitet wird. Anschließend wird einem Register noch eine oder mehrere Variable(n) zugewiesen, die dann einen Bereich im Speicher des Registers belegen.

Alpenstraße 34
87700 Memmingen

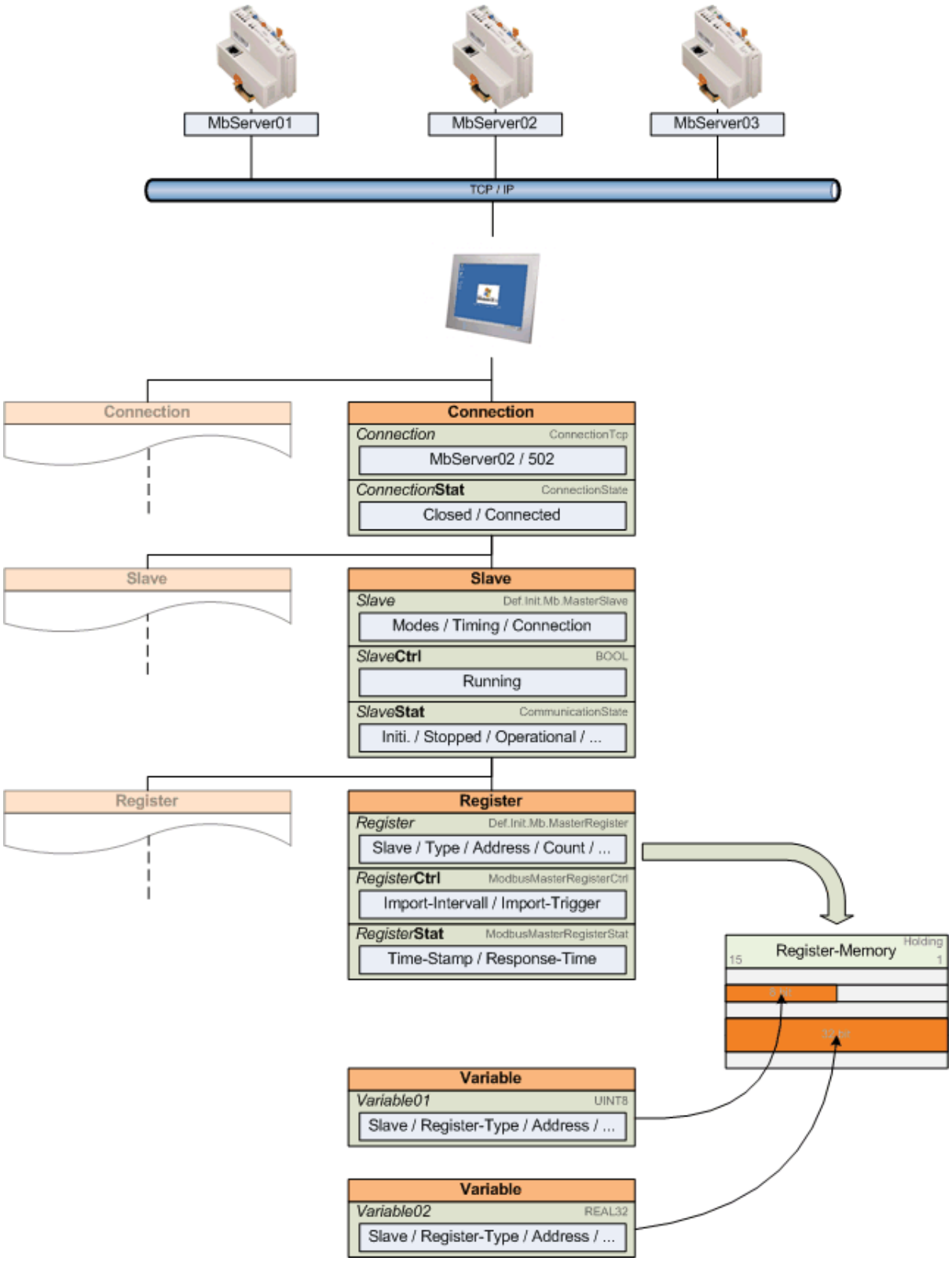


Abbildung 1: Struktureller Aufbau der MODBUS-Konfiguration.

Die nachfolgende Tabelle beschreibe die verwendeten Elemente aus Abbildung 1:

Variablen-Name	Variablen-Type	Beschreibung
Connection	ConnectionTcp	Gibt die IP-Adresse und den Port des Slaves am Netzwerk an.
*Connection***Stat**	ConnectionState	Zeigt den Status dieser Connection an (closed/connected).
Slave	DefaultInitializationModbusMasterSlave	Einstellung zu dem Slave (Mode/ Zeitliches Verhalten/ verwendete Connection/…)
*Slave***Ctrl**	BOOL	Schaltet die Kommunikation zu diesem Slave ein/aus (true/false).
*Slave***Stat**	CommunicationState	Zeigt den Status der Communication an (Initialization/Stopped/Operational/..)
Register	DefaultInitializationModbusMasterRegister	Die Register-Einstellung besteht u.a. aus Type und Größe des dargestellten Speicherblockes, sowie das Update-Intervall.
*Register***Ctrl**	ModbusMasterRegisterCtrl	Hier lässt sich das Import-Intervall für das Register dynamisch zur Laufzeit einstellen (0=kein Import). Außerdem ist ein triggern eines einmaligen Imports möglich.
*Register***Stat**	ModbusMasterRegisterStat	Der Zeitpunkt des letzten Imports und die dazu benötigte Zeit werden hier angezeigt.

Hinweise:

- Zur Verknüpfung der Initialisierungs-Variable mit den Control- und Status-Variablen bedient sich der Treiber des Namens und des Variablen-Typs. D.h. der Name von Control- und Status-Variable beginnt mit der gleichen Bezeichnung und wird lediglich um „**Stat**“ oder „**Ctrl**“ ergänzt.
- Die Werte der **Ctrl**-Variablen werden zur Laufzeit direkt beachtet, die der **Stat-**Variablen werden vom Treiber zur Laufzeit aktualisiert.
- Die Werte aus den Initialisierungs-Variablen werden nur während der Initialisierung des Treibers verwendet.
- Der vorhandene Speicherbereich in einem Slaven kann je nach Wichtigkeit durch mehrere definierte Register mit unterschiedlichen Update-Intervallen abgedeckt werden.
- Die Zuordnung welche Variable in welches Register gelegt wird erfolgt über die *Slave-Address*. Ebenso wird das Register über die *Slave-Address* dem Slaven zugeordnet.

Alpenstraße 34
87700 Memmingen

2 Visbee

2.1 Erstellen des Treibers

Nach einem Doppelklick auf *Driver* im *Solution Explorer* öffnet sich der *DriverBrowser*. Über das Kontextmenü oder im Reiter *DriverBrowser* ist via *Add* ein neuer Treiber anzulegen.

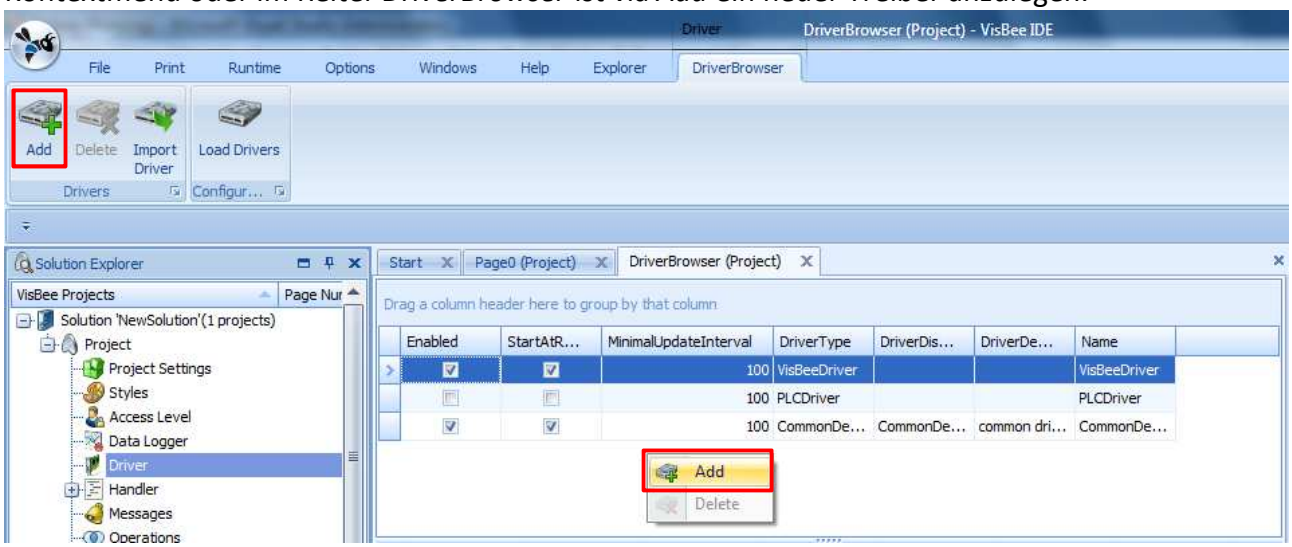


Abbildung 2: Modbus-Treiber – Einen neuen Treiber erstellen

Anschließend ist *DriverType ModbusMaster* auszuwählen. Auch sollte der Treiber mit einem sinnvollen Namen versehen werden.

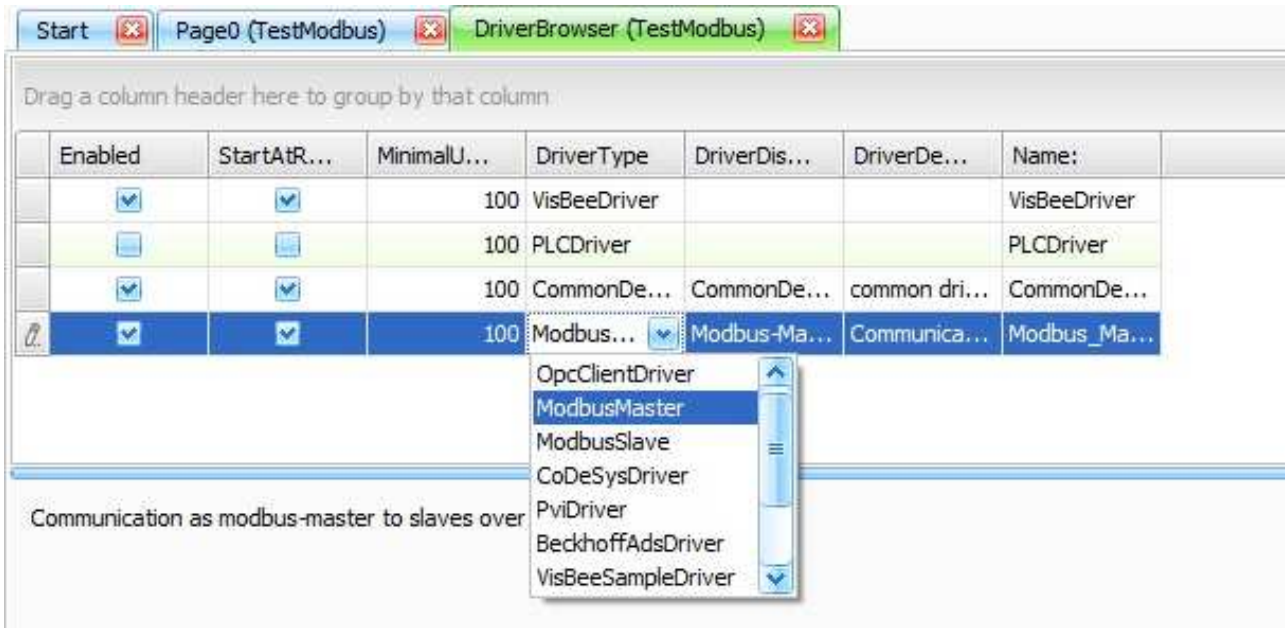


Abbildung 3: Modbus-Treiber – Treiber als ModbusMaster kennzeichnen

2.2 Konfiguration

Der ModbusMaster muss für einen Verbindungsaufbau noch konfiguriert werden. Dies kann man entweder über das Kontextmenü des Variablenfensters mit *Configure Driver* oder über das Ribbon-Symbol im Variablenfenster machen.

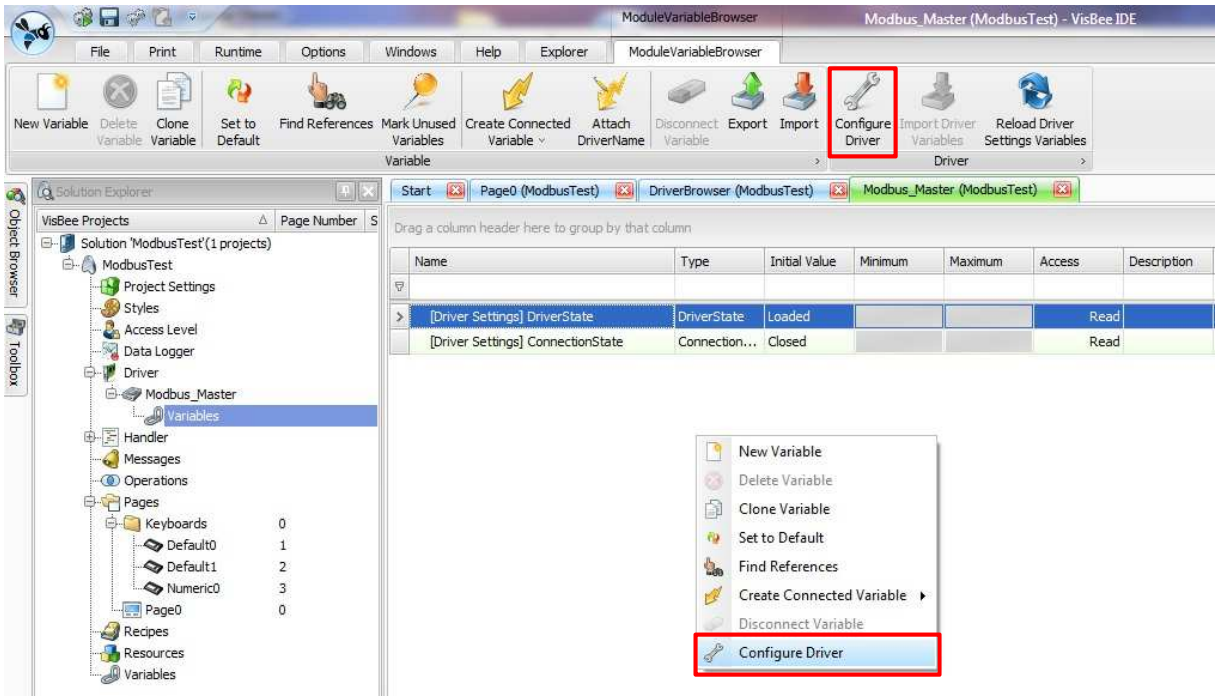


Abbildung 4: Modbus-Treiber – Konfigurationsdialog auswählen

Jetzt öffnet sich der Konfigurationsdialog, wo die unter Punkt 1.1 erklärte Struktur des Modbus aufgebaut werden muss. Hier fügen Sie also zuerst eine TCP- oder COM-Verbindung hinzu, dann einen oder mehrere Slave(s) und anschließend ein oder mehrere Register.

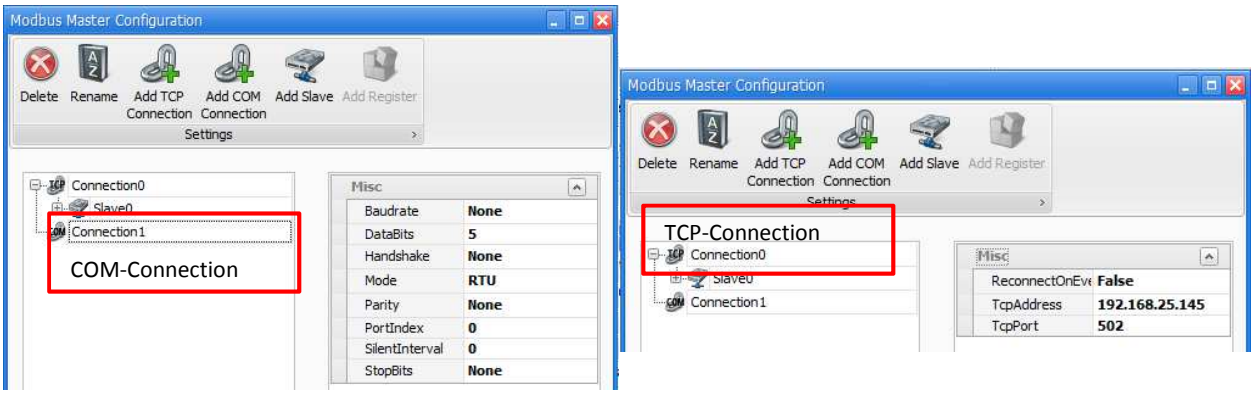


Abbildung 5: Modbus-Treiber - Konfigurationsdialoge der verschiedenen Connections

Abbildung 6: Modbus-Treiber - Konfigurationsdialog des Slave

Abbildung 7: Modbus-Treiber - Konfigurationsdialog eines Registers

Die Parameter der hinzugefügten Objekte können natürlich angepasst werden (Abbildung 5, Abbildung 6, Abbildung 7). Folgende Objekt-Parameter kann man in dem Konfigurationsdialog des ModbusMaster einstellen:

Objekt-Name	Beschreibung
Connection (TCP)	Gibt die IP-Adresse und den Port des Slaves am Netzwerk an
Connection (COM)	Gibt die Baudrate, die DataBits, den Handshake, den Mode, etc. des Slaves am Netzwerk an
Slave	Einstellung zu dem Slave (Mode / Timeout Ms / Connection Name / …)
Register	Die Register-Einstellung besteht u. a. aus Type und Größe des dargestellten Speicherblocks

Alle Variablen (auch die Control- und Status-Variablen) werden separat erstellt. Diese Settings sind dann mit OK zu bestätigen.

Bei den Registern unterscheidet man zwischen folgenden Typen:

Name	Size	Direction	Function-Code		
			Read multiple	Write single	Write multiple *)
Coils	Bit	Slave <-> Master	FC1	FC5	FC15
Discrete	Bit	Slave -> Master	FC2	---	---
Holding	Word	Slave <-> Master	FC3	FC6	FC16
Input	Word	Slave -> Master	FC4	---	---

*) Je nach Auswahl in der Initialisierung des Registers.

2.3 Variablen hinzufügen

Nachdem man eine Connection mit Slaves und Registern erstellt hat, muss man noch eine oder mehrere Variable/n anlegen, welche dann in das entsprechende Register ihren Wert speichern. Variablen legt man über das Kontextmenü oder die Ribbon-Schaltfläche *New Variable* im *ModuleVariableBrowser* an. Dieser sollte man noch einen eindeutigen Namen geben.

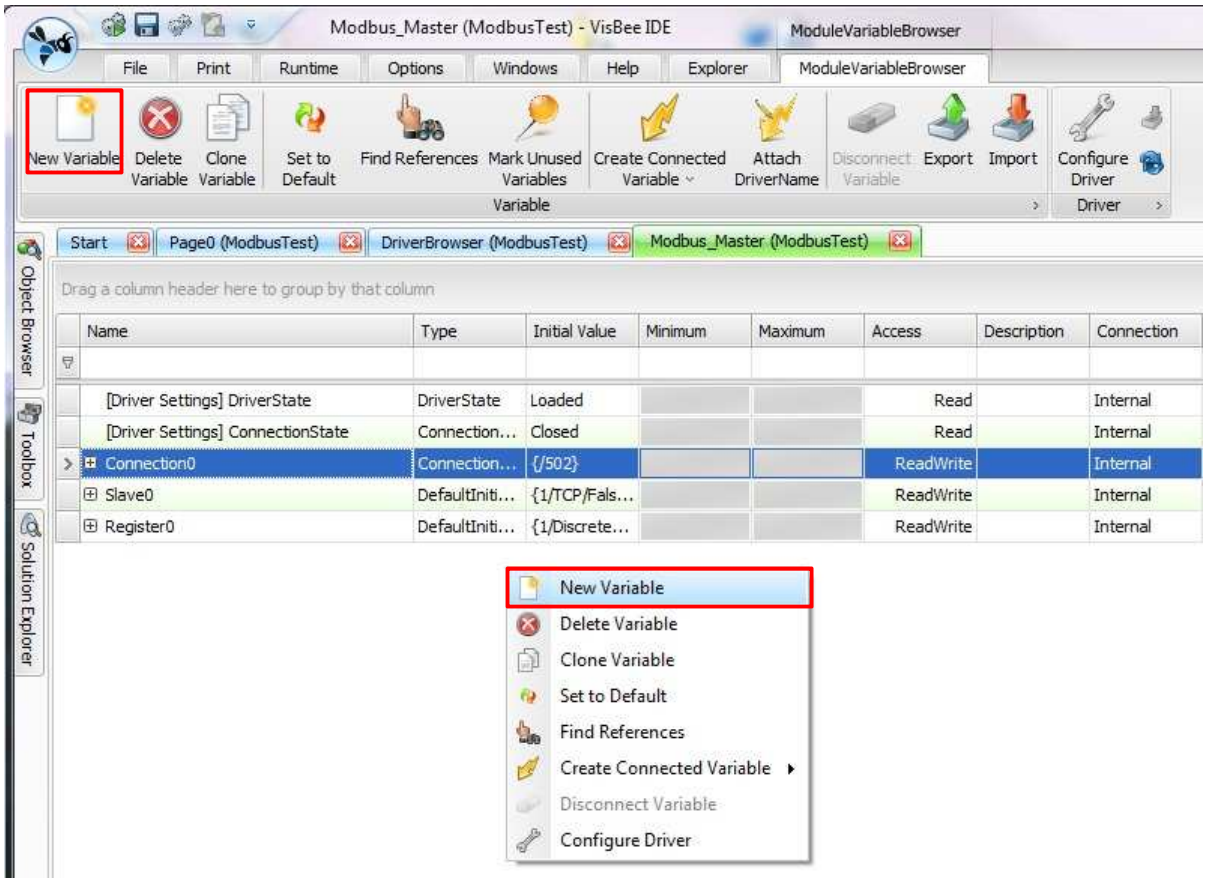


Abbildung 8: Modbus-Treiber – Anlegen einer Variablen

Alpenstraße 34
87700 Memmingen

Bei der Variablen ist darauf zu achten, das die *Slave Address* auf den entsprechenden Slave gesetzt ist, das der *Register Type* mit dem gewünschten erstellen Register übereinstimmt und das eine entsprechende *Register Address* gesetzt ist. Diese ist die Start-Adresse ist, ab der die Variable in den Registerspeicher kopiert wird. Größere Variablen belegen automatisch die nachfolgenden Register. Die *Bit Address* wird nur bei den Register Types „Holding“ und „Input“ verwendet, wenn die Variable kleiner als 16 Bit ist.

Abbildung 9: Modbus-Treiber – Parameter-Einstellungen der Variablen

2.4 Variable verknüpfen

Nach dem anlegen und erstellen der ModbusMaster-Variablen, gilt es jetzt noch diese als Visbee-Variable zu erstellen. Dies funktioniert über die Ribbon-Schaltfläche *Create Connected Variable* oder über das Kontextmenü.

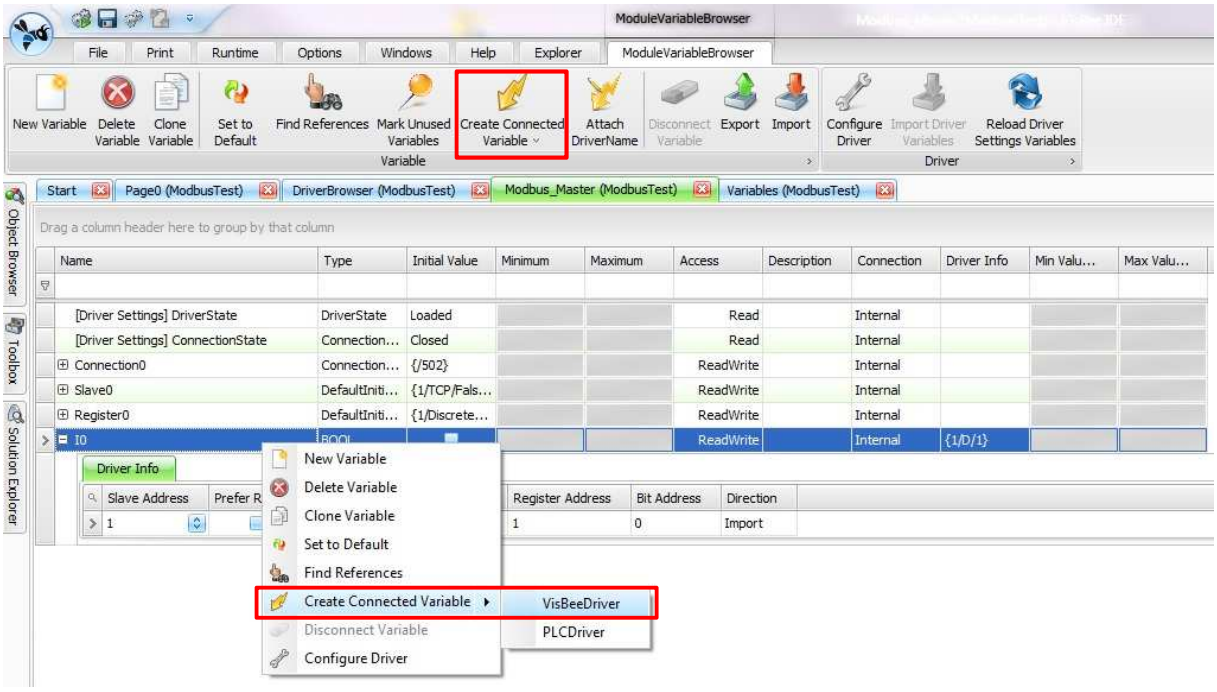


Abbildung 10: Modbus-Treiber – Create Connected Variable

Jetzt taucht die erstellte I0-Variable nach einem Doppelklick auf Variables im Solution Explorer auf. Diese sollte man jetzt noch umbenennen, sodass sofort ersichtlich wird, dass es eine Driver-Variable ist (also hier Name = Modbus_Master_I0).

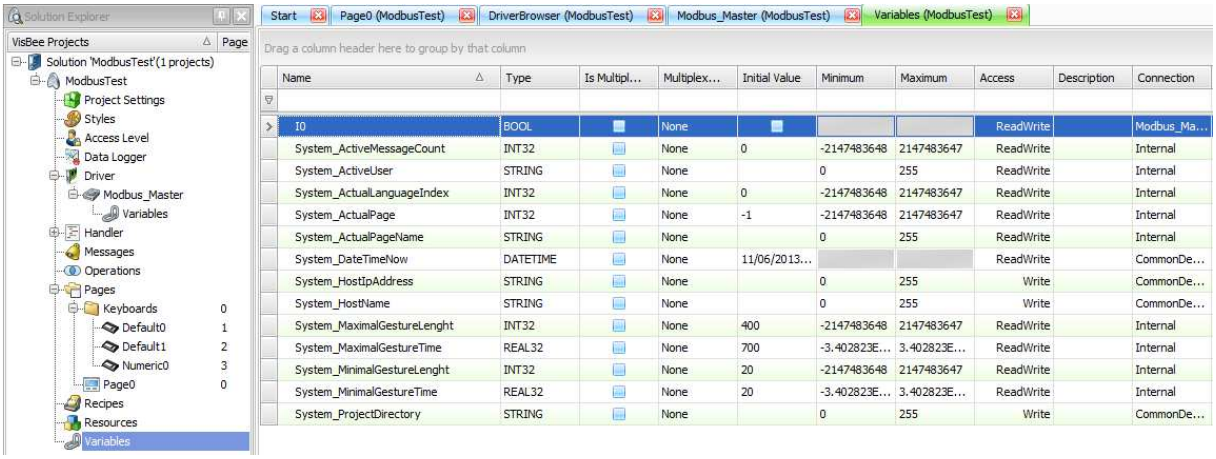
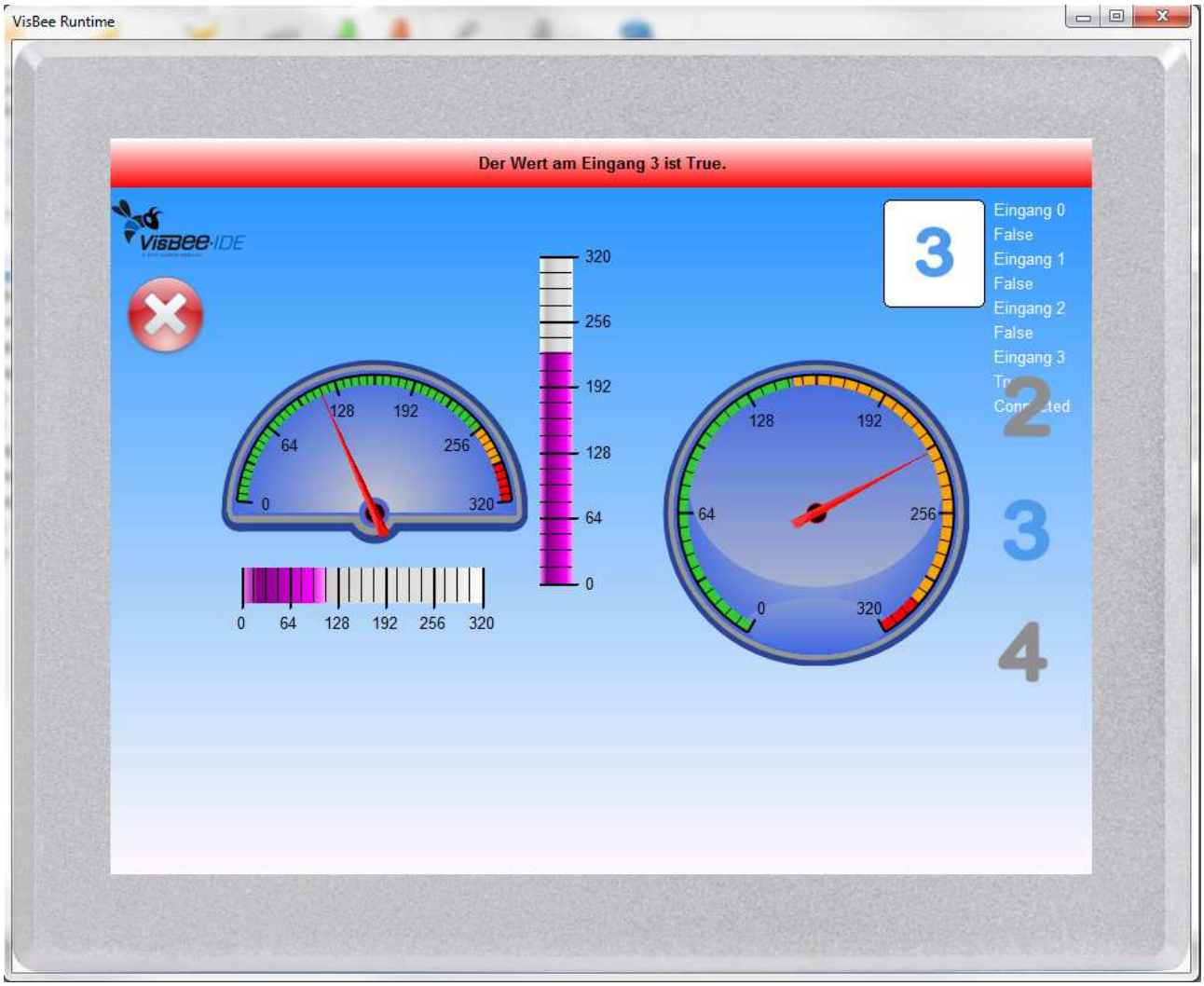


Abbildung 11: Modbus-Treiber – Connected Variable

Alpenstraße 34
87700 Memmingen

3 Beispiel-Solution

Die Visbee-Beispiel-Solution stellt eine TCP-Verbindung mit einer WAGO-Klemme her, wobei das Gerät mit der Applikation und die Klemme selbst im gleichen Netzwerk sind. An der Klemme sind Schalter und Regler angeschlossen, welche eine Zustandsänderung (z. B. Drücken eines Schalters) direkt an die Visbee-Applikation schickt.



Hier können Sie die Demo-Solution für den Modbus-Treiber herunterladen:
http://www.visbee.de/elements/resources/File/Demo-Projects/ModBusDemo.zip