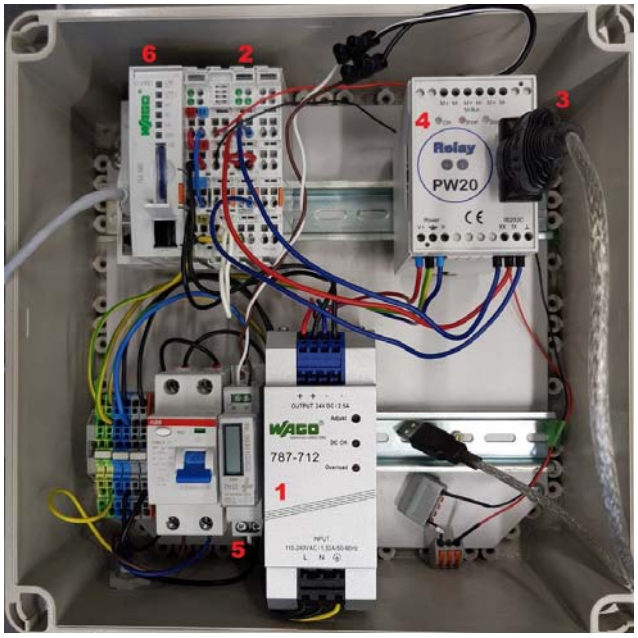
Hochschule **Rosenheim**
University of Applied Sciences

# WAGO mit M-Bus

# (mit CoDeSys V2.x)




## Tutorial für das Praktikum „Gebäudeautomation“

Version 02, 19. Juni 2018

Hochschule Rosenheim · Hochschulstrasse 1 · 83024 Rosenheim
*www.fh-rosenhheim.de* · *michael.kroedel@fh-rosenheim.de*

# Tutorial: WAGO und M-Bus

## Inhaltsverzeichnis



**Sonderbemerkung**

Dieses Tutorial wurde an der Hochschule Rosenheim im Rahmen von Projektarbeiten unter Leitung von Herrn Prof. Dr. Michael Krödel erstellt.

Mitgewirkt haben:

- Simon Hartmann, Deni Hemen (Ersterstellung 2014)
- Stephan Bröcker, Michael Höß (Überarbeitung 2015)
- Simon Gürzing, Albert Krutzenbichler, Andreas Molling, Lukas Stecher (Überarbeitung 2017)

# 1 Aufgabenstellung

Im Folgenden wird beschrieben, wie man mit Hilfe des WAGO-Controllers einen M-Bus-Zähler auslesen kann. Für das Beispiel wird ein Elektrozähler ausgelesen (andere Zählerarten können analog dazu ausgelesen werden), die empfangenen Werte werden in verschiedenen Variablen gespeichert und können bei Bedarf weiterverarbeitet werden.
Für das Praktikum wird der zweite WAGO-Controller benutzt, der im grauen Schaltkasten verbaut ist. Dieser Aufbau hat bereits einen MBus-fähigen Wechselstromzähler integriert, der den Strom für den gesamten Aufbau misst.

Für diese Aufgabe wird das CoDeSys Projekt **wago_mbus.pro** geöffnet. Hier ist die Controller IP-Adresse bereits hinterlegt (192.168.0.110). Der Controller muss entsprechend dieser IP-Adresse eingestellt werden (Programm: EthernetSettings).

# 2 Benötigte Komponenten und Verdrahtung

**Benötigte Hardware-Komponenten:**

1) Hutschienen-Netzteil (z.B. 787-712 – WAGO-Stromversorgung)

*Primär getaktete Stromversorgung, Selbstkühlung durch natürliche Konvektion bei horizontaler Einbaulage, Gekapselt, Parallelschaltbar, reihenschaltbar, Galvanisch getrennte Ausgangsspannung (SELV) gemäß EN 60950-1/UL 60950*

2) Serielle Klemme (z.B. 750-650/003-000 – RS232)

*Die in der Busklemme integrierte Schnittstelle ermöglicht den Anschluss von Geräten mit einer RS-232-C-Schnittstelle. Die Schnittstelle arbeitet normenkonform nach TIA/EIA-F, CCITT V.28/DIN 66259-1. Das angeschlossene Gerät kann über den eingesetzten Buskoppler mit der Steuerung direkt kommunizieren. Der aktive Kommunikationskanal arbeitet unabhängig vom überlagerten Bussystem im Vollduplexbetrieb mit bis zu 19200 Baud*

3) Adapterkabel von seriell auf USB (für die Programmierung der Primäradresse)

4) Pegelwandler (z.B. Relay PW20)

*Funktionsweise der Master-PW Geräte:*

*Die Geräte der PW-Serie sind M-Bus Master Interfaces für Netze mit bis zu 3, 20 oder 60 Endgeräte. Sie zeichnen sich durch eine kompakte Bauform (Wand- oder Schienenmontage) und einem weiten Betriebs-spannungsbereich aus. LEDs an der Frontseite zeigen den aktuellen Betriebszustand an. Alle Geräteversionen der PW-Serie sind mit einer RS232-Schnittstelle ausgestattet. Alternativ kann der Steuerrechner mit einem ZVEI-Optokopf über die optionale IR-opto-Schnittstelle mit den M-Bus Teilnehmern kommunizieren. Um größere Distanzen zwischen dem Steuerrechner und dem Pegelwandler zu ermöglichen, verfügt der PW60 zusätzlich über eine störsichere RS485- Schnittstelle.*

5) M-Bus-Zähler (z.B. NZR DHZ 5/63A Einphasen-Wechselstromzähler)

Die M-Bus Kommunikationsschnittstelle ermöglicht die Fernauslesung der Zählerdaten für Abrechnungs-, Energieoptimierungs-,Visualisierungs- oder Installationsüberwachungszwecke. Diese Version ist mit zwei zusätzlichen Menüpunkten ausgestattet, welche die Primär- und die Sekundäradresse anzeigen.

6) WAGO-Controller (z.B. 750-880 – Ethernetcontroller)

*In Verbindung mit dem WAGO-I/O-SYSTEM kann der ETHERNET-Controller als frei programmierbare Steuerung in ETHERNET-Netzwerken eingesetzt werden. Der Controller unterstützt digitale und analoge Module sowie Sondermodule der Serien 750/753 und eignet sich für die Datenübertragungen von 10/100 Mbit/s. Die zwei ETHERNET-Schnittstellen und der integrierte Switch ermöglichen die Verdrahtung des Feldbusses in Linientopologie. Zusätzliche Infrastrukturelemente wie Switch oder Hub können somit entfallen. Beide Schnittstellen unterstützen Autonegotiation und Auto-MDI(X).*

**Benötigte Software:**

WAGO-Laptop 98/003240 (PW: gapraktikum)

- *CoDeSys* (inkl. MBus-Bibliothek)
- *MBSheet*

# Tutorial: WAGO und M-Bus

**Verdrahtung:**

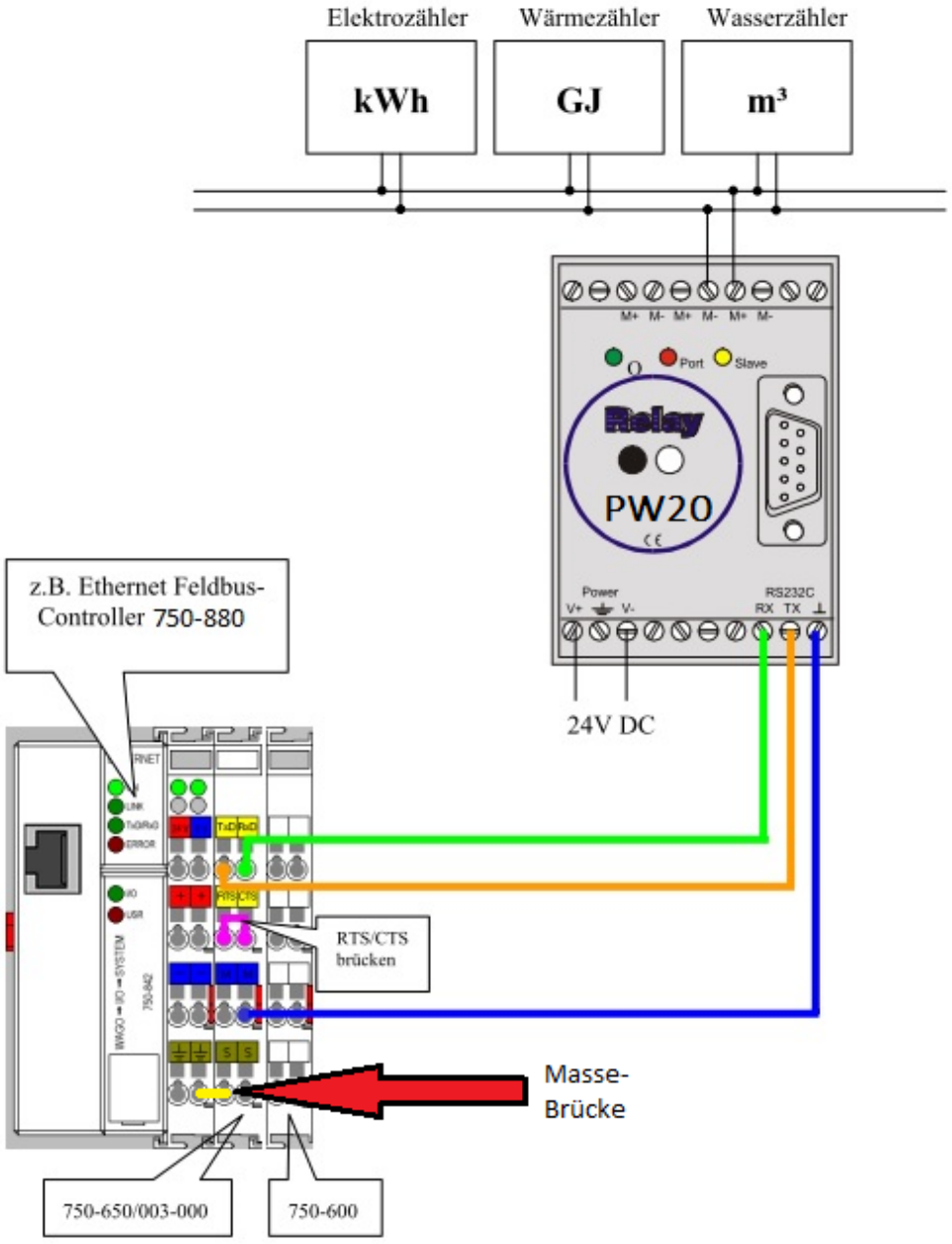



Abbildung 1: Verdrahtung Controller => Pegelwandler => MBus-Zähler

**Kontrollieren ob: (im Normalfall bereits vorhanden)**

Zunächst wird die serielle Klemme (z.B. *750-650/003-000*) in den Controller-Aufbau integriert und die Massebrücke gesteckt. Die serielle Verbindung von Pegelwandler zur seriellen Klemme wird wie in der oberen Abbildung gezeigt, hergestellt. Der 24V-Anschluss des Pegelwandlers wird mit dem Netzteil des Controllers verbunden. Anschließend muss noch der MBus-Zähler per zweiadriger Leitung (Polung ist egal) an die MBus-Klemmen des Pegelwandlers angeschlossen werden.

# 3 Die Primäradresse des Zählers

Über seine Primäradresse kann jeder Zähler vom Controller-Programm identifiziert und angesprochen werden. Pro MBus-Netzwerk stehen max. 255 mögliche Primäradressen zur Verfügung (1 Byte). Die Primäradresse eines Zählers ist somit eine Zahl von 0-254. Jede Adresse darf innerhalb des Netzwerks nur einmal vergeben werden. Kommt eine Adresse mehrfach vor, werden diejenigen Zähler nicht erkannt.

**Programmieren der Primäradresse:**

Werkseitig ist den meisten Zählern die Primäradresse „0“ zugewiesen. Möchte man nun mehrere, zu einem Netzwerk verbundene Zähler auslesen, muss man den ihnen verschiedene Primäradressen einprogrammieren. Dies geschieht über die Software „*MBSheet*“. Bevor die Software gestartet wird, muss der Pegelwandler über das serielle/USB Adapterkabel mit dem PC verbunden werden. Hierfür werden normalerweise spezielle Treiber benötigt. Spannungsversorgung sicherstellen.

Nach erfolgreicher Verbindung Pegelwandler ⇔ PC, wird *MBSheet* gestartet.

Die Software „*MBSheet*“ dient u.a. dazu, einen oder mehrere Zähler zu identifizieren und ihm/ihnen eine eindeutige Primäradresse einzuprogrammieren. Die Software ist leider nur als DEMO Version verfügbar, doch für die Programmierung der Primäradressen ausreichend.

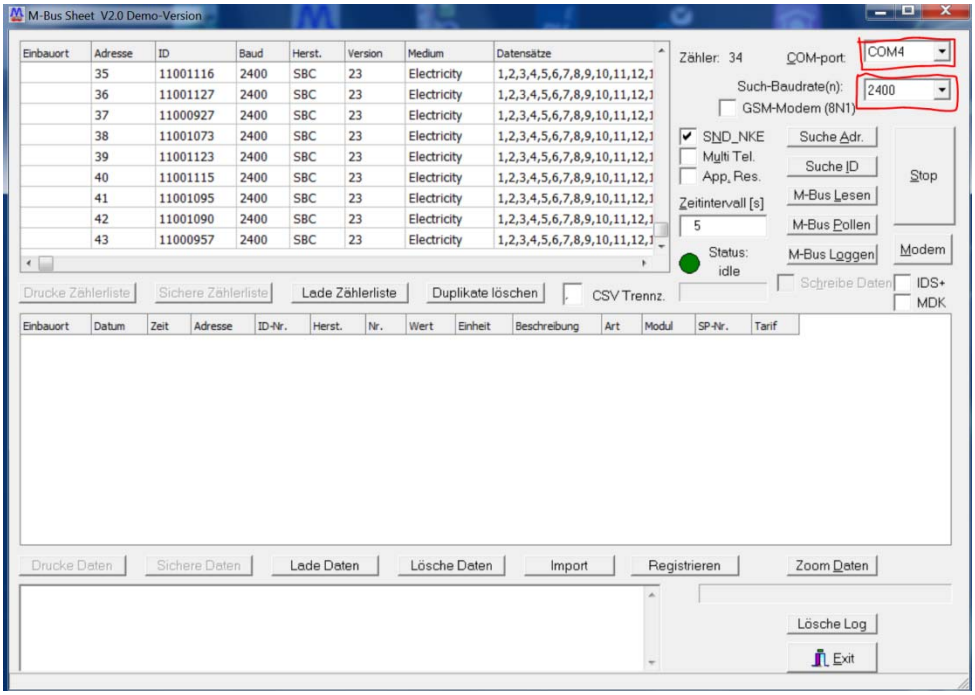



Abbildung 2: Programm MBSheet

Oben rechts im Fenster des Programms wird die Such-Baudrate auf 2400 gestellt. Der ComPort muss ebenfalls ausgewählt werden. Es ist der virtuelle ComPort, der über die seriell/USB-Treiber installiert wurde.

Ist man sich unsicher, welchen Port man auszuwählen muss, doppelklickt man auf das *„Arbeitsplatz“*-Symbol auf dem Desktop oder drückt die Tastenkombination *[Windowstaste]+[E]*. Im erscheinenden „Explorer“ klickt man mit der rechten Maustaste auf *„Computer“* und wählt *„Eigenschaften“* aus. Im folgenden Fenster wählt man den *„Geräte-Manager“* aus und wählt anschließend *„Anschlüsse (COM & LPT)“.*

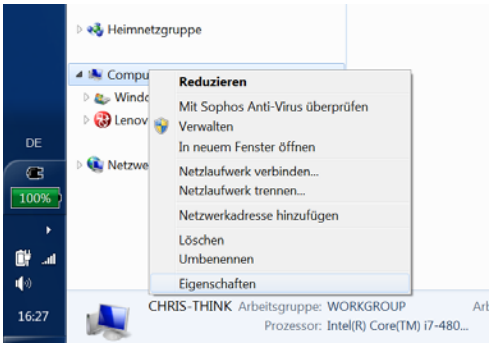



Abbildung 3: Computer => Eigenschaften

Dort kann nun in Erfahrung gebracht werden, welchen virtuellen COM-Port der Treiber für das Adapterkabel „Prolific USB-to-Serial Comm Port (COM…)“ belegt und für das MBSheet Programm gewählt werden muss.

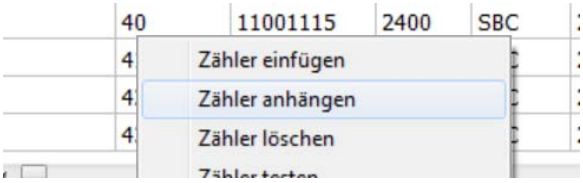



Abbildung 4: Zähler anhängen

Nun kann die eigentliche Programmierung beginnen. Man klickt mit der rechten Maustaste in eine evtl. schon bestehende Zählerliste und wählt *„Zähler anhängen“*.
Nun scrollt man herunter und sieht einen neuen Zähler am Ende der Liste.

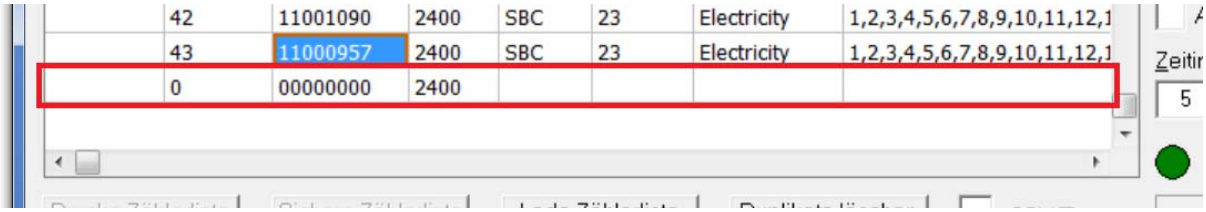



Abbildung 5: neuer Zähler

In die Spalte *„Adresse“* schreibt man nun die gewünschte Primäradresse rein (*in der Regel die nachfolgende Zahl, hier:44*). In die Spalte *„ID“* kommt die 8-stellige Seriennummer des Zählers. Die werksseitig vergebene Seriennummer hat immer 8 Stellen und ist am Zähler vermerkt. Über die Schaltfläche *„Suche ID“* (siehe Abb. 8) kann der Suchvorgang gestartet werden. Dabei öffnet sich ein Fragefenster „Löschen der aktuellen Zählerliste?“, dies kann mit „Ja“ bestätigt werden. Dabei werden alle angeschlossenen Zähler nach Seriennummer sortiert ausgegeben. Dieser Vorgang dauert einige Minuten.

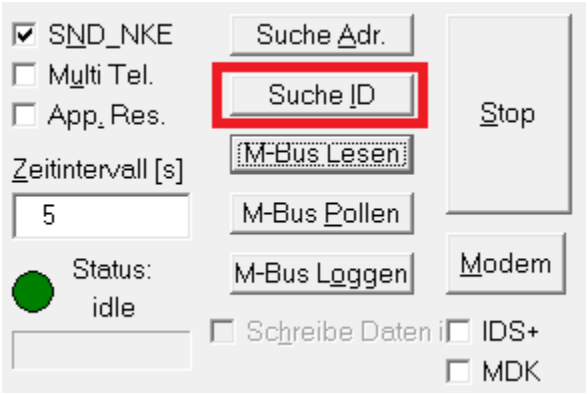



Abbildung 6: Suchvorgang starten, gesucht wird nach den Seriennummern

Anschließend klickt man mit der rechten Maustaste auf den neuen Zähler und wählt *„Setze PAdr via ID“* (siehe Abbildung 7). Hiermit wird dem Zähler über die ID (=Seriennummer) seine Primäradresse einprogrammiert.

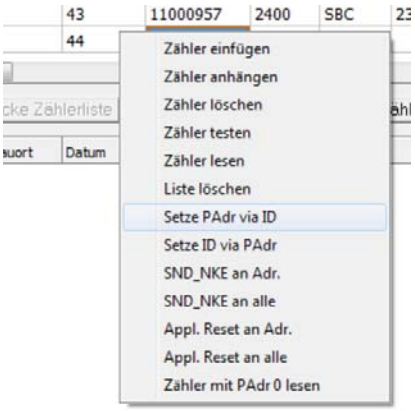



Abbildung 7: Primäradresse programmieren

Nach ein paar Augenblicken ist die Primäradresse programmiert. Durch erneutes rechts-klicken des Zählers öffnet sich das Auswahlfenster wieder und man kann *„Zähler testen“* auswählen. Nun wird der neue Zähler testweise angesprochen um zu sehen, ob er sich identifizieren lässt. Hat alles geklappt, kann das Programm *„MBSheet“* wieder geschlossen werden und das USB-Kabel vom Laptop getrennt werden, da die Zählerstände im Folgenden über das Bus-Kabel übertragen werden.

# 4 Knotenkonfiguration

Die serielle Klemme wird, falls nicht schon geschehen, analog zu den vorher im Tutorial genannten Zusatz-Klemmen in die bestehende WAGO Konfiguration eingefügt. Wird die Knotenkonfiguration mit dem Programm *„WAGO-I/O-Check 3“* (siehe Wago-Tutorial Kapitel 5) durchgeführt, kann die generierte XML-Datei entsprechend dem Wago-Tutorial-Skript Kapitel *6.4 „Knotenaufbau“* im *CoDeSys* Projekt unter *„Steuerungskonfiguration“* importiert werden. Eine andere Möglichkeit ist es, die Klemme manuell hinzuzufügen.

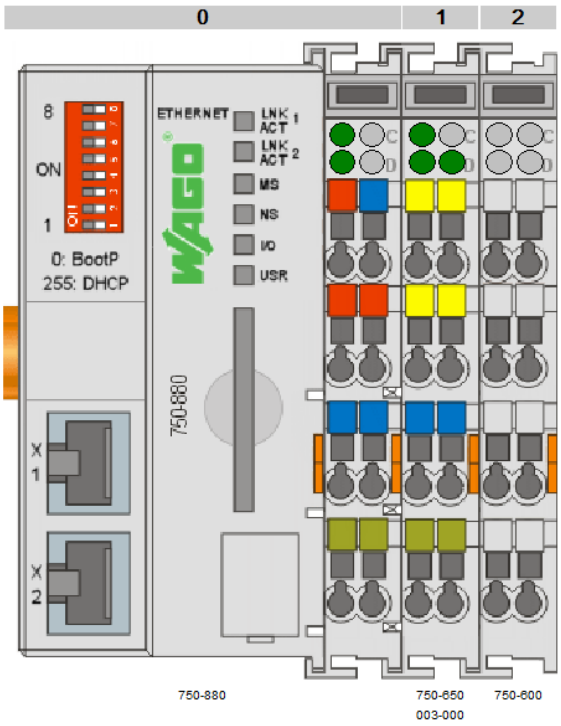



Abbildung 8: WAGO Knotenkonfiguration mit neuer serieller Klemme

Im nächsten Schritt wird unter *„Bibliotheksverwalter“* in *CoDeSys* eine weitere Bibliothek eingefügt. Die benötigte Bibliothek namens *„MBus_03.lib“* befindet sich im Ordner *„…/Libraries/Building“* (analog zum Kapitel „Einfache Demoprogramme – Schritt 07.10: Verwendung von Bibliotheksbausteinen“ im Wago-Tutorial!) oder muss zuvor über die *WAGO*-Website heruntergeladen werden (http://www.wago.com).

# 5 Programmierung in CoDeSys

Nun wird das eigentliche Programm erstellt: im Reiter „Bausteine – PLC_PRG“

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     MBus_Master     :   FbMBusMaster;
0004     MBus_Elektro    :   FbMBus_Electricity;
0005     typMBus         :   typMBus;
0006     MBus_Info       :   typMBusInfo;
0007     PAdrr           :   BYTE  :=  44;
0008     start           :   BOOL  :=  TRUE;
0009
0010     UC_Zaehlerstand :   FbUnitConverter;
0011     UC_Leistung     :   FbUnitConverter;
0012     Error           :   BYTE;
0013     Zaehlerstand    :   REAL;
0014     Leistung        :   REAL;
0015 END_VAR
```

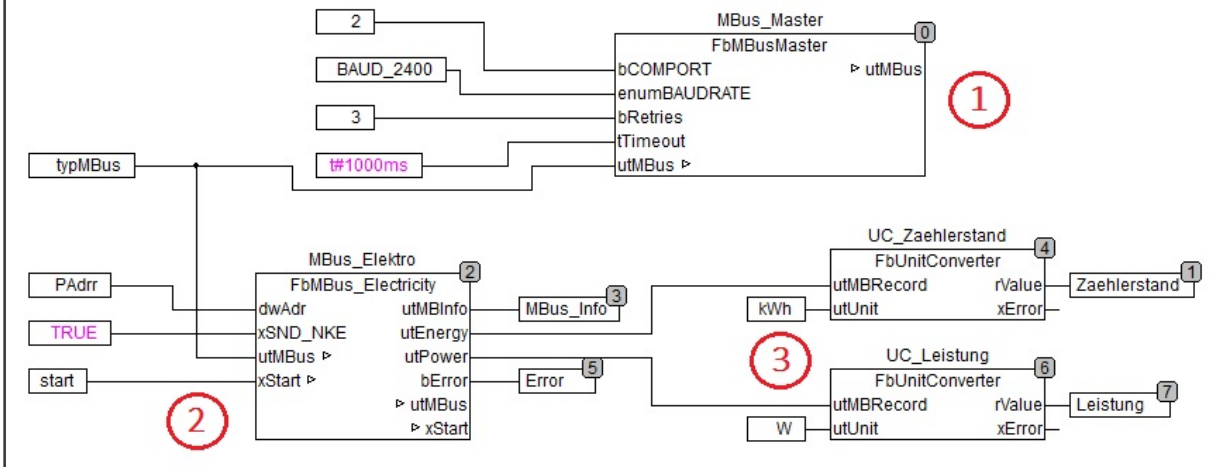



Abbildung 9: CoDeSys-Programm zur Auslesung eines MBus-Elektrozählers

Im Hauptprogramm PLC_PRG (Programmiersprache CFC - Continuous Flow Chart verwenden!) werden folgende Funktionsbausteine (Fb‘s) eingefügt:

- ***FbMBusMaster*** (neuen Baustein einfügen ( *[STRG]+[B]* ) => auf das Textfeld im Baustein klicken *[F2]* => *Standard-Funktionsblöcke* => C:\PROGRAMM FILE …*MBus_03.lib* => *FbMBusMaster*).
  Dieser wird als **MBus_Master** benannt.
- ***FbMBus_Electricity*** (neuen Baustein einfügen ( *[STRG]+[B]* ) => auf das Textfeld im Baustein klicken *[F2]* => *Standard-Funktionsblöcke* => C:\PROGRAMM FILE …*MBus_03.lib* => *FbMBus_Electricity*).
  Dieser wird als **MBus_Elektro** benannt.
- *2x* ***FbUnitConverter*** (neuen Baustein einfügen ( *[STRG]+[B]* ) => auf das Textfeld im Baustein klicken *[F2]* => *Standard-Funktionsblöcke* => C:\PROGRAMM FILE …*MBus_03.lib* => *FbUnitConverter*).
  Diese werden als **UC_Zaehlerstand** und **UC_Leistung** benannt

Die Eingänge/Ausgänge mit jeweils STRG+E/STRG+A eingefügt. Anschließend gibt man den Namen des Eingangs/Ausgang laut Abbildung 10 ein. Es öffnet sich automatisch ein Fenster „Variablendeklaration“, hier muss der richtige Typ (Abbildung 10) ausgewählt werden. Bei „PAdrr“ sowie bei „start“ muss ein zusätzlich ein Initialwert eingegeben werden. Der Initialwert bei „PAdrr“ ist die Primäradresse die man im Schritt 1.3 gewählt hat. Bei „start“ ist als Initialwert „TRUE“ einzugeben.

# Tutorial: WAGO und M-Bus

**Infos zu den Bausteinen**

**MBus_Master**: Der Masterbaustein dient zur Kommunikation mit der seriellen Klemme. Dieser Baustein wird einmalig für alle evtl. folgenden MBus-Funktionsblöcke benötigt (z.B. *FbMBus_Electricity*, *FbMBus_Heat* usw.
Am Eingang „***bCOMPORT***“ wird die Portnummer eingestellt - für die erste serielle Klemme am Controller liegt der Wert bei „*2*“, für evtl. weitere serielle Klemmen dann entsprechen 3, 4, 5 usw. *Nicht der COMPort an dem der MBus-Zähler angeschlossen wurde!*
Am Eingang „***enumBAUDRATE***“ wird die Baudrate, also die Übertragungsgeschwindigkeit für den Auslesevorgang der MBus-Zähler eingestellt. Hier wird „*BAUD_2400*“ gesetzt, da viele MBus-Zähler keine höhere Auslesegeschwindigkeit unterstützen.
Der angeschlossene Wert für „***bRetries***“ gibt an, wie viele Verbindungsversuche unternommen werden, falls die Verbindung zum Zähler erfolglos war.
Am Eingang „***tTimeout***“ wird festgelegt, wie lange versucht wird, eine Verbindung zum Zähler aufzubauen. Der Standardwert sollte bei einer Sekunde liegen (=> t#1000ms)
„***utMBus***“ ist eine Ein-/Ausgangsvariable, die für die Kommunikation der MBus-Bausteine untereinander benötigt wird. An diese wird die Variable „***typMBus***“ vom gleichnamigen Variablentyp angeschlossen. Alle MBus-Funktionsblöcke müssen über diese Variablenstruktur verbunden werden, um die Synchronisierung der Bausteine zu gewährleisten.

**MBus_Elektro**: Dieser Funktionsblock dient zum Auslesen eines Elektrozählers.
Am Eingang „***dwAdr***“ wird die Primäradresse eingestellt.
Wird „***xSND_NKE***“ auf „*TRUE*“ gesetzt, wird beim Auslesen des Zählers immer das erste Telegramm vom Zähler gesendet. Bei Zählern mit vielen verschiedenen Messwerten kann es sein, dass die Werte in zwei oder drei Telegramme aufgeteilt werden. Da beim verwendeten Elektrozähler ohnehin nur ein Telegramm gesendet wird, könnte der Wert für „*xSND_NKE*“ in diesem Fall auch auf „*FALSE*“ gesetzt werden.
„***utMBus***“ wird über die Variablenstruktur „*typMBus*“ mit dem Masterbaustein verbunden.
An den Eingang „***xStart***“ wird die boolesche Variable „*start*“ angeschlossen. Für dieses einfache Beispiel sollte die Variable standardmäßig auf „*TRUE*“ gesetzt werden. Dadurch wird der Zähler beim Betrieb ständig ausgelesen und die Mess-/Zählerwerte aktualisiert.
Der Ausgang „*utMBInfo*“ liefert detaillierte Informationen über den Auslesevorgang, sowie alle Daten über den ausgelesenen Zähler. Diese Infos werden in der Variablen „*MBus_Info*“ gespeichert. Für den Betrieb des Programms ist die angeschlossene Variable nicht essentiell.
Die Ausgänge „***utEnergy***“ und „***utPower***“ liefern den aktuellen Zählerstand (*utEnergy*), sowie die momentan anliegende Leistung (*utPower*). Je nach Zählertyp und Hersteller, kommen diese Werte im STRING-Format in unterschiedlicher Einheit an (z.B. in kW, W, MW oder auch mW). Um die gewünschte Einheit zu erhalten, werden die Ausgänge noch an einen UnitConverter (Einheiten-Konvertierer) angeschlossen.
„***bError***“ gibt einen Zahlenwert aus (0-7), der einem Fehlercode entspricht, falls beim Auslesevorgang Fehler aufgetreten sind. Die Fehlercodes können anhand der „*Bausteinbeschreibungen für M-Bus*“ (PDF-Datei, zu finden unter http://www.wago.com ) identifiziert werden. „0“ bedeutet: kein Fehler.

***UC_Zaehlerstand*** und **UC_Leistung** sind die bereits erwähnten Einheiten-Konvertierer. Die Eingänge „***utMBRecord***“ werden mit den Ausgängen „*utEnergy*“ und „*utPower*“ verbunden.

Über die Eingänge „***utUnit***“ wird festgelegt, in welche Einheiten bzw. in welchen Dimensionen die Messwerte ausgegeben werden sollen. Dabei sind z.B. folgende Einheiten und Dimensionen möglich:

mi = milli, k = kilo, M = Mega, G = Giga
ps = pro Sekunde, pmin = pro Minute, ph = pro Stunde
l = Liter, m3 = m³, W = Watt, J = Joule
HCA = ohne Einheit, none = ungültig

Entsprechend sind auch (*sinnvolle*) Kombinationen möglich, z.B.:

kWh = Kilowattstunden,
miJps = Millijoule pro Sekunde,
gpmin = Gramm pro Minute,

„***Zaehlerstand***“ und „***Leistung***“ sind Variablen im *REAL*-Format, in die die konvertierten Werte geschrieben werden. Sie müssen an die Ausgänge „***rValue***“ angeschlossen werden.

Nun kann das Programm wie gewohnt auf den Controller geschrieben werden. Voraussetzungen sind wie immer die richtigen Kommunikationsparameter (z.B. die IP-Adresse und Portnummer).
Über „*Online*“ => „*Einloggen*“ wird der Controller programmiert, anschließend wird im selben Menü „*Werte forcen*“ eingestellt und das Programm mit „*Start*“ gestartet. Durch „W*erte forcen*“ wird sichergestellt, dass der Zähler ständig ausgelesen wird und die Werte immer aktuell sind.

# 6 Visualisierung in Codesys

Im Reiter „Visualisierung“ wird zunächst ein neues Objekt eingefügt.

Dazu wird ein Rechteck mit folgenden Parametern eingefügt. Über Rechtsklick -> „Konfigurieren“ -> „Text“

*Leistung:*

Text: Die aktuelle Leistung beträgt:

%2.lf W *(Absatz mit Strg + Enter)*

Variablen → Textausgabe: PLC_PRG.Leistung

*Zählerstand:*

Text: Der aktuelle Zählerstand beträgt:

%2.4lf W *(bei 4 -> Nachkommastellen)*

Variablen → Textausgabe: PLC_PRG.Zählerstand

Nun werden die Leistung und der Zählerstand in der Visualisierung dargestellt.

# 7 Integration eines Temperaturfühlers

Im Folgenden wird nun erklärt wie ein PT-Fühler in Codesys integriert werden kann.

Zuerst wird der PT100 mit dem speziell dafür vorgesehenen WAGO-Controller (0750-0646#02) verbunden.

*Hinweis: Die Eingangsklemme erlaubt den direkten Anschluss von Pt- und Ni-Widerstandssensoren, sowie Potentiometern. Sie kann als 2-Kanal- (2- und 3-Leiter-Technik) oder 4-Kanal- (2-Leiter-Technik) Klemme betrieben werden. Die Linearisierung über den gesamten Temperaturbereich übernimmt die Busklemme. Ein Kurzschluss oder die Unterbrechung der Sensorleitung sowie eine Bereichsüberschreitung wird durch eine rote Fehler-LED angezeigt. Die Klemme ist frei konfigurierbar über WAGO-I/O-CHECK und GSD-Dateien. Vielfältige Einstellmöglichkeiten und die hohe Genauigkeit zeichnen sie aus. Die Variante 750-464/020-000 bietet die Möglichkeit zum Anschluss von NTC-Sensoren. Abweichende technische Daten für 750-464/020-000:*

- *Anzahl der Eingänge: 4*
- *Sensorarten: NTC 10 kOhm, NTC 20 kOhm, NTC 10 kOhm (Thermokon)*
- *Sensoranschluss: 2-Leiter*
- *Temperaturbereich: -30 °C … +120 °C*
- *Messfehler: ≤ 2 K über den gesamten Temperaturbereich*

1. Zuerst wird Codesys gestartet. Dabei kann ein neues oder das voranstehende Projekt geöffnet werden (Klemme muss in jedem Fall in der Konfiguration enthalten sein).
2. Eine Überprüfung ob die Klemme in der Konfiguration enthalten ist sollte in jedem Fall durchgeführt werden. In der Steuerungskonfiguration wird nun „Ch_1 signed Input value“ (erster Wert der Klemme 750-464) mit „PT100“ benannt.
3. Folgende Bausteine werden in der Programmierung benötigt:
   a. Im „Bausteine“-Fenster wird ein REAL-Eingang mit dem Namen „PT100“ erstellt (mit F2 muss dieser wie gehabt dem vorher benannten „PT100“ zugeordnet werden).
   b. Zusätzlich werden ein Eingang, welcher der Wert „10.0“ zugeordnet wird, sowie ein DIV-Baustein eingefügt.
   c. Ausgangsbaustein „PT100_C“
4. Ordnen Sie nun die Bausteine wie folgt an:

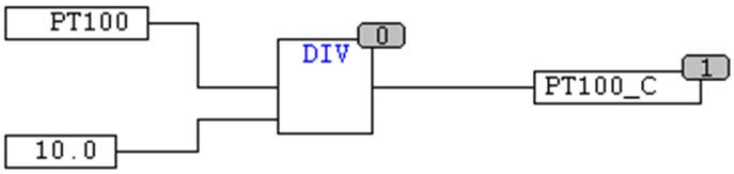



Abbildung 10: Einfügen der Bausteine

5. Nun wird die Visualisierung hinzugefügt:
   Einfügen einer Form mit dem Text „Die Innentemperatur beträgt %2.2lf °C“.
   Unter Variablen wird im Punkt Textausgabe der „PT100_C“ ausgewählt (also PLC_PRG.PT100_C).
   Nach dem Einfügen in das voranstehende Projekt sollte die Visulisierung folgendermaßen aussehen:

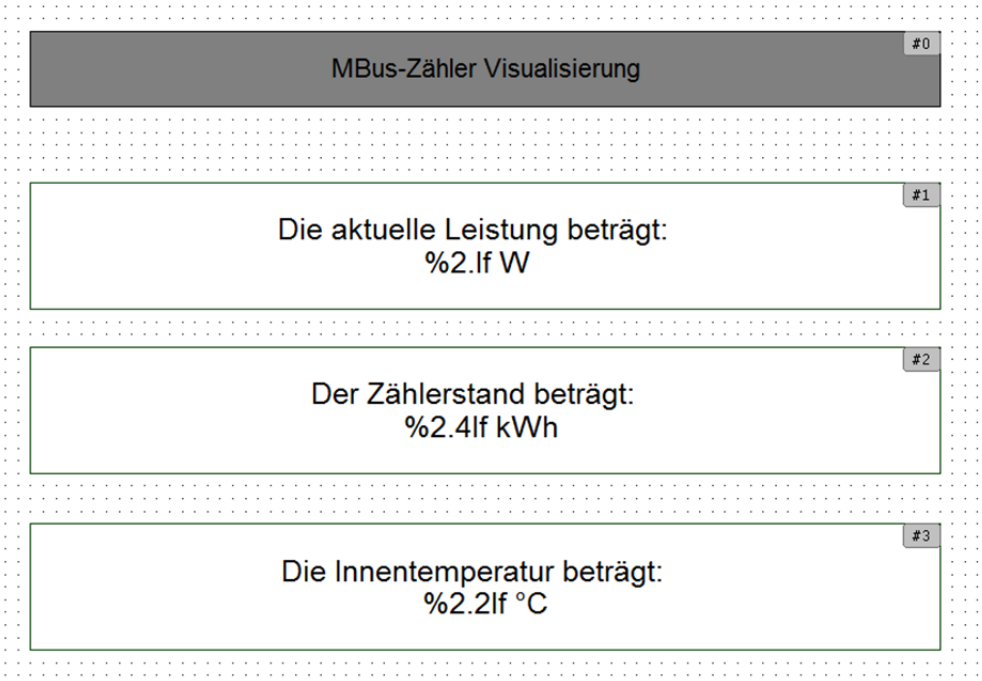



Abbildung 11: Ansicht Visualisierung

6. Nun kann man unter „Online -> Einloggen“ sowie „Start“ das Programm auf den Controller laden und die aktuellen Werte des PT100 sollten in der Visualisierung angezeigt werden.