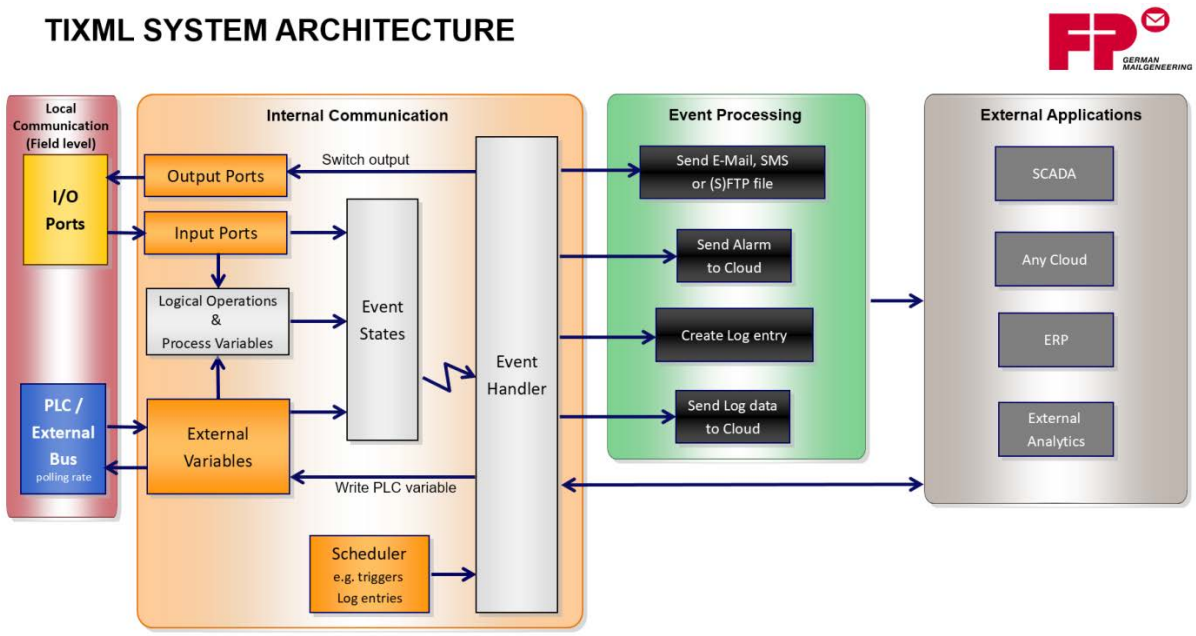
FP SPS TiXML Handbuch



Für die Linux-Serie: OTGuard Hx600
ENGuard Hx600
ENGuard Wx600
ENGuard Wx550

Für Vorgänger-Serie: Hx100/400

Version: 3.7.1

www.inovolabs.com

Redaktionsschluss: 23.02.2021

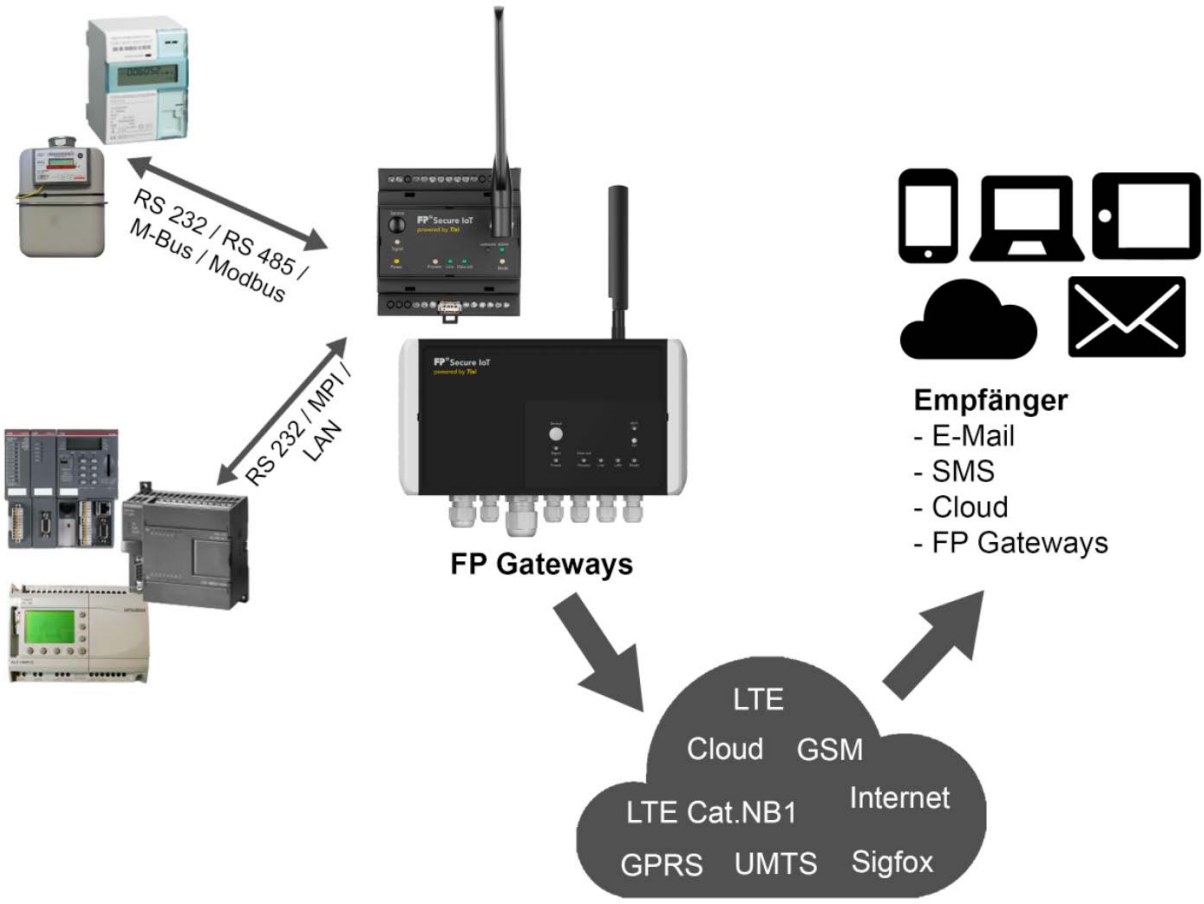
Inhaltsverzeichnis

1 Übersicht

FP IoT Gateways lassen sich ohne großen Aufwand in bestehende Systeme einbinden („Retrofit“).

Die Kommunikationsprotokolle verbreiteter SPS-Systeme sind bereits in FP IoT Gateways vorhanden, somit ist keine Anpassung in der SPS notwendig. Andere SPS-Systeme können das FP-Gateway über einfache TiXML-Textbefehle steuern.

Eine beispielhafte Anwendung für FP-Gateways:



Dieses Handbuch beschreibt die notwendigen Einstellungen zum Anschluss an eine SPS. Zusätzlich stellt es eine Übersicht der derzeit unterstützten SPS-Systeme und deren Variablenumfang, sowie Befehle zur Formatierung von SPS-Variablenwerten dar.

Die meisten der hier beschriebenen Protokolle und Steuerungen gelten für FP IoT Gateways ab der sechsten Generation einschließlich der Wand.Box. Das SML-Protokoll ist nur auf FP IoT Gateways der achten Generation verfügbar.

2 FP IoT Gateways für SPS-Systeme parametrieren

2.1 SPS- und Bussysysteme konfigurieren

Um FP IoT Gateways mit der SPS kommunizieren zu lassen, muss die External-Datenbank des FP IoT Gateways parametriert werden. Diese Beschreibung setzt die grundlegende Kenntnis der TiXML-Sprache und des FP IoT Gateways voraus.

Die Datenbank der SPS-Konfiguration ist `PROCCFG/External` und sieht wie folgt aus:

```
<External>
   <Bus _="Bus" BusId="BusId" {Name="Alias"} protocol="Proto"
    type="BType" {baud="Speed"} {format="Dataformat"}
    {handshake="handshake"} {Mem="Memory"} [TS="OwnID"]
    MAXADR="Range"] [GUF="Factor"] [RC="Retries"]>
    {<Condition _="Name" Variable="Path" Pollrate="Rate"/>}
       <Device _="ID" {Name="Alias"} {Pollrate="Rate TUnit"}
        [CharTimeout="CT TUnit "] [Pause="Wait TUnit "]
        [Timeout="Timeout TUnit"] [DWordInc="AI"] [DwordSwap="Swap"]
        [ForceSingleWordWrite="Funct"] [PrimaryAddr="PA"]
        [SecondaryAddr="SA"] [FabricationAddr="FA"]
        [ManufactoryCode="MC"] [Generation="Gen"] [Medium="Med"]
        [devType="DType"] [UseCache="Cache"]
        [MaxElements="Elements"] {Condition="Name"}>
           <VName _="VType" {simpleType="BasicType" } [exp="Exp"]
           [precision="Precision"] [size="ArraySize" ] acc="Rights
           Storage" [ind="Index"] [subind="Index2"] [no="Array"]
           {def="default"} {multip="Factor"}  {format="Format"}
           {write="wFunct"} {read="rFunct"}/>
      </Device>
   </Bus>
</External>
```

Attribute in {…} sind optional.
Attribute in […] werden nur für bestimmte Geräte oder Bussysteme benötigt.

Übersicht der möglichen Parameter:

<Bus>-Parameter

Bus Definiert die Schnittstelle, an der die SPS angeschlossen ist. Mögliche Werte:

COM1	SPS an COM1 (Nur Hx-Geräte ab FW 1.80.0.0, erfordert BusId)
COM2	SPS an COM2 (Nur Hx-Geräte ab FW 1.80.0.0, erfordert BusId)
COM3	SPS an COM3 (normalerweise M-Bus Geräte; nur bestimmte Geräte)
COM4	SPS an COM4 (RS485 Geräte; nur bestimmte Geräte)
COM5	SPS an COM5 (RS485 Geräte; nur bestimmte Geräte)
ETH	SPS an LAN Interface (Modbus TCP, Siemens S7)

BusId Ermöglicht die Adressierung des SPS-Bus unabhängig von der Schnittstelle.
Darf nicht mir „Name“ vermischt werden.

Alias Ermöglicht die Benennung des SPS-Busses und dadurch eine schnittstellenunabhängige Adressierung. Darf nicht mir „BusId“ vermischt werden. (max. 20 alphanumerische Zeichen, keine Sonderzeichen, darf nicht mit einer Zahl beginnen)

Proto Bestimmt das SPS-Protokoll oder Bus-Protokoll.
Abhängig vom angeschlossenen Gerät, SPS oder Bussystem.
Die gültigen Werte können Kapitel 3 entnommen werden.

BType Bestimmt den Kommunikationsmodus gegenüber der SPS: `Master` oder `Slave`. Die gültigen Werte können Kapitel 3 entnommen werden.

Speed Bestimmt die Baudrate in bps zwischen dem FP IoT Gateway und der SPS bzw. beschreibt die Baudrate auf dem Bussystem (z.B. `19200`). Die gültigen Werte können Kapitel 3 entnommen werden.

Dataformat Bestimmt das Datenformat auf der seriellen Schnittstelle. Das Datenformat wird abhängig von der verwendeten SPS oder dem Bussystem (z.B. 8E1 bei S7-200) gewählt. Die gültigen Werte können Kapitel 3 entnommen werden.

Syntax: DataBitsParityBitsStopBits

DataBits:

- 8..8 Datenbits
- 7..7 Datenbits

ParityBits:

- N..kein Paritätsbit
- E..gerade Parität
- O..ungerade Parität

StopBits:

- 1..1 Stopbit
- 2..2 Stopbits

Handshake Bestimmt das Software- oder Hardware-Handshake zwischen Gerät und SPS oder Bussystem.

None	Kommunikation ohne Handshake
XONXOFF	Software Handshake
XONXOFFPASS	Software Handshake, XONXOFF wird an Anwendung weitergeleitet
RTSCTS	Hardware Handshake mit RTS CTS
DTRDSR	Hardware Handshake mit DTR DSR
HALF	Halbduplex RS 485 Kommunikation
FULL	Vollduplex RS 485/422
HALFX	Halbduplex RS 485 Kommunikation mit XON XOFF
FULLX	Vollduplex RS 485/422 mit XON XOFF
noDTR	deaktiviert die DTR

Memory Definiert, wieviel Speicher (in byte) für den Bus reserviert wird. Insgesamt stehen für alle möglichen Bus-Definitionen bis zu 20.000.000 byte (20 MB, nur bei Linux-Geräten ab FW 5.1.6.8) zur Verfügung.

OwnID Legt die Stationnummer des Geräts fest. Ob dieses Attribut verwendet wird und welche Werte gültig sind kann aus Kapitel 3 entnommen werden.

Range Maximal abzufragende Stationensnummern. Ob dieses Attribut verwendet wird und welche Werte gültig sind kann aus Kapitel 3 entnommen werden.

Factor "Gap update factor" zur Erkennung weiterer Slaves. Ob dieses Attribut verwendet wird und welche Werte gültig sind, kann aus Kapitel 3 entnommen werden.

Retries Anzahl Wiederholungen bei Kommunikationsfehlern. Ob dieses Attribut verwendet wird und welche Werte gültig sind kann aus Kapitel 3 entnommen werden.

Je nach Modell können bis zu 5 verschiedene SPS-Systeme definiert werden. Wird eine der Schnittstellen für MPI-Kommunikation genutzt, so muss dieser an erster Stelle in der External parametriert sein.

<Device>-Parameter

ID Legt die Stationsnummer des abzufragenden Gerätes oder SPS fest. Diese muss mit der in der SPS-Station (Geräte-Station) parametrierten Kennung übereinstimmen, (z.B. 1). Einige Bus-Systeme erlauben mehrere Stationen (Netzwerk).

Alias Ermöglicht die Benennung der SPS-Station und dadurch eine Stationsnummernunabhängige Adressierung.

Rate Abfragezyklus des FP IoT Gateways im Master-Mode bzw. Kommunikationstimeout im Slave-Mode in der angegeben Zeiteinheit (siehe *TUnit*, default ist 60s).

CT Timeout zwischen den Zeichen in der angegeben Zeiteinheit (siehe *TUnit)* Ob dieses Attribut verwendet wird und welche Werte gültig sind kann aus Kapitel 3 entnommen werden.

Wait Pause zwischen den Nachrichten in der angegeben Zeiteinheit (siehe *TUnit)* Ob dieses Attribut verwendet wird und welche Werte gültig sind kann aus Kapitel 3 entnommen werden.

Timeout Timeout für Antwort in der angegeben Zeiteinheit (siehe *TUnit)* Ob dieses Attribut verwendet wird und welche Werte gültig sind kann aus Kapitel 3 entnommen werden.

AI AdressIncrement zwischen zwei aufeinanderfolgenden DWORDSs (2, nur Modbus RTU). Ob dieses Attribut verwendet wird und welche Werte gültig sind kann aus Kapitel 3 entnommen werden.

Swap Muss gesetzt werde, wenn Low vor High Word in DWORD gesendet wird (0, nur Modbus RTU). Ob dieses Attribut verwendet wird und welche Werte gültig sind kann aus Kapitel 3 entnommen werden.

Funct Setzen, wenn Funct 0x06 anstelle von 0x10 für einzel WORD Schreiben verwendet werden soll (0, nur Modbus RTU). Ob dieses Attribut verwendet wird und welche Werte gültig sind kann aus Kapitel 3 entnommen werden.

PA Primäradresse (Nur M-Bus). Ob dieses Attribut verwendet wird und welche Werte gültig sind kann aus Kapitel 3 entnommen werden.

SA Sekundäradresse (Nur M-Bus). Ob dieses Attribut verwendet wird und welche Werte gültig sind kann aus Kapitel 3 entnommen werden.

FA Fabrikationsadresse (Nur M-Bus). Ob dieses Attribut verwendet wird und welche Werte gültig sind kann aus Kapitel 3 entnommen werden.

MC Herstellercode (Nur M-Bus). Ob dieses Attribut verwendet wird und welche Werte gültig sind kann aus Kapitel 3 entnommen werden.

Gen Gerätegeneration (Nur M-Bus). Ob dieses Attribut verwendet wird und welche Werte gültig sind kann aus Kapitel 3 entnommen werden.

Med Gerätemedium (Nur M-Bus). Ob dieses Attribut verwendet wird und welche Werte gültig sind kann aus Kapitel 3 entnommen werden.

DType CPU-Typ (Nur Mitsubishi FX). Ob dieses Attribut verwendet wird und welche Werte gültig sind kann aus Kapitel 3 entnommen werden.

Cache Ist der Wert auf 0 gesetzt, wird das Zusammenfassen von aufeinanderfolgenden Variablen zu blockweisen Abfragen (Caching) deaktiviert. Alle Variablen werden in einzelnen Abfragen geholt. (Default: 1, nur Modbus RTU). Ob dieses Attribut verwendet wird und welche Werte gültig sind kann aus Kapitel 3 entnommen werden.

Elements Anzahl der Variablen die pro Modbus-Telegramm abgefragt werden (caching). Ob dieses Attribut verwendet wird und welche Werte gültig sind kann aus Kapitel 3 entnommen werden.

Name Name der Bedingung zur Änderung der Pollrate.

TUnit Zeiteinheit für Zeitangaben (ms, s, m, h).

<Condition>-Parameter

Conditions werden verwendet, um die Pollrate abhängig von einer Bedingung zu verändern

Name Name der Bedingung, auf die im Device-Abschnitt verwiesen wird.

Path Variable, durch welche die Bedingung als erfüllt gilt (=1). Wird ohne Referenz-Zeichen eingegeben.

Rate Abfragezyklus des FP IoT Gateways im Master-Mode bzw. Kommunikationstimeout im Slave-Mode (z.B. `1s,2m,6h`)

<Variablen>-Parameter

Aus Performancegründen wird empfohlen, nicht mehr als 1000 Variablen zu definieren. In einer Station (Device) darf es keine doppelten Variablen mit identischem Typ und Index geben (z.B. zweimal Merker 2 mit unterschiedlicher Formatanweisung).

VName Name der im Folgenden parametrierten Variable. Der Name darf bis 30 Zeichen lang sein.

VType Bestimmt den Variablentyp der SPS bzw. des Busprotokolls, z.B. einen Zähler oder Merker. Die gültigen Typen können in Kapitel 3 nachgeschlagen werden.

BasicType Bestimmt den Basistyp der Variablen, wie er in der jeweiligen Anwendung verwendet wird. Der Basistyp bestimmt die zugehörige Formatierungs-möglichkeiten und die native Darstellung im TiXML-Protokoll. Einem Variablen Typ (Attribute = "_") können mehrere Basistypen (Attribut „simpleType") zugeordnet sein von denen je Variableneintrag eine ausgewählt werden muss. Für verschiedene Basistypen müssen weitere Attribute angegeben werden (siehe „exp", „size"). Folgende Werte sind möglich:

BasicType Wert	Bedeutung	Weitere Attrib.	native Darstellung Beispiele
Uint8	Vorzeichenloser 8 Bit Wert (0…255)	exp	“123” (exp = 0) “12.3” (exp = -1) “12300” (exp = 2) “0.0123” (exp = -4)
Uint16	Vorzeichenloser 16 Bit Wert (0…65535)	exp	“12345” (exp = 0) “1234.5” (exp = -1) “1234500” (exp = 2)
Uint32	Vorzeichenloser 32 Bit Wert (0…4294967295)	exp	“1234567” (exp = 0) “123456.7” (exp = -1) “123456700” (exp = 2)
Int8	Vorzeichenbehafteter 8 Bit Wert (-128…+127)	exp	„123“ (exp=0) „-123“ (exp=0) “1.23” (exp=-2) “12300” (exp=2)
Int16	Vorzeichenbehafteter 16 Bit Wert (-32768…32767)	exp	“-12345” (exp = 0) “ 12345” (exp = 0) “-1234.5” (exp = -1) “ 1234.5” (exp = -1) “-1234500” (exp = 2) “ 1234500” (exp = 2)
Int32	Vorzeichenbehafteter 32 Bit Wert (-2147483648…2147483647)	exp	“-1234567” (exp = 0) “ 1234567” (exp = 0) “-123456.7” (exp = -1) “ 123456.7” (exp = -1) “-123456700” (exp = 2) “ 123456700” (exp = 2)
String	Text (0…size Zeichen)	size	„Das ist ein Text „ (nur die ersten 100 Zeichen)
Blob	Binärdaten Array (0...size Byte) currently not supported	size	“AF037FFF” Byteweise hexadezimal (nur die ersten 100 Byte)
Bit	Digitale Wert (0...1)		„0“ „1“
Float	Gleitkommazahl einfache Genauigkeit ($\pm 3.402823466 * 10^{38}$)	exp	12.34567 -0.001234 -0.123456 E-7 0.123456 E+7 (Exponentialdarstellung bei |Exponent| > 6)
Double	Gleitkommazahl doppelte Genauigkeit ($\pm 1.7976931348623158 * 10^{308}$)	exp	12.345678910 -0.0012345678 -0.1234567890 E-11 0.1234567890 E+11 (Exponentialdarstellung bei |Exponent| > 11)

Rights Legt die Zugriffsrechte des FP IoT Gateways auf die Variable fest:

R Lesezugriff
W Schreibzugriff
RW Lese-/Schreibzugriff

Die gültigen Werte dieses Attributwertes sind vom Variablentyp abhängig und können Kapitel 3 entnommen werden.

Storage Legt weitere Speicher und Zugriffsoptionen fest:

L (RWL) gemeinsamer Speicher (nur bei S-Bus Variablen mit Lese-Schreibzugriff)

A (RA, WA, RWA) Aktiviert aktiven Variablenzugriff (Siemens)

C Aktiviert den gecacheten Zugriff auf Variablenblöcke, falls der generelle Cache via 'UseCache' (s.o.) deaktiviert wurde. Ist das Flag gesetzt, wird die betreffende Variable nicht sofort abgefragt, sondern geprüft, ob die folgende auch noch in der Abfrage mitgelesen werden kann. Hat diese einen anderen Typ, dann erfolgt die Abfrage trotzdem einzeln.

Für verschiedene SPS bzw. Geräte werden weitere Zugriffsoptionen definiert. Die gültigen Werte dieses Attributswertes sind vom Variablentyp abhängig und können Kapitel 3 entnommen werden.

Index Variablenadresse. Sie ist abhängig vom gewählten Variablentyp und muss mit der Parametrierung des Geräts bzw. Busses übereinstimmen. Die gültigen Adressen können Kapitel 3 entnommen werden.

Index2 Variablensubadresse ist abhängig vom gewählten Variablentyp und muss mit der Parametrierung des Geräts bzw. Busses übereinstimmen. Die gültigen Adressen können Kapitel 3 entnommen werden.

Array Anzahl der Elemente, die als Array abgefragt werden.

Default Startwert der Variable. Bei Variablen mit Schreibzugriff wird dieser bei jedem (!) Systemstart in die SPS geschrieben. Bei Variablen mit Lesezugriff wird der Wert beim Systemstart verwendet, bis das Gerät den tatsächlichen Wert aus der SPS erhält. Der Startwert muss passend zum „exp" und simpleType angegeben werden (siehe Tabelle in Kapitel 2.3).

ArraySize 1. simpleType = String *(siehe BasicType)*
Die maximale Anzahl von ASCII Zeichen in einem Textwert (nur gültig für den Typ simpleType="String", 0…65535). Bei null-terminierten Strings muss das Null-Zeichen mit berechnet werden. Der Wert ist abhängig vom Gerätetyp bzw. vom Busprotokoll (optional, abhängig vom Basistyp, Gerät bzw. Bussystem).

2. simpleType= Blob (siehe *BasicType*)
Die maximale Anzahl von Bytes in einem Bytearray (nur gültig für den Typ simpleType="Blob", 0…65535). Der Wert ist abhängig vom Gerätetyp bzw. vom Busprotokoll (optional, abhängig vom Basistyp, Gerät bzw. Bussystem).

Factor Der vom Gerät empfangene Wert wird mit diesem Faktor multipliziert, bevor er weiter verarbeitet wird. Optional kann noch ein Offset angegeben werden:

valueGateway = Factor * valueDevice
valueDevice = 1/Factor * valueGateway + Offset

Der Faktor wird durch einen Bruch dargestellt z.B.: „1/1000+20“ oder „3600/1-10“. Der Nenner und der Zähler dürfen nicht Null sein.
Die Verwendung dieses Attributs ist vom Variablentyp abhängig.
Die gültigen Werte können Kapitel 3 entnommen werden.

Beispiel
multip="10/32-20" multip="40/100" multip="1/300+50"

Exp Exponent zur Basis 10 der die Auflösung einer Festkommazahl vom Typ **simpleType = Uint8, Uint16, Uint32, Int8, Int16, Int32 (siehe *BasicType*)** beschreibt.
Der im FP IoT Gateway gespeicherte Werte wird mit 10 exp(*Exp*) multipliziert (nachdem der *Factor* angewandt wurde), um den Wert des Parameters zu ermitteln.

valueParameter = 10^{Exp} * valueGateway.

Der Exponent bestimmt damit die Position des Kommas bei einer Festkommazahl an.

Es gibt folgende Werte für den Exponenten

Exp Wert	Beschreibung
-6	Auflösung = 0,000001
-5	Auflösung = 0,00001
-4	Auflösung = 0,0001
-3	Auflösung = 0,001
-2	Auflösung = 0,01
-1	Auflösung = 0,1
0	Auflösung = 1 (Standard)
1	Auflösung = 10
2	Auflösung = 100
3	Auflösung = 1000
4	Auflösung = 10000
5	Auflösung = 100000
6	Auflösung = 1000000

Precision Genauigkeit der Integerdarstellung eines Wertes mit **simpleType = Float , Double (siehe BasicType)**.

Der im FP IoT Gateway gespeicherte Wert wird mit 10 exp(*Exp*) multipliziert, um den Wert in die Integerdarstellung zu wandeln. Die Integerdarstellung wird bei der Berechnung der Prozessvariablen z.B. mit den Befehlen (GT, LT etc.) verwendet. Er gibt somit die Genauigkeit bei der Berechnung der Prozessvariablen an und ist Abhängig von der Anwendung zu definieren.

Integerdarstellung = Ganzzahl(10^{Exp} * valueParameter).

Die Werte des entsprechen der folgenden Tabelle.

Precision Wert	Beschreibung
-6	Faktor = 0,000001
-5	Faktor = 0,00001
-4	Faktor = 0,0001

Precision Wert	Beschreibung
-3	Faktor = 0,001
-2	Faktor = 0,01
-1	Faktor = 0,1
0	Faktor = 1 (Standard)
1	Faktor = 10
2	Faktor = 100
3	Faktor = 1000
4	Faktor = 10000
5	Faktor = 100000
6	Faktor = 1000000

Format Die Formatoptionen kennzeichnen die Standardformatierung des Wertes, wie sie bei Ausgaben z.B. in E-Mails oder beim Get Befehl (siehe Kapitel 5) erfolgt.

wFunct Funktionscode, der beim Schreiben einer Variable verwendet wird. Ob dieses Attribut verwendet wird und welche Werte gültig sind kann aus Kapitel 3 entnommen werden.

rFunct Funktionscode, der beim Lesen einer Variable verwendet wird. Ob dieses Attribut verwendet wird und welche Werte gültig sind kann aus Kapitel 3 entnommen werden.

2.2 Gerätevariablen lesen

Der Wert einer Variablen aus der External Datenbank kann, wie jeder andere Variablenwert, per TiXML Befehl „Get“ abgefragt werden. Dieser Befehl wird für die Variablen aus der External Datenbank um das Attribut „**format**“ erweitert.

Befehl:

```
[<Get _="Vpath" format ="Format" ViewProperties="ViewProperties"/>]
```

Übersicht der möglichen Parameter (kursiv geschriebene Werte):

Vpath Pfad zur Adressierung des Parameters oder des Gerätes oder des Busses.

Format Formatangabe, folgende Werte sind möglich:

Formatwert	Beschreibung
„integer“	1. simpleType = Uint8, Uint16, Uint32, Int8, Int16, Int32 (siehe 2.1 *BasicType*). Der Wert wird in der Integer-Darstellung ausgegeben. Die Integer-Darstellung berechnet sich mit Hilfe des für die Variable definierten Exponenten aus dem Wert der Variablen (siehe Kapitel 2.1 *Exp*) : Wert in Integer Darstellung = $10^{-Exp} * Wert$ 2. simpleType = float, double (siehe 2.1 *BasicType*). Der Wert wird in der Integer-Darstellung ausgegeben. Die Integer-Darstellung berechnet sich mit Hilfe der für die Variable definierten Präzision aus dem Wert der Variablen (siehe Kapitel 2.1. *Precision*) : Wert in Integer Darstellung = $10^{-Precisoin} * Wert$ 3. alle anderen Datentypen (siehe 2.1 *BasicType*).

Formatwert	Beschreibung
	Der Wert wird in der **nativen** Darstellung ausgegeben (native Darstellung siehe Kapitel 2.1 *BasicType*).
Leer(„“) oder “native”	Der Wert wird in der nativen Darstellung ausgegeben (native Darstellung siehe Kapitel 2.1 *BasicType*)
Formatstring	Der Wert wird entsprechend der angegeben Formatierung ausgegeben (siehe Kapitel 5, Formatieren von SPS Variablenwerten).
ViewProperties	Wird in der Bus-Definition das Attribute AddProperties="Name,TimeStamp" definiert, kann bei einem Get-Befehl mit Hife der ViewProperties der Klarname des Busses (wenn definiert) und ein Zeitstempel ausgegeben werden. Der Zeitstempel liefert die Zeit des letzten erfolgreichen Lesens der Variable.

Fehlt das Formatattribut, wird der Wert in dem Format ausgegeben, das als Standardformat in der Variablendefinition (siehe Kapitel 2.1) angegeben wurde. Ist kein Standardformat definiert worden wird der Wert in diesem Fall in der nativen Darstellung ausgegeben (native Darstellung siehe Kapitel 2.1 *BasicType*).

Antwort:
`[<Get _="Value" />]`

Value Wert in der eingestellten Formatierung bzw. Darstellung.

2.3 Gerätevariablen setzen

Der Wert einer Variablen aus der `External` Datenbank kann, wie jeder andere Variablenwert, per TiXML Befehl „Set“ gesetzt werden. Dieser Befehl wird für die Variablen aus der `External` Datenbank um das Attribut „**format**“ erweitert.

Befehl:
`[<Set _="Vpath" value="Value" format ="Format" />]`

Übersicht der möglichen Parameter (kursiv geschriebene Werte):

Vpath Pfad zur Adressierung des Parameters.
Value Wert des Parameters .
Format Formatangabe, folgende Werte sind möglich:

Formatwert	Beschreibung
„integer“	**1. simpleType = Uint8, Uint16, Uint32, Int8, Int16, Int32 (siehe 2.1 *BasicType*).** Der Wert wird in der Integer-Darstellung eingegeben. Die Integer-Darstellung berechnet sich mit Hilfe des für die Variable definierten Exponenten (siehe Kapitel 2.1 *Exp*) : **Wert in Integer Darstellung** = 10 $^{-Exp}$ * **Wert** **2. simpleType = float, double (siehe 2.1 *BasicType*).**

Formatwert	Beschreibung
	Der Wert wird in der Integer-Darstellung eingegeben. Die Integer-Darstellung berechnet sich mit Hilfe der für die Variable definierten Präzision : **Wert in Integer Darstellung** = 10 $^{-Precision}$ * **Wert** **3. alle anderen Datentypen (siehe 2.1 *BasicType*).** Der Wert wird in der nativen Datsellung eingegeben (native Darstellung siehe Kapitel 2.1 *BasicType*)
Leer(„“) oder “native”	Der Wert wird in der nativen Darstellung eingegeben (native Darstellung siehe Kapitel 2.1 *BasicType*)
Formatstring	Der Wert wird entsprechend der angegeben Formatierung eingegeben (**z.Z. nicht implementiert**).

Fehlt das Formatattribut, wird der Wert wie im Falle des Formatwertes „native“ eingegeben.

Antwort:
`[<Set/>]`

2.4 Arrays

Sind Variablen in der External als Array angelegt (Attribut “no“), z.B.

```
<Variable_0 _="B" ind="22" no="8" acc="RW"/>
```

so lassen sich die einzelnen Werte der Arrays durch einen Variablensuffix in eckigen Klammern adressieren:

2.4.1 Arrays lesen

Das adressierte Element im Array wird in eckige Klammern angehängt,

```
[<Get _="Vpath[element] "/>]
```

Vpath Pfad zur Adressierung des Arrays.
element Adressiertes Element des Arrays.

z.B. der dritte Wert im Array:

```
[<Get _=“/Process/COM?/D?/Variable_0[3] “/>]
```

Wird kein Suffix angegeben, werden bei der Ausgabe alle Elemente durch Komma getrennt aufgelistet:

```
[<Get _="Vpath" >]
```

Antwort:

```
[<Get _="Value1,Value2,Value3,…Value8" />]
```

2.4.2 Arrays schreiben

Das adressierte Element im Array wird in eckige Klammern angehängt,

```
[<Set _="Vpath[element]" value="Value" />]
```

Vpath Pfad zur Adressierung des Arrays.
element Adressiertes Element des Arrays.
Value Wert des Elements.

z.B. der dritte Wert im Array:
`[<Set _="/Process/Bus?/D?/Variable_0[3]" value="20" />]`

Wird kein Suffix angegeben, müssen im Value alle Elemente durch Komma getrennt aufgelistet werden. Lücken sind nicht zulässig:

```
[<Set _="Vpath"
value="Value1,Value2,Value3,Value4,Value5,Value6,Value7,Value8" />]
```

Antwort:
`[<Set/>]`

2.5 Fehlerzustände bei Gerätevariablen verarbeiten

Die in der `External` Datenbank definierten Variablen können, abhängig vom Gerätetyp bzw. Bussystem, auch unterschiedliche Fehlerzustände speichern. Insbesondere bei der Abfrage über ein Kommunikationsprotokoll können beispielsweise Kommunikationsfehler oder Protokollfehler auftreten, so dass der Variablenwert (Wert des letzten fehlerfreien Zugriffs bzw. initaler Wert) ungültig ist.

Der Fehlerzustand der Variablen wird durch die beiden Fehlercodes ErrorClass und ErrorNumber dargestellt. Beide können ausgelesen bzw. für die Generierung eines Alarms weiterverarbeitet werden.

2.5.1 Fehlerzustand auslesen

Ist der Wert einer Variablen ungültig, d.h. es ist ein Fehler erkannt und im Fehlerzustand gespeichert worden, so wirkt sich dies in folgender Weise auf den Lesebefehl „Get“ aus.

1. TiXML „Get“ Befehl zum Auslesen der **Variablengruppe** (z.B. `[<Get _="/Process/COM2/D2"/>]):`
   Parametereinträge mit ungültigen Werten werden in der Antwort **nicht aufgeführt**.

2. TiXML „Get“ Befehl zum Auslesen **einer Variablen**:
   (z.B. `[<Get _="/Process/COM2/D2/Sprache" />])`

   In der Antwort wird ein TiXML Fehler ausgegeben (siehe TiXML Error Frame in TiXML Reference Manual) :
   ErrorText „Variable exists but does not contain data“
   ErrNo -2194

   Hinweis:
   Beide Befehle liefern den zuletzt fehlerfrei gelesenen Wert anstelle einer Fehleranzeige, wenn mindestens ein Zugriff nach dem Upload der Variablenkonfiguration oder einem Systemstart (Reset, Power On) fehlerfrei war.

Für das direkte Auslesen des Fehlerzustandes eines Parameters oder von Parametergruppen wird der TiXML „Get“ Befehl (siehe TiXML Reference Manual) um ein XML Attribut erweitert, das den „Get“ Befehl veranlasst anstelle des Parameterwertes den Wert einer Zusatzinformation zu einem Parameter auszugeben in diesem Falle den Fehlerzustand:

Befehl:
`[<Get _="VPath" AddInfo="AddInfo" />]`

Übersicht der möglichen Parameter (kursiv geschriebene Werte):

Vpath Pfad zur Adressierung des Parameters.

AddInfo **Error .....** Gibt Fehlerzustand des Wertes zurück.

Das Attribut „AddInfo“ wird bei Werten, die keine Parameter von externen Geräten sind ignoriert. Und der Wert des Parameters zurückgegeben.

Antwort:
AddInfo = Error
[<Get _="***ErrorClass,ErrorValue***" />]

ErrorClass: Fehler Klasse
0 Kein Fehler
>0 Fehler

ErrorValue: Fehlerwert

ErrorClass	ErrorValue	Bedeutung
0	0	Kein Fehler
1	*n* > 0	Fehler in der Zugriffsimplementation des FP IoT Gateways. Die Nummer *n* ist > 0 und abhängig vom jeweiligen Gerät bzw. Bussystem.
c >1	*n*	Fehlermeldung des angeschlossenen Gerätes bzw. des Busprotokolls. Die Nummern *n* und *c* sind abhängig vom jeweiligen Gerät bzw. Bussystem.

Hinweis:
Der Fehlerzustand wird durch einen nachfolgenden fehlerfreien Zugriff auf die SPS bzw. auf den Bus wieder gelöscht (d.h. auf 0,0 gesetzt).

2.5.2 Fehlerzustand verarbeiten

Um den Fehlerzustand zur Erzeugung von Alarmen (oder allgemein Ereignissen) zu nutzen wird ein spezieller Ladebefehl in der Instruction List des „<Value/>“ Eintrags bei der Definition einer Prozessvariablen (siehe TiXML Reference Manual) definiert:

Befehl:
<LDS _="***Vpath***" AddInfo="***AddInfo***" />

Übersicht der möglichen Parameter (kursiv geschriebene Werte):

Beschreibung:
Liest den Fehlerzustand des Parameters aus, der in *Vpath* referenziert wird und schreibt in den Verarbeitungsstack folgende Werte:

High-Teil (Bit 16 - 31)	Low Teil (Bit 0 - 15)
Wert der Fehlerklasse	Wert des Fehlers

Vpath: Pfad zur Adressierung des Parameters.

AddInfo:
ErrorCode ..... Gibt Fehlerzustand des Wertes zurück.

3 Unterstützte SPS-Systeme

3.1 Mitsubishi Alpha XL

Die zu überwachenden Variablen der angeschlossenen Mitsubishi Alpha XL müssen im FP IoT Gateway definiert sein.
Die Mitsubishi Variablen sind in der External-Gruppe der 'PROCCFG' Datenbank gespeichert:

Verwende COM-Port COM2

```
<External>
   <Bus _="COM2" protocol="Mitsubishi,Alpha2" type="Master"
    baud="9600">
      <Device _=”0” Pollrate=”1s”>
         <Input1 _="I" ind="1"/>
         <ExtInput129 _="EI" ind="129" acc="R"/>
         <M1 _="M" ind="1" acc="R"/>
         <Keypad1 _="K" ind="1" acc="R" />
         <Display _="N" ind="3" acc="R" />
         <InAnalog7 _="AI" ind="7" acc="R" />
         <Counter1  _="CB" ind="1" acc="R" />
         <CW2 _="CW" ind="2" />
         <Out2 _="O" ind="2" acc="W"/>
         <ExtOut129 _="EO" ind="129" acc="W"/>
      </Device>
   </Bus>
</External>
```

Master-Kommunikation

Variablenliste

Der BUS-Parameter enthält die Adresse der Erweiterungskarte, den Protokoll-Hersteller “Mitsubishi”, den Typ der angeschlossenen Steuerung “Alpha2”, den Kommunikationsmodus “Master” und die verwendete Baudrate.

Die Device-ID (Stationskennung) muss mit der SPS-Adresse übereinstimmen (Standard 0).

Wenn das FP IoT Gateway an der “MB”-Schnittstelle (Mainboard) angeschlossen ist, startet die SPS-Kommunikation direkt nach Abziehen des PC-Kabels, und unterbricht automatisch bei Erkennung eines TiXML-Befehls.

Es kann eine Liste von Variablen definiert werden:

Variablentyp in der Alpha-XL (I,O,AI...)

```
<Alarm11 _="I" ind="5" acc="R"/>
```

Jede Zeile definiert einen logischen Namen (Alias, z.B. Alarm11) und den Typ der Variable in der Mitsubishi SPS (siehe: Liste der unterstützten Variablen)
Der ‘ind’ Parameter bestimmt die Adresse der Variable in der Mitsubishi Steuerung und der ‘acc’ Parameter das Zugriffsrecht. Das Zugriffsrecht kann entweder ‘R’ oder ‘RW’ für Lese oder Lese-/Schreibzugriff sein, abhängig von der gewählten Variable.

Der ‘def’ Parameter bestimmt den Startwert der Variable. Eine Variable mit Schreibzugriff enthält diesen Startwert bis zum ersten Schreibzugriff. Eine Variable mit Lesezugriff enthält diesen Wert bis das Gerät den tatsächlichen Wert von der SPS erhalten hat.

Liste der unterstützten Variablen für Alpha XL:

Typ	index	access	Kommentar
M	1-14	R	Systembits
I	1-15	R	Digitale Eingänge
EI	129-132	R	Eingänge Erweiterungsmodul
O	1-9	RW	Ausgänge
EO	129-132	RW	Ausgänge Erweiterungsmodul
K	1-8	R	BTasten
E	1-4	R	Link Eingänge
A	1-4	RW	Link Ausgänge
N	1-4	RW	Kontrollbits
AI	1-8	R	Analogeingang
CB	1-100	RW	Bitoperanden
CW	1-100	RW	Wordoperanden

Wenn nur Lesezugriff ‚R' möglich ist, kann der Parameter ‚acc' weggelassen werden.
Wenn die SPS im "RUN" ist, können einige Variablen durch das SPS-Programm überschrieben werden. In diesem Fall müssten Merker vorgeschaltet werden.

Hinweis: Wenn der automatisch generierte DeviceState (siehe Kapitel 6.3) "1" ist, aber keine Variablenwerte empfangen werden, so existieren im Gerät oder der Alpha XL Bit- oder Wordoperanden, welche auf einer der beiden Seiten nicht definiert sind. Diese Variablen müssen immer auf beiden Geräten vorhanden sein, ansonsten antwortet die Alpha XL nicht.

Eine Alpha XL anschließen

Das FP IoT Gateway muss an die Alpha XL über ein Mitsubishi GSM-CAB verbunden werden.

Beachten Sie folgende Hinweise:

1. In der Alpha XL muss ein Program mit aktivierter "seriellller Kommunikation" auf 9600/8N1 vorhanden sein. (siehe Alpha Programming Software online Hilfe). Nach dem Aktivieren muss die Alpha neu gestartet werden.
2. Das GSM-CAB kann direkt an die Mainboard RS232 Schnittstelle (MB) angeschlossen werden.
3. Wenn Sie das GSM-CAB an eine FP IoT Gateway Erweiterungskarte RS232-2 (C1) anschließen, müssen Sie ein Nullmodem-Kabel zwischen Gerät und GSM-CAB verwenden.

Fernzugriff

Für den Fernzugriff auf die SPS ist folgender "TransMode"-Befehl notwendig: (siehe TiXML-Reference manual für weitere Informationen)

Alpha XL:
`[<TransMode baud="9600" format="8N1" handshake="noDTR" com="COM1"/>]`
(verwenden Sie `com="COM2",` wenn die Alpha XL an der COM2 RS232 angeschlossen ist)

3.2 Mitsubishi MELSEC FX

Die zu überwachenden Variablen der angeschlossenen Mitsubishi FX Steuerung müssen im FP IoT Gateway definiert sein.
Die Mitsubishi Variablen sind in der External-Gruppe der 'PROCCFG' Datenbank gespeichert:

```
<External>
   <Bus _="COM2" protocol="Mitsubishi,Format1" type="Master"
    baud="9600">
      <Device _="0" Pollrate="1s" devType="FX1N">
         <Input1 _="X" ind="1"/>
         <M1 _="M" ind="1" acc="R"/>
         <Timer _="TS" ind="1" acc="R" />
         <Counter _="CS" ind="3" acc="R" />
         <CN2 _="CN" ind="2" />
         <Out2 _="Y" ind="2" acc="W"/>
      </Device>
   </Bus>
</External>
```

Verwende COM-Port COM2

Master-Kommunikation

Variablenliste

Der BUS-Parameter enthält die Adresse der Erweiterungskarte, den Protokoll-Hersteller “Mitsubishi”, den Protokoll-Typ “Format1”, den Kommunikationsmodus “Master” und die verwendete Baudrate.

Die Device-ID (Stationskennung) muss mit der SPS-Adresse übereinstimmen (Standard 0).
Das Format1-Protokoll ist Netzwerkfähig (über RS485), wodurch mehrere Devices mit unterschiedlichen IDs existieren können.
Der Parameter „devType“ bestimmt den CPU-Typ: FX1S,FX1N oder FX2N.

Wenn das FP IoT Gateway an der “MB”-Schnittstelle (Mainboard) angeschlossen ist, startet die SPS-Kommunikation direkt nach Abziehen des PC-Kabels, und unterbricht automatisch bei Erkennung eines TiXML-Befehls.

Es kann eine Liste von Variablen definiert werden:

```
<Alarm11 _="Y" ind="5" acc="R" />
```

Variablentyp in der FX (X,Y,..)

Jede Zeile definiert einen logischen Namen (Alias, z.B. Alarm11) und den Typ der Variable in der Mitsubishi SPS (siehe: Liste der unterstützten Variablen)
Der ‘ind’ Parameter bestimmt die Adresse der Variable in der Mitsubishi Steuerung und der ‘acc’ Parameter das Zugriffsrecht. Das Zugriffsrecht kann entweder ‘R’ oder ‘RW’ für Lese oder Lese-/Schreibzugriff sein, abhängig von der gewählten Variable.

Der ‘def’ Parameter bestimmt den Startwert der Variable. Eine Variable mit Schreibzugriff enthält diesen Startwert bis zum ersten Schreibzugriff. Eine Variable mit Lesezugriff enthält diesen Wert bis das Gerät den tatsächlichen Wert von der SPS erhalten hat.

Liste der unterstützten Variablen für MELSEC FX „Format1“:

Typ	index	access	Kommentar
M	0-8255	RW	Merker
Y	0-377	RW	Ausgänge (oktal)
X	0-377	R	Eingänge (oktal)
S	0-999	RW	Schrittstatus
CS	0-255	RW	Zähler Kontakt
TS	0-255	RW	Timer Kontakt

Typ	index	access	Kommentar
TN	0-255	RW	Timer Istwert vorzeichenlos
TNI	0-255	RW	Timer Istwert vorzeichenbehaftet
CN	0-199	RW	Zähler Istwert (16Bit)
CD	200-255	RW	Zähler Istwert (32Bit) vorzeichenlos
CDI	200-255	RW	Zähler Istwert (32Bit) vorzeichenbehaftet
D	0-8255	RW	Register (16 Bit) vorzeichenlos
DI	0-8255	RW	Register (16 Bit) vorzeichenbehaftet
DW	0-8254	RW	Register (32 Bit) vorzeichenlos
DWI	0-8254	RW	Register (32 Bit) vorzeichenbehaftet

Wenn nur Lesezugriff ‚R' möglich ist, kann der Parameter ‚acc' weggelassen werden.
Wenn die SPS im "RUN" ist, können einige Variablen durch das SPS-Programm überschrieben werden. In diesem Fall müssten Merker vorgeschaltet werden.

Zusätzlich zu dem offenen Protokoll „Format1" ist das FX-interne Protokoll in den Geräten integriert. Die Auswahl erfolgt über den Bus-Parameter
`protocol="Mitsubishi,FX".` Dieses Protokoll ist nicht netzwerkfähig, bietet jedoch zusätzlich Zugriff auf die Contacts und Resets der Timer/Counter:

Liste der zusätzlich unterstützten Variablen für MELSEC FX „FX Protocol":

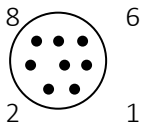
Typ	index	access	Kommentar
CC	0-255	R	Zähler Spule
TC	0-255	R	Timer Spule
CR	0-255	R	Zähler Reset
TR	0-255	R	Timer Reset

Eine Mitsubishi FX anschließen

Das FP IoT Gateway kann an die FX interne RS422 Schnittstelle oder über eine zusätzliche Schnittstellenerweiterung RS232-BD / RS422-BD / RS485-BD angeschlossen werden.
Wenn Sie eine BD-Erweiterung verwenden, muss diese Schnittstelle über die GX Developer Software mit den Parametern 9600/7E1 aktiviert werden.
Beide Schnittstellen können simultan verwendet werden, um an die FX z.B. ein Gerät und ein Display gleichzeitig anzuschließen.

RS422 Anschluss:

FP IoT Gateway (RS422 Erweiterungskarte)		MELSEC FX Port (8pin weiblich)
T+	----------	2
T -	----------	1
R-	----------	4
R+	----------	7



RS485 Anschluss:

FP IoT Gateway (RS485 Erweiterungskarte)		MELSEC FX RS485-BD (5pol Schraubklemme)
T+	----------	RDA
T -	----------	RDB
R-	----------	SDB
R+	----------	SDA
0V	----------	SG

Fernzugriff
Für den Fernzugriff auf die SPS ist folgender “TransMode”-Befehl notwendig: (siehe TiXML-Reference manual für weitere Informationen)

MELSEC FX:

FP IoT Gateway RS232 Schnittstelle:
```
[<TransMode baud="9600" format="7E1" com="COM1"/>]
```
(Verwenden Sie `com=”COM2”` wenn die SPS an der COM2 RS232 steckt.)

FP IoT Gateway RS422 Schnittstelle:
```
[<TransMode baud="9600" format="7E1" handshake=”FULL” com="COM1"/>]
```

3.3 Siemens Simatic S7-200 über MPI-Schnittstelle

Die zu überwachenden Variablen der angeschlossenen Siemens Simatic S7-200 müssen im FP IoT Gateway definiert sein.
Die Siemens Variablen sind in der External-Gruppe der 'PROCCFG' Datenbank gespeichert:

RS485 Schnittstelle auf COM2

```
<External>
   <Bus _=”COM2” protocol=”Siemens,S7-200” type=”Master” baud=”9600”
    handshake=”HALF” TS=”0” MAXADR=”15” GUF=”1” RC=”1”>
      <Device _=”2” Pollrate=”1s”>
         <M30 _="M" ind="30" acc="R" />
      </Device>
      <Device _=”3” Pollrate=”60s”>
         <V100 _="V" ind="100" acc="R" />
         <VS50 _="VS" size=”4” ind="100" acc="R" />
      </Device>
   </Bus>
</External>
```

Station #2

Station #3

Der BUS-Parameter enthält die Adresse des COM-Ports, den Protokoll-Hersteller “Siemens”, den Typ der angeschlossenen Steuerung “S7-200”, den Kommunikationsmodus “Master”, die verwendete Baudrate und das notwendige Handshake (bei Verwendung der RS485 Erweiterungskarte). TS ist die Stationsnummer des Geräts, MAXADR der Bereich der abzufragenden Stationsnummern, GUF der “gap update factor” um andere Slaves zu erkennen und RC die Anzahl der Wiederholungen bei Kommunikationsfehlern.

Das FP IoT Gateway kann gegenüber der S7-200 nur Master sein. Die Standardbaudrate ist 9600.

Wenn das FP IoT Gateway an der “MB”-Schnittstelle (Mainboard) angeschlossen ist, startet die SPS-Kommunikation direkt nach Abziehen des PC-Kabels, und unterbricht automatisch bei Erkennung eines TiXML-Befehls.

Für jede Steuerung muss ein ‘Device’-Abschnitt eingefügt warden, welcher die Stationsnummer (‘_’ – Attribut) und den Abfragezyklus enthält. Nach jedem Abfragezyklus werden die Variablenwerte erneut eingelesen.

Es kann eine Liste von Variablen definiert werden:

Variablentyp in der S7-200

```
<AlarmM10 _="M" ind=”10” acc=”R”/>
```

Jede Zeile definiert einen logischen Namen (Alias, z.B. Alarm11) und den Typ der Variable in der Siemens SPS (siehe: Liste der unterstützten Variablen).

Der ‘ind’ Parameter bestimmt die Adresse der Variable in der Simatic Steuerung und der ‘acc’ Parameter das Zugriffsrecht. Das Zugriffsrecht kann entweder ‘R’ oder ‘RW’ für Lese oder Lese-/Schreibzugriff sein, abhängig von der gewählten Variable.

Es muss das zusätzliche Zugriffsrecht ‘A’ angefügt werden, welches den aktiven Zugriff auf die Variablen erlaubt.

Der ‘def’ Parameter bestimmt den Startwert der Variable. Eine Variable mit Schreibzugriff enthält diesen Startwert bis zum ersten Schreibzugriff. Eine Variable mit Lesezugriff enthält diesen Wert bis das Gerät den tatsächlichen Wert von der SPS erhalten hat.

Liste der unterstützten Variablen für S7-200 CPUs

Typ	index	access	Kommentar
V	0-10239.7	R/W	Variablenspeicher (Bit)
VB	0-10239	R/W	Variablenspeicher (Byte)
VW	0-10238	R/W	Variablenspeicher (Word)
VD	0-10236	R/W	Variablenspeicher (Doppel-Word)
VS	0-10239	R/W	Variablenspeicher (string, erfordert Parameter “size”)
I	0-15.7	R	Eingänge
Q	0-15.7	R/W	Ausgänge
M	0-31.7	R/W	Merker (Bit)
MB	0-31	R/W	Merker (Byte)
MW	0-30	R/W	Merker (Word)
MD	0-28	R/W	Merker (Doppel-Word)
SM	0-549.7	R	Sondermerker
S	0-31.7	R/W	Ablaufsteuerungsrelais
T	0-255	R/W	Timer
C	0-255	R/W	Zähler
AI	0-62	R	Analogeingang
AQ	0-62	R/W	Analogausgang
HC	0-5	R	High speed Zähler

Eine S7-200 anschließen

Die S7-200 kann über ein PPI-Kabel (RS232) oder über einen Profibus-Adapter (RS485 Erweiterungskarte) angeschlossen werden.

FP IoT Gateway (RS485 Erweiterungskarte) — S7-200 (Profibus-Adapter)

T+ ----------B1

T- ----------A1

Fernzugriff

Für den Fernzugriff auf die SPS ist folgender “TransMode”-Befehl notwendig: (siehe TiXML-Reference Manual für weitere Informationen).

FP IoT Gateway RS232 Schnittstelle:

```
[<TransMode baud="9600" format="8E1" hanshake="noDTR" com="COM1"/>]
```

(Verwenden Sie `com=”COM1”` wenn die SPS an COM1 (RS232) steckt.)

FP IoT Gateway RS485 Schnittstelle:

```
[<TransMode baud="9600" format="8E1" handshake="HALF" com="COM1"/>]
```

Um sich mit einer S7-200 zu verbinden, muss lokal ein 11-Bit modem (8E1 Datenformat) verwendet werden. In der MicroWin Software müssen zudem über einen auf der Tixi CD mitgelieferten Registry-Patch die Timings angepasst werden.

3.4 Siemens Simatic S7-300/400 über MPI-Schnittstelle

Die zu überwachenden Variablen der angeschlossenen Siemens Simatic S7-300/400 müssen im FP IoT Gateway definiert sein.
Die Siemens Variablen sind in der External-Gruppe der 'PROCCFG' Datenbank gespeichert:

```
<External>
    <Bus Name="Bus1" _="COM2" family="Siemens" protocol="Siemens,S7-300/400-A"
     type="Master" TS="0" MAXADR="15" GUF="1" RC="1">

        <Device _="2" Name="Device_2" Pollrate="2s">
          <AlarmS7300 _="M"  ind="0.0" acc="RA"  def="0" format="?Alarm,OK"/>
          <Confirmation _="M"  ind="0.1" acc="RWA" def="1"/>
          <I124 _="VM" ind="0" db="15" acc="RA" def="0"/>
          <Test _="M" ind="0.3" acc="RA" def="0" format="?Alarm,OK"/>
        </Device>
        <Device _="3" Name="Device_3" Pollrate="60s">
          <AlarmS7300 _="M" ind="0.0" acc="RA" def="0" format="?Alarm,OK"/>
          <Confirmation _="M" ind="0.1" acc="RWA" def="1"/>
        </Device>
    </Bus>
</External>
```

Der BUS-Parameter enthält die Adresse des COM-Ports, den Protokoll-Hersteller “Siemens”, den Typ der angeschlossenen Steuerung “S7-300/400-A” und den Kommunikationsmodus “Master” oder „Slave“. TS ist die Stationsnummer des Geräts, MAXADR der Bereich der abzufragenden Stationsnummern, GUF der “gap update factor” um andere Slaves zu erkennen und RC die Anzahl der Wiederholungen bei Kommunikationsfehlern.

Beim Laden einer External-Definition werden die von der Siemens Teleservice Software im TS-Adapter hinterlegten Parameter deaktiviert.

Das FP IoT Gateway kann gegenüber der S7-300/400 Master oder Slave sein.

Für jede Steuerung muss ein ‘Device’-Abschnitt eingefügt warden, welcher die Stationsnummer (‘_’ – Attribut) und den Abfragezyklus enthält. Nach jedem Abfragezyklus werden die Variablenwerte erneut eingelesen (Master) oder ein Kommunikationstimeout erkannt (Slave).

Es kann eine Liste von Variablen definiert werden:

```
<AlarmM10 _="DBX" db=”2” ind=”0.0” acc=”R”/>
```

Variablentyp in der S7-300

Datenbaustein der Variablen

Jede Zeile definiert einen logischen Namen (Alias, z.B. AlarmM10) und den Typ der Variable in der Siemens SPS (siehe: Liste der unterstützten Variablen).
Der ‘ind’ Parameter bestimmt die Adresse der Variable in der Simatic Steuerung und der ‘acc’ Parameter das Zugriffsrecht. Das Zugriffsrecht kann entweder ‘R’ oder ‘RW’ für Lese oder Lese-/Schreibzugriff sein, abhängig von der gewählten Variable.

Der ‚db‘ Paramter bestimmt die Nummer des Datenblocks in dem sich die Variable befindet. Der Wert kann zwischen 1 und 65535 liegen.

Der ‘def’ Parameter bestimmt den Startwert der Variable. Eine Variable mit Schreibzugriff enthält diesen Startwert bis zum ersten Schreibzugriff. Eine Variable mit Lesezugriff enthält diesen Wert bis das Gerät den tatsächlichen Wert von der SPS erhalten hat.

Liste der unterstützten Variablen für S7-300/400 CPUs

Typ	index	access	Kommentar
DBX	0.0-65535.7	R/W	Datenbaustein (Bit, erfordert Parameter “db”)
DBB	0-65535	R/W	Datenbaustein (Byte, erfordert Parameter “db”)
DBW	0-65534	R/W	Datenbaustein (Word, erfordert Parameter “db”)
DBD	0-65532	R/W	Datenbaustein (DoubleWord, erfordert Parameter “db”)
DBS	0-65535	R/W	Datenbaustein (string, erfordert Parameter “size” und “db”)
I	0-16384.7	R	Eingänge
Q	0-16384.7	R/W	Ausgänge
M	0-65535.7	R/W	Merker Bit
MB	0-65535	RW	Merker Byte
MW	0-65534	RW	Merker Wort
MD	0-65532	RW	Merker DWort
T	0-2074	R/W	Timer
C	0-2074	R/W	Zähler

Eine S7-300/400 anschließen

Die S7-300/400 wird über einen Profibus-Adapter an das FP IoT Gateway über MPI angeschlossen.

FP IoT Gateway S7-300/400
(Profibus-Adapter) (Profibus-Adapter)
B1 ----------B1
A1 ----------A1

Fernzugriff

Der Fernzugriff erfolgt völlig transparent über die TS-Adapter Funktion.

Folgende Bedingungen gelten für den TS-Adapter-Zugriffschutz:

1. Der lokale Zugriff auf die S7 durch das Gerät (COM1->COM2) lässt sich nicht sperren. (Verhält sich also wie der Original Siemens-TS-Adapter).

2. Die über die SIMATIC-Software eingestellten TS-Benutzer (Menü "TS-Adapter-Parametrieren") funktionieren wie beim Original Siemens-TS-Adapter, d.h. es lässt sich ein ADMIN und 2 User anlegen, sowie eine CallBack-Funktion für die Benutzer.

3. Die TS-Adapter-Einstellungen werden beim Factory-Reset gelöscht.

4. Sobald im Gerät eine S7 Anbindung (External) UND AccRights User (siehe TiXML-Reference-Manual) definiert wurden, gelten nurnoch die in den AccRights angelegten User mit "TSAdapter" Dienst. Bei diesen lässt sich auch ein Callback über das gleichnamige Userattribut anlegen. Die im TS-Adapter parametrierten User bleiben erhalten, sind aber inaktiv. Erst nach Löschen der AccRights werden diese wieder wirksam.

Weiterer Vorteil: In den AccRights lassen sich quasi beliebig viele User anlegen, eine Beschränkung auf 3 User (wie im TS-Adapter) gibt es hier nicht.

Beispiel TS-Adapter AccRights:

```
[<SetConfig _="USER" ver="y">
   <AccRights>
      <Groups>
         <Hausmeister>
            <TSAdapter AccLevel="1"/>
         </Hausmeister>
      </Groups>
      <User>
         <Krause Plain="Dackel" Group="Hausmeister" Callback="123"/>
      </User>
   </AccRights>
</SetConfig>]
```

3.5 Siemens Simatic S7-200/300/400/1200/1500 über LAN-Schnittstelle

Die Kommunikation der S7-Steuerungen über LAN (TCP/IP) erfolgt mit Hilfe des Protokolls "ISO-ON-TCP" nach RFC1006 (manchmal auch als "AGLink" Protokoll bezeichnet). Unterstützt werden derzeit Siemens S7-200, S7-300/400, S7-1200, S7-1500.
Die Konfiguration ist für die einzelnen SPS Typen prinzipiell gleich.

3.5.1 Allgemeines und Hinweise

Zur Zeit (Firmware Version 3.04.01.016) muss nach jedem Laden einer neuen Konfiguration (`PROCCFG`) ein Reset durchgeführt werden. Bei den Bus- und Deviceparametern wird Groß- und Kleinschreibung unterschieden, "`PLCC`" ist also nicht gleich "`plcc`" .

3.5.2 Busparameter

```
<Bus _="ETH" BusId="1" name="S7_300" protocol="Siemens,AGLink" type="Master">
```

_=“ETH”	Bezeichnet die Ethernetschnittstelle des FP IoT Gateways (auf dem Mainboard)
“BusId”	Ermöglicht die Adressierung des SPS-Bus unabhängig von der Schnittstelle. Darf nicht mit "Name" vermischt werden.
“Name”	Ermöglicht die Benennung des SPS-Bus und damit eine von der Schnittstelle unabhängige Adressierung. Darf nicht mit "BusId" vermischt werden.
“protocol”	Name des Protokolls, muss auf ”Siemens,AGLink” eingestellt sein
“type”	Art des Protokolls, muss immer auf “Master” eingestellt sein

3.5.3 Device-Parameter

```
<Device _="2" Name="Dev2" Pollrate="1s" IP="193.101.167.203"
 GW="193.101.167.193" mask="255.255.255.0" rack="0" slot="2" PLCC="0">
```

"2"	FP IoT Gerätenummer; entspricht der PLC-Nummer der angeschlossenen CPU
“IP”	IP-Adresse der Siemens-SPS
“GW”	Zu verwendendes Netzwerk-Gateway
“mask”	Netzmaske der IP Adresse (default: 255.255.255.0)
“rack”	Racknummer der Siemens CPU (default: 0)
“slot”	Slotnummer der Siemens CPU (default: 300:2, 400:3, 1200:1)
“PLCC”	PLC Class. Die Art der angeschlossenen Steuerung. 0: 300/400, 1: 200, 2: 1200 (default: 1)

3.5.4 Weitere Parameter nur für den internen Gebrauch:

“credit“ Bei einzelnen Fremdfabrikaten (SPS) notwendig, nicht bei Siemens

“ctype“ Typ der angeschlossenen Kommunikationseinheit. 0: PG, 1: OP, 2: Sonstige. (PG = Programmiergerät; OP = Operator Panel)

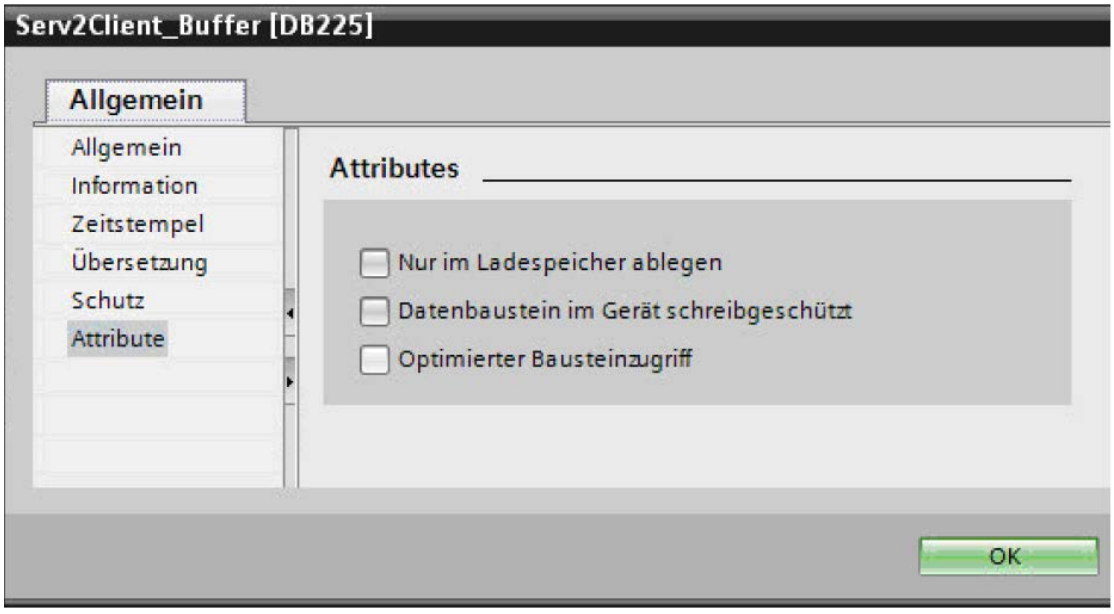
“port“ Die Portnummer für den TCP/IP-Verkehr ist standardmäßig 102. Soll ausnahmsweise eine andere Portnummer verwendet werden, so muss sie hier angegeben werden. Bei Siemens-Steuerungen wird immer Port 102 verwendet.

“Timeout“ Standardtimeout der Kommunikation. Ohne Maßeinheit und mit der Maßeinheit „ms“ (Millisekunden) wird der Wert so (als ms) übernommen. Bei anderen gültigen Maßeinheiten wird entsprechend auf ms umgerechnet. Ohne Angabe des Parameters, oder bei Überschreiten des Wertebereichs eines Doppelwortes (32 bit) wird der Defaultwert 6000(ms) eingestellt. Bei den meisten Anwendungen kann auf die Angabe dieses Parameters verzichtet werden.

Besonderheiten für S7-1200 und S7-1500

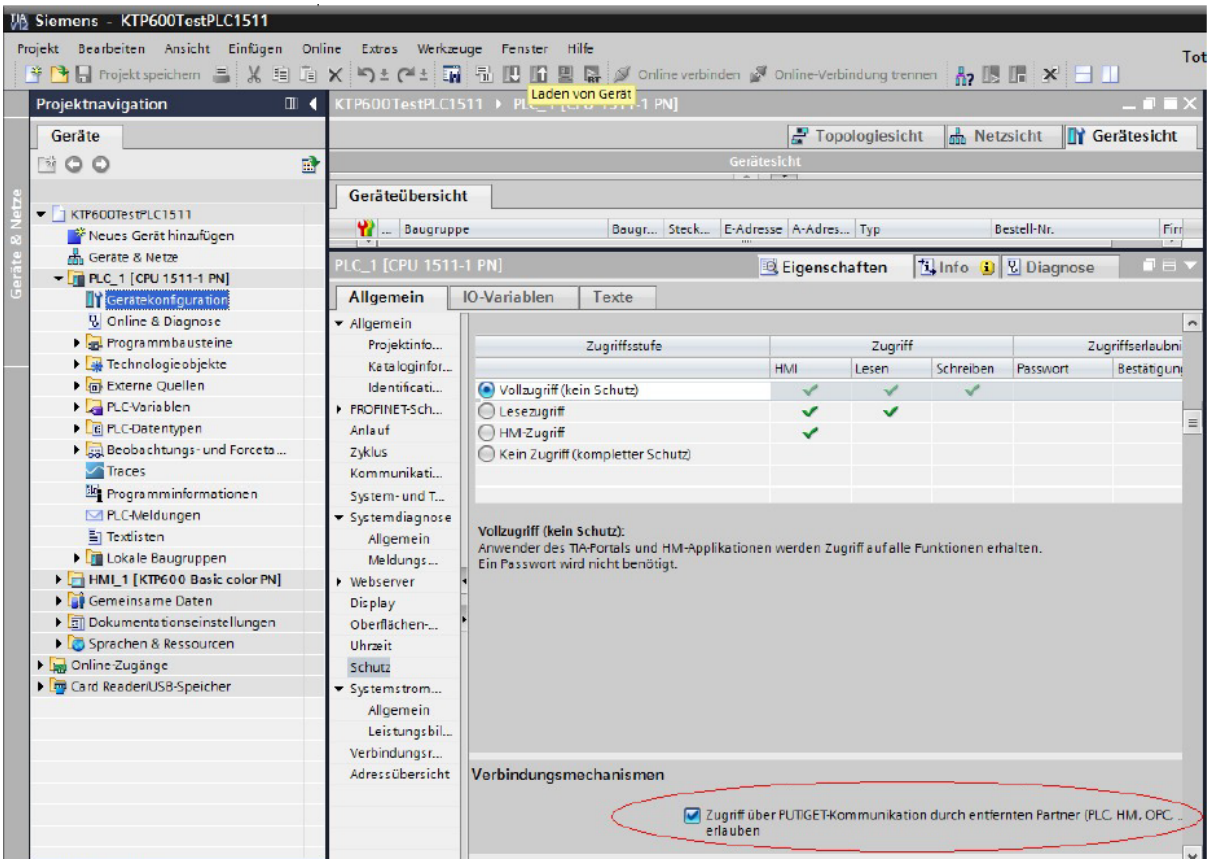
Der Zugriff auf S7-1200/1500 ist grundsätzlich möglich und ähnelt dem Zugriff auf eine S7-300 bzw. S7-400. Jedoch sind die folgenden Punkte dabei zu beachten:

- Der Zugriff auf Datenbausteine ist nur mit absoluter Adressierung möglich. Deshalb muss im Projekt der SPS das Attribut "Optimierter Bausteinzugriff" in den Eigenschaften der Datenbausteine deaktiviert werden.



Der Zugriff auf Peripherieeingangs- und Peripherieausgangsdaten einer S7-1200/1500 ist nicht möglich.

- Zugriff auf Timer (Zeiten), Counter (Zähler) und 64-Bit Datentypen einer S7-1200/1500 ist nicht möglich.
- Bei der S7-1500 und der S7-1200 (ab Firmware Version 4) muss die PUT/GETKommunikation zwingend erlaubt werden. Dies muss in der Gerätekonfiguration im SPS- Projekt aktiviert werden.



- Auf die neuen Datentypen, welche mit der S7-1200/1500 erschienenen sind, ist kein direkter Zugriff möglich.
  Auf die 64-Bit-Datentypen kann zurzeit nicht zugriffen werden. Eventuell lässt sich der neue Datentyp in einen bekannten 32-Bit Datentyp (REAL oder DWORD) umrechnen und in einem anderen Datenbereich (z. B. als Merker, DB, …) ausgeben.
  Für den Zugriff auf die restlichen der neuen Datentypen, kann ein vergleichbarer Datentyp einer S7-300/400 verwendet werden.

Es gibt folgende neuen Datentypen bei der S7-1500 mit teilweise vergleichbaren Datentypen der S7-300/400:

Datentyp S7-1500	Vergleichbarer Datentyp S7-300/400
USINT (8 Bit)	BYTE
SINT (8 Bit)	BYTE
UINT (16 Bit)	WORD
UDINT (32 Bit)	DWORD
ULINT (64 Bit)	zur Zeit nicht möglich
LINT (64 Bit)	zur Zeit nicht möglich
REAL (32 Bit)	REAL
LREAL (64 Bit)	zur Zeit nicht möglich

3.5.5 Variablen-Parameter

3.5.5.1 S7-300 / 400, S7-1200, S7-1500

Unterstützte Variablen für S7-300 / 400 CPUs über LAN
(abweichende Werte für S7-1200 / S7-1500 in Klammern)

Typ	index	access	Kommentar
DBX	0.0-65535.7	R/W	Datenbaustein (Bit, erfordert Parameter “db”)
DBB	0-65535	R/W	Datenbaustein (Byte, erfordert Parameter “db”)
DBW	0-65534	R/W	Datenbaustein (Word, erfordert Parameter “db”)
DBD	0-65532	R/W	Datenbaustein (DoubleWord, erfordert Parameter “db”)
DBS	0-65535	R/W	Datenbaustein (string, erfordert Parameter “size” und “db”) (S7-1200: Bitte DBB verwenden !)
I	0.0-16384.7	R	Eingänge (S7-1200: index 0.0 .. 1023.7)
Q	0.0-16384.7	R/W	Ausgänge (S7-1200: index 0.0 .. 1023.7)
M	0-65535.7	R/W	Merker Bit
MB	0-65535	R/W	Merker Byte
MW	0-65534	R/W	Merker Wort
MD	0-65532	R/W	Merker DWort
T	0-999	R/W	Timer
C	0-999	R/W	Zähler

Das Zugriffsrecht kann entweder ‘R’ oder ‘RW’ für Lese oder Lese-/Schreibzugriff sein, abhängig von der gewählten Variable. Es muss das zusätzliche Zugriffsrecht ‘A’ angefügt werden, welches den aktiven Zugriff auf die Variablen erlaubt (also `acc="RA"` für Lesen, `acc="RWA"` für Lesen + Schreiben).

Der ‚db‘ Parameter bestimmt die Nummer des Datenblocks in dem sich die Variable befindet. Der Wert kann zwischen 1 und 65535 liegen.


```
<AlarmM10 _="DBX" db="2" ind="0.0" acc="RA" />
```

"DBX"	Variablentyp "Datenbaustein"
"2"	Datenbaustein der Variablen
"0.0"	Index der Variable innerhalb des Datenbausteins
"RA"	Zugriffsrecht "Lesen"

3.5.5.2 S7-200

Unterstützte Variablen für S7-200 CPUs über LAN:

Typ	index	access	Kommentar
DBX	0.0-10239.7	R/W	Entspricht Datentyp V (Variablenspeicher Bit). Zugriff über db=1: db="1" DBX="0.0 – 10239.7"
DBB	0-10239	R/W	Entspricht Datentyp VB (Variablenspeicher Byte). Zugriff über db=1 DBB="..": db="1" DBB="0 – 10239"
DBW	0-10238	R/W	Entspricht Datentyp VW (Variablenspeicher Word). Zugriff über db=1 DBW="..": db="1" DBW="0 – 10238"
DBD	0-10236	R/W	Entspricht Datentyp VD (Variablenspeicher DWord). Zugriff über db=1 DBD="..": db="1" DBD="0 – 10236"
I	0.0-15.7	R	Eingänge
Q	0.0-15.7	R/W	Ausgänge

Typ	index	access	Kommentar
M	0-31.7	R/W	Merker Bit
MB	0-31	R/W	Merker Byte
MW	0-30	R/W	Merker Wort
MD	0-28	R/W	Merker Dwort
M	256.0 – 805.7	R	Entspricht Datentyp SM (Sondermerker Bit). Zugriff über M mit Offset 256 M="256.0" entspricht SM="0.0", M="805.7" entspricht SM="549.7"
T	0-255	R/W	Timer
C	0-255	R/W	Zähler

3.5.6 Beispiele

External-Konfiguration für eine Siemens S7-300

```
[<SetConfig _="PROCCFG" ver="v">
<External>
   <Bus _="ETH" BusId="1" Name="S7_300" protocol="Siemens,AGLink" type="Master">
      <Device _="2" Name="Dev2" Pollrate="1s"
        IP="193.101.167.203" GW="193.101.167.193"
        mask="255.255.255.0" rack="0" slot="2" PLCC="0">
         <MB0 _="MB" acc="RWA" def="" ind="0"/>
         <MB1 _="MB" acc="RWA" def="" ind="1"/>
         <MW2 _="MW" acc="RWA" def="" ind="2"/>
         <M0.0 _="M" acc="RWA" def="" ind="0.0"/>
         <M0.1 _="M" acc="RWA" def="" ind="0.1"/>
         <T0 _="T" acc="RWA" def="" ind="0"/>
         <C0 _="C" acc="RWA" def="" ind="0"/>
         <I0.0 _="I" acc="RA" def="" ind="0.0"/>
         <I0.1 _="I" acc="RA" def="" ind="0.1"/>
      </Device>
   </Bus>
</External>
</SetConfig>]
```

External-Konfiguration für eine Siemens S7-1200

```
[<SetConfig _="PROCCFG" ver="v">
<External>
   <Bus _="ETH" BusId="1" Name="S7_1200" protocol="Siemens,AGLink" type="Master">

      <Device _="1" Pollrate="1s" IP="193.101.167.135" GW="193.101.167.193"
              mask="255.255.255.128" rack="0" slot="1" PLCC="2">

         <!--Eingaenge-->
         <I0.0 _="I" acc="RA" def="" ind="0.0"/>

         <!--Ausgaenge-->
         <Q0.0 _="Q" acc="RA" def="" ind="0.0"/>

      </Device>
   </Bus>
</External>
</SetConfig>]
```

3.6 VIPA CPU 100/200/300

Die zu überwachenden Variablen der angeschlossenen VIPA CPU müssen im FP IoT Gateway definiert sein.

Die VIPA-CPUs können natürlich auch an ein Gerät mit MPI Schnittstelle angeschlossen werden. Der Funktionsumfang ist identisch (siehe Kapitel 3.4)

Die VIPA Variablen sind in der External-Gruppe der 'PROCCFG' Datenbank gespeichert:

Verwende COM-Port auf COM2

```
<External>
   <Bus _=”COM2” protocol=”Vipa,GreenCable” type=”Master”
    baud=”38400” TS=”1”>
      <Device _=”2” Pollrate=”1s”>
         <M30 _="M" ind="30" acc="R" />
      </Device>
   </Bus>
</External>
```

Station #2

Der BUS-Parameter enthält die Adresse des COM-POrts “COM2“, den Protokoll-Hersteller “Vipa”, das Protokoll “GreenCable”, den Kommunikationsmodus “Master”, die Baudrate “38400“ und den Parameter TS, welcher die Stationsnummer des FP IoT Gateways enthält.

Das FP IoT Gateway ist gegenüber der VIPA immer ”Master”.
Die Device-ID (Stationskennung) muss mit der SPS-Adresse übereinstimmen.

Wenn das FP IoT Gateway an“COM1” angeschlossen ist, startet die SPS-Kommunikation direkt nach Abziehen des PC-Kabels, und unterbricht automatisch bei Erkennung eines TiXML-Befehls.

Für jede Steuerung muss ein ‘Device’-Abschnitt eingefügt warden, welcher die Stationsnummer (‘_’ – Attribut) und den Abfragezyklus enthält. Nach jedem Abfragezyklus werden die Variablenwerte erneut eingelesen (Master) oder ein Kommunikationstimeout erkannt (Slave).

Es kann eine Liste von Variablen definiert werden:

Variablentyp in der VIPA

```
<AlarmM10 _="M" ind=”10” acc=”R”/>
```

Jede Zeile definiert einen logischen Namen (Alias, z.B. AlarmM10) und den Typ der Variable in der VIPA CPU (siehe: Liste der unterstützten Variablen)

Der ‘ind’ Parameter bestimmt die Adresse der Variable in der Simatic Steuerung und der ‘acc’ Parameter das Zugriffsrecht. Das Zugriffsrecht kann entweder ‘R’ oder ‘RW’ für Lese oder Lese-/Schreibzugriff sein, abhängig von der gewählten Variable.

Der ‘def’ Parameter bestimmt den Startwert der Variable. Eine Variable mit Schreibzugriff enthält diesen Startwert bis zum ersten Schreibzugriff. Eine Variable mit Lesezugriff enthält diesen Wert bis das Gerät den tatsächlichen Wert von der SPS erhalten hat.

Liste der unterstützten Variablen für VIPA System 200V CPU

Typ	index	access	Kommentar
M	0-2047.7	R/W	Merker (Bit)
MB	0-2047	R/W	Merker (Byte)
MW	0-2046	R/W	Merker (Word)
MD	0-2044	R/W	Merker (Doppel-Word)
I	0-15.7	R	Eingänge
Q	0-15.7	R/W	Ausgänge

Das FP IoT Gateway unterstützt auch VIPA System 100V CPU und VIPA System 300V CPU.
Bei Geräten der Aluline Reihe wird die VIPA nur bis Firmware Version 2.0 unterstützt.

Fernzugriff
Für den Fernzugriff auf die SPS ist folgender "TransMode"-Befehl notwendig: (siehe TiXML-Reference manual für weitere Informationen)

```
[<TransMode baud="38400" format="8O1" com="COM1"/>]
```

(Verwenden Sie `com="COM2"` wenn die VIPA an der COM2 RS232 angeschlossen ist)

Um eine Verbindung zu einer VIPA CPU herzustellen, ist lokal ein 11-Bit modem (Unterstützung für 8O1 Datenformat) zu verwenden.

3.7 Moeller Easy 400 / 500 / 600 / 700 / 800 / MFD

Die zu überwachenden Variablen der angeschlossenen Moeller Easy (z.B. easy 400/600/800/MFD) müssen im FP IoT Gateway definiert sein.
Die Moeller Variablen sind in der External-Gruppe der 'PROCCFG' Datenbank gespeichert:

```
<External>
   <Bus _="COM1" protocol="Moeller,Easy 800" type="Master"
    baud="9600" handshake="noDTR">
      <Device _="0" Pollrate="1s">
         <Input1 _="I" ind="1"/>
         <UIn _="AI" ind="7" acc="R" format="F.1;°C" def="1"/>
         <Out4 _="M" ind="2" acc="RW"/>
         <OutClock _="OU" ind="0" acc="R" />
         <OutCount _="OC" ind="0" acc="R" />
         <OutAnalog _="OA" ind="0" acc="R" />
         <OutTimer _="OT" ind="0" acc="R" />
         <Counter  _="IC" ind="0" acc="R" />
         <TextMarker _="D" ind="0" />
         <Timer _="IT" ind="0" />
         <Out2 _="Q" ind="2" acc="W"/>
         <Flag _="M" ind="8"/>
         <Clock _="U" ind="0" acc="RW"/>
      </Device>
   </Bus>
</External>
```

Verwende COM-Port auf Erweiterungskarte C1

Easy 800 Master

Variablenliste

Der BUS-Parameter enthält die Adresse der seriellen Schnittstelle "COM1", den Protokoll-Hersteller "Moeller", das angeschlossene Gerät "Easy 400/600", "Easy 500/700" oder "Easy 800", den Kommunikationsmodus "Master", die Baudrate und das notwendige „noDTR" Handshake.

Für Easy 400//600 wählen Sie "Moeller, Easy 400/600", Baudrate: 4800.

Für Easy 500//700 wählen Sie “Moeller, Easy 500/700”, Baudrate: 4800.
Für Easy 800/MFD wählen Sie “Moeller, Easy 800”, Baudrate: 9600 oder 19200.

Easy 400//600: Der ‘Device’ Eintrag muss “1” sein und den Abfragezyklus enthalten.
Easy 500/700/800/MFD: Der ‘Device” Eintrag muss die Stationskennung der angeschlossenen Easy enthalten. Bei einer einzelnen Easy 500/700/800/MFD ist dies immer “0”. Die Easy 800/MFD unterstützt mehrere Stationen (1-8).

Wenn das FP IoT Gateway an der “MB”-Schnittstelle (Mainboard) angeschlossen ist, startet die SPS-Kommunikation direkt nach Abziehen des PC-Kabels, und unterbricht automatisch bei Erkennung eines TiXML-Befehls.
Es kann eine Liste von Variablen definiert werden:

Variablentyp in der Easy

```
<Alarm11 _="I" ind=”16” acc=”R”/>
```

Jede Zeile definiert einen logischen Namen (Alias, z.B. Alarm11) und den Typ der Variable in der Moeller Easy (siehe: Liste der unterstützten Variablen)
Der ‘ind’ Parameter bestimmt die Adresse der Variable in der Moeller Easy und der ‘acc’ Parameter das Zugriffsrecht. Das Zugriffsrecht kann entweder ‘R’ oder ‘RW’ für Lese oder Lese-/Schreibzugriff sein, abhängig von der gewählten Variable.

Der ‘def’ Parameter bestimmt den Startwert der Variable. Eine Variable mit Schreibzugriff enthält diesen Startwert bis zum ersten Schreibzugriff. Eine Variable mit Lesezugriff enthält diesen Wert bis das Gerät den tatsächlichen Wert von der Easy erhalten hat.

Liste der unterstützten Variablen für Easy 400/600:

Typ	index	access	Kommentar
IK	1-16	R	Eingänge Erweiterung
I	1-8	R	Eingänge
OU	1-4	R	Zeitschaltuhr Ausgang
OC	1-8	R	Zähler Ausgang
OA	1-8	R	Analogwertvergleicher Ausgang
OT	1-8	R	Timer Ausgang
IC	1-8	R	Zähler Istwert
D	1-8	RW	Textanzeige
IT	1-8	R	Timer Istwert
Q	1-8	R(W)	Ausgänge (Schreiben nur im STOP)
M	1-16	RW	Merker
U	-	RW	Uhr
ST	1-8	R	Timer Sollwert
SC	1-8	R	Counter Sollwert
AW	1-8	R	Analogwertvergleicher Sollwert
QK	1-8	R(W)	Ausgänge Erweiterung (Schreiben nur im STOP)
Key	1-4	R	Testen
AI	7,8	R	Analogeingang

Liste der unterstützten Variablen für Easy 500/700:

Typ	index	access	Kommentar
I	1-16	R	Eingänge
Q	1-8	R	Ausgänge
R	1-16	R	Eingang EASY-LINK
S	1-8	R	Ausgang EASY-LINK
OU	1-8	R	Zeitschaltuhr Ausgang, wie OUW (Easy 400/600 Kompatibilität)

Typ	index	access	Kommentar
OUW	1-8	R	Wochenzeitschaltuhr Ausgang
OUY	1-8	R	Jahreszeitschaltuhr Ausgang
OC	1-16	R	Zähler Ausgang
OA	1-16	R	Analogwertvergleicher Ausgang
OT	1-16	R	Timer Ausgang
IC	1-16	RW	Zähler Istwert
D	1-16	R	Textanzeige
IT	1-16	RW	Timer Istwert
IW	1-4	RW	Betriebsstundenzähler Istwert
SW	1-4	RW	Betriebsstundenzähler Sollwert
N	1-16	RW	Merker N
M	1-16	RW	Merker M
U	-	RW	Uhr
ST	1-16	RW	Timer Sollwert 1
ST2	1-16	RW	Timer Sollwert 2
SC	1-16	RW	Zähler Sollwert
AW	1-16	RW	Analogwertvergleicher Sollwert, wie AWI2 (Easy 400/600 Kompatibilität)
AWI1	1-16	RW	Analogwertvergleicher Sollwert 1
AWI2	1-16	RW	Analogwertvergleicher Sollwert 2
AWF1	1-16	RW	Analogwertvergleicher Verstärker
AWF2	1-16	RW	Analogwertvergleicher Verstärker 2
AWOS	1-16	RW	Analogwertvergleicher Offset
AWHY	1-16	RW	Analogwertvergleicher Hysterese
Key	1-4	R	Tasten
AI	7,8.11,12	R	Analogeingang

Liste der unterstützten Variablen für Easy 800 / MFD:

Typ	index	access	Kommentar
I	1-16	R	Eingänge
Q	1-8	RW	Ausgänge (Schreiben nur im STOP)
R	1-16	R	Eingänge Erweiterung
S	1-8	RW	Ausgänge Erweiterung (Schreiben nur im STOP)
M	1-96	RW	Merker
MB	1-96	RW	Byte Merker
MW	1-96	RW	Word Merker
MD	1-96	RW	Doppelword Merker
Key	1-4	R	Tasten
IA	1-4	R	Analogeingänge
QA	-	RW	Analogausgänge
U	-	RW	Uhr

Existiert für eine Variable nur eine Adresse, z.B. “U” ‚kann der Parameter ‚ind’ weggelassen werden.
Wenn nur Lesezugriff ‚R’ möglich ist, kann der Parameter ‚acc’ weggelassen werden.
Wenn die SPS im “RUN” ist, können einige Variablen durch das Easy-Programm überschrieben werden. In diesem Fall müssten Merker vorgeschaltet werden.

Für jedes parametrierte Gerät wird automatisch eine **Type** Variablen eingefügt:

```
[<Get _=”/Process/COM?/D?/Type”/>]
```

Diese Variable enthält den Typ der angeschlossenen Easy, z.B. “412-DC-Rx”.

Fernzugriff
Für den Fernzugriff auf die Easy ist folgender “TransMode”-Befehl notwendig: (siehe TiXML-Reference manual für weitere Informationen)

Easy 400/500/600/700
`[<TransMode baud="4800" format="8N1" handshake=”noDTR” com="COM2"/>]`
(Verwenden Sie `com=”COM1”` wenn die SPS an der COM1 RS232 steckt.)

Easy 800/MFD
`[<TransMode baud="9600" format="8N1" handshake=”noDTR” com="COM2"/>]`
(Verwenden Sie `com=”COM1”` wenn die SPS an der COM1 RS232 steckt.)

Um sich via GSM mit einer Easy zu verbinden, ist mindestens Easy-Soft 5.01 notwendig (Wartezeit einstellbar).

3.8 Moeller PS30 & PS4/40

Die zu überwachenden Variablen der angeschlossenen PS30 oder PS4/40 müssen im FP IoT Gateway definiert sein.
Die Moeller Variablen sind in der External-Gruppe der 'PROCCFG' Datenbank gespeichert:

```
<External>
   <Bus _=”COM1” protocol=”Moeller,SucomA” type=”Master”
    baud=”9600”>
      <Device _=”7” Pollrate=”1s”>
          <Value1 _="S" ind="1" acc=”R”/>
          <Alarmflag _="F" ind="7.0" acc="R"/>
          <Temp _="B" ind="2" acc="RW"/>
          <Power _="R" ind="127" acc="RW" />
          <CountVal _="D" ind="3212" acc="RW" />
      </Device>
   </Bus>
</External>
```

Verwende COM-Port COM1

SucomA Master

Variablenliste

Der BUS-Parameter enthält die Adresse des COM-Ports“COM1“, den Protokoll-Hersteller “Moeller”, das Protokoll “SucomA”, den Kommunikationsmodus “Master” und die Baudrate (4800 bis 57600).

Die Stationsnummer muss “7“ sein.

Wenn das FP IoT Gateway an der “MB”-Schnittstelle (Mainboard) angeschlossen ist, startet die SPS-Kommunikation direkt nach Abziehen des PC-Kabels, und unterbricht automatisch bei Erkennung eines TiXML-Befehls.

Es kann eine Liste von Variablen definiert werden:

Variablentyp in der PS30 & PS4/40

```
<Alarm11 _="F" ind=”16” acc=”R”/>
```

Jede Zeile definiert einen logischen Namen (Alias, z.B. Alarm11) und den Typ der Variable in der Moeller SPS (siehe: Liste der unterstützten Variablen)
Der ‘ind’ Parameter bestimmt die Adresse der Variable in der Moeller SPS und der ‘acc’ Parameter das Zugriffsrecht. Das Zugriffsrecht kann entweder ‘R’ oder ‘RW’ für Lese oder Lese-/Schreibzugriff sein, abhängig von der gewählten Variable.

Der ‘def’ Parameter bestimmt den Startwert der Variable. Eine Variable mit Schreibzugriff enthält diesen Startwert bis zum ersten Schreibzugriff. Eine Variable mit Lesezugriff enthält diesen Wert bis das Gerät den tatsächlichen Wert von der SPS erhalten hat.

Liste der unterstützten Variablen für PS30 & PS4/40:

Typ	index	access	Kommentar
S	0-65535	R	Status WORD
F	0-14999.7	R	Merker BIT
B	0-14999	RW	Merker BYTE
R	0-14998	RW	Merker WORD
D	0-14996	RW	Merker DWORD

Wenn nur Lesezugriff ‚R' möglich ist, kann der Parameter ‚acc' weggelassen werden.
Wenn die SPS im "RUN" ist, können einige Variablen durch das SPS-Programm überschrieben werden. In diesem Fall müssten Merker vorgeschaltet werden.

Fernzugriff
Für den Fernzugriff auf die SPS ist folgender "TransMode"-Befehl notwendig: (siehe TiXML-Reference manual für weitere Informationen)

```
[<TransMode baud="9600" format="8N1" com="COM1"/>]
```
(Verwende `com="COM2"` wenn die SPS an der COM2 RS232 angeschlossen ist)

3.9 SAIA Burgess S-Bus

Die zu überwachenden Variablen der angeschlossenen SAIA-Steuerung (z.B. PCD2) müssen im FP IoT Gateway definiert sein.
Die SAIA Variablen sind in der External-Gruppe der 'PROCCFG' Datenbank gespeichert:

```
<External>
   <Bus _="COM2" protocol="Saia,SBus-DataMode" type="Master"
    baud="19200" handshake="HALF">
      <Device _="1" Pollrate="1s">
          <F100 _="F" ind="100" acc="R" />
      </Device>
      <Device _="3" Pollrate="60s">
          <R102 _="R" ind="102" acc="R" />
      </Device>
   </Bus>
</External>
```

Verwende RS485-Port auf COM2
aktiviert RS485 Modus
S-Bus Station #1
S-Bus Station #3

Das FP IoT Gateway kann als S-BUS Master oder Slave parametriert werden. Alle S-BUS Baudraten werden unterstützt, die Standardbaudrate (kein Eintrag) ist 19200.
Wenn das FP IoT Gateway an der "MB"-Schnittstelle (Mainboard) angeschlossen ist, startet die SPS-Kommunikation direkt nach Abziehen des PC-Kabels, und unterbricht automatisch bei Erkennung eines TiXML-Befehls.

Der Parameter `handshake="HALF"` aktiviert den RS485 Modus auf speziellen Erweiterungskarten.
Für jede S-Bus Station muss ein 'Device'-Eintrag erstellt werden, welcher die Stationsnummer der angeschlossenen Slave-PCD2s bzw. die eigene Stationsnummer als
S-BUS-Slave enthält ('_' – Attribute), sowie den Abfragezyklus.
Nach Ablauf des Abfragezyklus werden die Variablenwerte neu eingelesen (Master) oder ein Kommunikationstimeout erkannt (Slave).

Es kann eine Liste von Variablen definiert werden: Variabletyp im S-BUS (F,T,C,I,O,R)

```
<Alarm11 _="F" ind=”11” acc=”R”/>
```

Jede Zeile definiert einen logischen Namen (Alias, z.B. Alarm11) und den Typ der Variable auf dem S-BUS (siehe: Liste der unterstützten Variablen)
Der ‘ind’ Parameter bestimmt die Adresse der Variable in der PCD2 und der ‘acc’ Parameter das Zugriffsrecht. Das Zugriffsrecht kann entweder ‘R’ oder ‘RW’ für Lese oder Lese-/Schreibzugriff sein, abhängig von der gewählten Variable.
Ein zusätzliches Zugriffsrecht ist ‘RWL’, welches einen gemeinsamen Lese-und Schreibspeicher aktiviert, welcher jedoch nur im Slave-Modus verwendet werden darf. Mit dem ‘RW’ Zugriffsrecht hat das FP IoT Gateway einen getrennten Speicher für Schreib- und Lesezugriffe. In diesem Fall kann die SPS einen Variablenwert in den Lesespeicher des FP IoT Gateways schreiben und das FP IoT Gateway schreibt in den Schreibspeicher der gleichen Variable. Dadurch kann es zu der Situation kommen, dass die SPS z.B. eine 1 schreibt, beim nachfolgenden Auslesen jedoch eine 0 erhält.
Mit ‘RWL’ verwendet das FP IoT Gateway für beide Zugriffe den gleichen Speicher, d.h. es gilt immer der zuletzt geschriebene Wert, egal von wem er gesetzt wurde.

Der ‘def’ Parameter bestimmt den Startwert der Variable. Eine Variable mit Schreibzugriff enthält diesen Startwert bis zum ersten Schreibzugriff. Eine Variable mit Lesezugriff enthält diesen Wert bis das Gerät den tatsächlichen Wert von der SPS erhalten hat.

Liste der unterstützten Variablen für PCD2

Typ	index	access	Kommentar
F	0-8191	R/W	Flag
R	0-4095	R/W	Register
T	0-1600	R	Timer
C	0-1600	R	Counter
O	0-256	R/W	Ausgang
I	0-256	R	Eingang (nur als Master)

PCD2 anschließen

Das FP IoT Gateway kann an der PCD2 an allen 3 seriellen Schnittstellen S0-S2 angeschlossen werden. Es ist lediglich eine 3-Draht-Leitung (RX,TX,GND) notwendig.

Beachten Sie folgende Hinweise:

1. Wenn Sie das FP IoT Gateway an den PGU-Port (S0) der PCD2 anschließen, darf die DSR-Leitung nicht mitgeführt werden, da die PCD2 sonst den S-BUS deaktiviert.
2. Wenn das FP IoT Gateway mit der Mainboard RS232 (MB) an der PCD2 angeschlossen wird, darf die DTR Leitung nicht mitgeführt warden, da der S-BUS sonst deaktiviert wird.

Tixi-MB	PCD2	Tixi-Cx	PCD2	Tixi-RS485	PCD2
2-RxD------	RxD-12/32	2-RxD-----	TxD-12/32	1-GND-----	GND-10/35
3-TxD------	TxD-11/31	3-TxD-----	RxD-11/31	4-R(-)----	RX-TX-11/36
5-GND------	GND-10/30	5-GND-----	GND-10/30	5-R(+)----	/RX-/TX-12/37

Fernzugriff

Für den Fernzugriff auf die SPS ist folgender “TransMode”-Befehl notwendig: (siehe TiXML-Reference manual für weitere Informationen)

FP IoT Gateway RS232 Schnittstelle:
`[<TransMode baud="9600" format="8N1" com="COM1"/>]`
(Verwenden Sie `com=”COM2”`, wenn das FP IoT Gateway an der COM2 RS232 angeschlossen ist)

FP IoT Gateway RS485 Schnittstelle:
`[<TransMode baud="9600" format="8N1" handshake=”HALF” com="COM1"/>]`

Wählen Sie die gleiche Baudrate wie die SPS.

3.10 Carel Macroplus

Die zu überwachenden Variablen der angeschlossenen Carel-Steuerung (z.B. Macroplus) müssen im FP IoT Gateway definiert sein.
Die Carel Variablen sind in der External-Gruppe der 'PROCCFG' Datenbank gespeichert:

```
<External>                                          Verwende RS422-Port auf COM2
   <Bus _="COM2" protocol="Carel,PC2" type="Master"
    handshake="FULL">                               Aktiviert RS422-Modus
      <Device _=”1” Pollrate=”1s”>                  CarelBus Steuerung #1
         <Alarm11 _="D" ind="22" acc="RW"/>
      </Device>
      <Device _=”3” Pollrate=”60s”>                 CarelBus Steuerung #3
         <Alarm31 _="D" ind="22" acc="RW"/>
      </Device>
   </Bus>
</External >
```

Der Parameter `handshake=”FULL”` aktiviert den RS422 Modus. Ohne diesen Eintrag wird RS232 Kommunikation verwendet.

Wenn das FP IoT Gateway an der “MB”-Schnittstelle (Mainboard) angeschlossen ist, startet die SPS-Kommunikation direkt nach Abziehen des PC-Kabels, und unterbricht automatisch bei Erkennung eines TiXML-Befehls.

Für jede auf dem Carel-Bus befindliche Steuerung muss ein ‘Device’ Eintrag eingefügt werden, welcher die Stationsnummer auf dem Carel-Bus (‘_’ – Attribut) sowie den Abfragezyklus enthält. Das FP IoT Gateway fragt die Steuerung nach geänderten Variable ab solange es keinen NULL-Frame erhält, der dem Gerät mitteilt, dass keine weiteren Änderungen vorhanden sind.

Es kann eine Liste von Variablen definiert werden:

```
<Alarm11 _="D" ind=”22” acc=”RW”/>
```

Variablentyp in der Carel SPS (D,I,A)

Jede Zeile definiert einen logischen Namen (Alias, z.B. Alarm11) und den Typ der Variable in der Carel-Steuerung (siehe: Liste der unterstützten Variablen)
Der ‘ind’ Parameter bestimmt die Adresse der Variable in der Carel Steuerung und der ‘acc’ Parameter das Zugriffsrecht. Das Zugriffsrecht kann entweder ‘R’ oder ‘RW’ für Lese oder Lese-/Schreibzugriff sein, abhängig von der gewählten Variable.

Der ‘def’ Parameter bestimmt den Startwert der Variable. Eine Variable mit Schreibzugriff enthält diesen Startwert bis zum ersten Schreibzugriff. Eine Variable mit Lesezugriff enthält diesen Wert bis das Gerät den tatsächlichen Wert von der SPS erhalten hat.

Liste der unterstützten Variablen für Macroplus

Typ	index	access	Kommentar
D	1-183	R/W	Bits
I	1-50	R/W	Integer
A	1-50	R/W	Analog

Macroplus anschließen

Die Macroplus kann über einen RS422-RS232 Adapter oder direkt an eine RS422 Erweiterungskarte angeschlossen werden.

FP IoT Gateway (RS422 Erweiterungskarte) — Macroplus (9pin RS422 weiblich)

T+ ----------R+ (4)
T- ----------R- (5)
R- ----------T- (1)
R+ ----------T+ (2)

Fernzugriff

Für den Fernzugriff auf die SPS ist folgender “TransMode”-Befehl notwendig: (siehe TiXML-Reference manual für weitere Informationen)

FP IoT Gateway RS232 Schnittstelle:

```
[<TransMode baud="1200" format="8N2" com="COM1"/>]
```

(Verwenden Sie `com=”COM2”`, wenn das FP IoT Gateway an der COM2 RS232 angeschlossen ist)

FP IoT Gateway RS422 Schnittstelle:

```
[<TransMode baud="1200" format="8N2" handshake=”FULL” com="COM1"/>]
```

3.11 ABB AC010, AC031, CL Reihe

Der AC010 Logikcontroller ist eine OEM-Version der Moeller EASY 400/600 Produkte.
Siehe Kapitel 3.7 für Details.

Die AC031 wird über das MODBUS RTU Protokol unterstützt.
Siehe Kapitel 4.3 für Details.

Die CL Reihe ist eine OEM-Version der Moeller EASY 500/700 Produkte.
Siehe Kapitel 3.7 für Details.

3.12 Allen Bradley Pico

Diese Serie intelligenter Kleinsteuerrelais ist eine OEM-Version der Moeller Kleinsteuerungen.
Pico Serie A = Moeller Easy 400/600
Pico Serie B = Moeller Easy 500/700
Pico GFX = Moeller MFD

Siehe Kapitel 3.7 für Details.

3.13 Theben PHARAO II

Die PHARAO II Kleinsteuerung ist eine OEM-Version der Mitsubishi Alpha2 (XL).
Siehe Kapitel 3.1 für Details.

4 Feldbus Unterstützung

4.1 Tixi-Bus

Tixi-Bus ist ein einfaches Feldbus Protokol um Variablen effektiv auch mit mehreren SPS-Systemen auszutauschen.
(Schauen Sie auf der InovoLabs Webseite nach weiteren Informatioenen, oder senden Sie eine E-Mail an tixi-support@inovolabs.com)

Die Tixi-Bus Variablen werden in der External Gruppe der 'PROCCFG' Datenbank definiert.

```
<External>
   <Bus _="COM2" protocol="Tixi.Com,Tixi-Bus" type="Master"
    handshake="none" TS="0">
      <Device _="1" Pollrate="1s">
         <Alarm11 _="F" ind="22" acc="RW"/>
      </Device>
      <Device _="3" Pollrate="60s">
         <Alarm31 _="R" ind="243" acc="RW"/>
         <Array _="B" ind="1" no="8" acc="RW"/>
      </Device>
   </Bus>
</External>
```

Verwende RS422-Schnittstelle auf COM2

SPS #1

SPS #3

Das Gerät kann nur als Tixi-Bus “Master“ agieren.

Der Parameter `handshake="mode"` aktiviert verschiedene Kommunikationsmodi:
Bei einer RS232 Schnittstelle ist keine Angabe notwendig.
Mit handshake=“HALF” wird RS485-Kommunikation, und mit “FULL” wird RS 422 Kommunikation verwendet. Für RS 485/422 ist eine spezielle Erweiterungskarte notwendig.

TS ist die Stationsnummer des Geräts.

Wenn das FP IoT Gateway an der “MB”-Schnittstelle (Mainboard) angeschlossen ist, startet die SPS-Kommunikation direkt nach Abziehen des PC-Kabels, und unterbricht automatisch bei Erkennung eines TiXML-Befehls.

Für jede TixiBus Station muss ein ‘Device’-Eintrag erstellt werden, welcher die Stationsnummer der angeschlossenen Slave-SPS-Systeme sowie den Abfragezyklus enthält. Nach Ablauf des Abfragezyklus werden die Variablenwerte neu eingelesen.

Es kann eine Liste von Variablen definiert werden:

```
<Alarm11 _="F" ind="22" acc="RW"/>
```

Variablentyp im TixiBus Protokoll (F,W,…)

Jede Zeile definiert einen logischen Namen (Alias, z.B. Alarm11) und den Typ der Variable in der Steuerung (siehe: Liste der unterstützten Variablen)
Der ‘ind’ Parameter bestimmt die Adresse der Variable in der Steuerung und der ‘acc’ Parameter das Zugriffsrecht. Das Zugriffsrecht kann entweder ‘R’ oder ‘RW’ für Lese oder Lese-/Schreibzugriff sein, abhängig von der gewählten Variable.

Der ‘def’ Parameter bestimmt den Startwert der Variable. Eine Variable mit Schreibzugriff enthält diesen Startwert bis zum ersten Schreibzugriff. Eine Variable mit Lesezugriff enthält diesen Wert bis das Gerät den tatsächlichen Wert von der SPS erhalten hat.

Bei Tixi-Bus können Variablen als Arrays über das Attribut „no“ angelegt werden.
Siehe dazu auch Kapitel 2.4.

Liste der unterstützten Variablen für Tixi-Bus

Typ	index	access	Kommentar
C	0-65535	R/W	Flag (Coil)
W	0-65535	R/W	Word
DW	0-65535	R/W	DWord
F	0-65535	R/W	Float
DF	0-65535	R/W	Double
B	0-65535	R/W	Byte
S	0-65535	R/W	String (erfordert Angabe von “size“)

4.2 ASCII Protocol

Das ASCII protocol ist eine einfache Möglichkeit mit dem FP IoT Gateway Werte aus Geräten auszulesen,die Daten über ein Textprotokoll zu Verfügung stellen.

Das ASCII Protokoll wird in der External Gruppe der 'PROCCFG' Datenbank definiert.

Verwende RS422-Schnittstelle auf COM2

Variablen des abzufragenden Gerätes

```
<External>
   <Bus _="COM2" protocol="Tixi.Com,ASCII" type="Master"
    handshake="none" baud="115200" format="8N1">
      <Device _=”1” Pollrate=”1s”>
         <Float  _="DF"  Pos="14" End="24" acc="R" Radix="K"
          Request="&#13;" ResEnd="Ende&#13;" ResTime="10s"/>
         <Flag _="C"  Pos="0" acc="R" />
         <Word _="W"  Pos="3" End="5" acc="R" />
         <BinWord _="W"  Pos="7" Radix="B" acc="R" />
         <String _="S"  Size="30" Pos="3" Width="10" acc="R"/>
      </Device>
   </Bus>
</External>
```

Der Parameter `handshake=”mode”` aktiviert verschiedene Kommunikationsmodi:
Bei einer RS232 Schnittstelle ist keine Angabe notwendig.
Mit handshake=“HALF” wird RS485-Kommunikation, und mit “FULL” wird RS 422 Kommunikation verwendet. Für RS 485/422 ist eine spezielle Erweiterungskarte notwendig.

Wenn das FP IoT Gateway an der “MB”-Schnittstelle (Mainboard) angeschlossen ist, startet die ASCII-Kommunikation direkt nach Abziehen des PC-Kabels, und unterbricht automatisch bei Erkennung eines TiXML-Befehls.

Für das angeschlossene Gerät muss ein ‘Device’-Eintrag erstellt werden, welcher die Stationsnummer sowie den Abfragezyklus enthält Nach Ablauf des Abfragezyklus werden die Variablenwerte neu eingelesen.

Es kann eine Liste von Variablen definiert werden: Variablentyp im ASCII Protokoll (C,W,…)

```
<Float  _="DF"  Pos="14" End="24" acc="R" Radix="K"
Request="&#13;" ResEnd="Ende&#13;" ResTime="10s"/>
```

Jede Zeile definiert einen logischen Namen (Alias, z.B. Alarm11), den Typ der Variable im Gerät (siehe: Liste der unterstützten Variablen) und der ‘acc’ Parameter das Zugriffsrecht ’R’ für “Lesen“. Die weiteren Parameter werden in den folgenden Kapiteln beschrieben.

Der ‘def’ Parameter bestimmt den Startwert der Variable. Eine Variable enthält diesen Wert bis das Gerät den tatsächlichen Wert von der SPS erhalten hat.

4.2.1 Definition der Variablenabfrage-Strings

Das Format der Variablenabfragen kann für jede Variable mittels folgender Parameter bestimmt werden:

Request String, der an das Gerät zum Abfordern des Datenpaketes gesendet werden soll.

Wait Gibt die Zeit an, die das FP IoT Gateway wartet, bevor der Requeststring an das Gerät geschickt wird.

ResTime Gibt die Zeit an, die das FP IoT Gateway wartet bis das ersten Zeichen der Antwort kommt.

CharTime Gibt die Zeit an, die maximal zwischen den Zeichen vergehen darf. Wird diese Zeit überschritten, gilt die Antwort als beendet.

Wenn **ResTime** ohne **CharTime** verwendet wird, ist ResTime die Gesamtzeit für den Empfang der Antwort. Nach dieser Zeit gilt die Antwort als abgeschlossen, egal ob noch weitere Zeichen kommen.

ResEnd String, der das Ende der Nachricht markiert. Wird ResEnd nicht angegeben, gilt nur das angegebene Timeout (ResTime/CharTime). Ansonsten müssen beide erfüllt sein.

4.2.2 Auswertung der Strings

Die Paramater gelten solange, d.h. für alle nach einer Anfrage erhaltenen Antworten, bis ein neues Format einer anderen Variablenabfrage bestimmt wurde.

Find Im Empfangspuffer wird nach diesem String gesucht.

Pos Das einzulesende Datenfeld befindet sich an der angegebeben Position.
Diese bezieht sich auf den Anfang des empfangenen Textes oder, wenn angegeben, auf das Ende des Suchstrings (Find).

FindPos Damit kann die Suche ab der angegebenen Position gestartet werden.

End Das Ende des einzulesende Datenfeldes befindet sich an der angegebeben Position. Diese bezieht sich auf den Anfang des empfangenen Textes oder, wenn angegeben, auf das Ende des Suchstrings (Find).

Width Alternativ zur Endposition kann die Größe des Datenfeldes angegeben werden.

4.2.3 Modifizierung der Werte

Die Datentypen können durch die Angabe von "**Radix**" modifiziert werden.

"D" dezimal
"H" hexadezimal
"O" oktal
"B" binaer

Fließkommaformate:

"K" Vor- und Nachkommastellen durch Kommata getrennt. Punkte werden als Tausendertrennzeichen gewertet und ignoriert.
Ein 'E/e' wird als Exponent gewertet. '+/-' als Vorzeichen. Alles andere außer Ziffern stoppt die Wandlung.

"P" Vor- und Nachkommastellen durch Punkt getrennt. Die Kommata werden als Tausendertrennzeichen gewertet und ignoriert.
Ein 'E/e' wird als Exponent gewertet. '+/-' als Vorzeichen. Alles andere außer Ziffern stoppt die Wandlung.

Die Default-Radix sind abhängig vom Datentyp:

Datentyp	Radix
B,W,DW,I	D
F,DF	P
C,S	"" (leerer Text)

Anmerkung zu den Flags:
Die Flags werden als ein Zeichen eingelesen. Ist dieses "J/j/Y/y/1" ergibt es ein TRUE (1). Die Zeichen "N/n/0" ergeben ein FALSE (0).

Liste der unterstützten Variablen für das ASCII Protokoll

Typ	index	access	Kommentar
C	0-65535	R/W	Flag
W	0-65535	R/W	Word
DW	0-65535	R/W	DWord
F	0-65535	R/W	Float
DF	0-65535	R/W	Double
B	0-65535	R/W	Byte
S	0-65535	R/W	String
I	0-65535	R/W	Integer

4.3 Modbus RTU Master

Die zu überwachenden Variablen der angeschlossenen Modbus-Steuerung müssen im FP IoT Gateway definiert sein.
Die Modbus Variablen sind in der External-Gruppe der 'PROCCFG' Datenbank gespeichert:

Verwende COM-Port auf COM2

```
<External>
   <Bus _="COM2" protocol="Modbus,RTU" type="Master" baud="19200">
      <Device _="1" Pollrate="1s" DWordInc="1" Timeout="3000s">
         <Coil1 _="C" ind="0x2000" acc="RW"/>
         <Input1 _="I" ind="0x03E0" acc="R"/>
         <InputReg5 _="R" ind="0x2005" acc="R"/>
         <Register10 _="H" ind="0x200A" swap="1" acc="RW" def="1"/>
         <Register3 _="D" ind="0x4003" acc="RW" def="1"/>
      </Device>
   </Bus>
</External >
```

Variablenliste

Der BUS-Parameter enthält die Portaddresse der Erweiterungskarte, das Protokol “Modbus,RTU”,den Kommunikationsmodus “Master” und die verwendete Baudrate.

Unterstützte Baudraten: 1200, 4800, 9600, 14400, 19200, 38400, 57600, 115200

Wenn das FP IoT Gateway an der “MB”-Schnittstelle (Mainboard) angeschlossen ist, startet die SPS-Kommunikation direkt nach Abziehen des PC-Kabels, und unterbricht automatisch bei Erkennung eines TiXML-Befehls.

Für die Modbus Station muss ein ‘Device’-Eintrag erstellt werden, welcher die Stationsnummer der angeschlossenen Slave-SPS enthält (‘_’ – Attribute), sowie den Abfragezyklus. Nach Ablauf des Abfragezyklus werden die Variablenwerte neu eingelesen.

Im Device Abschnitt sind spezielle Parameter möglich, die die Modbus Kommunikation zwischen Gerät und SPS regeln:

CharTimeout	Timeout zwischen den Zeichen (50ms,)
Pause	Pause zwischen den Nachrichten (50ms)
Timeout	Timeout für Antwort (300ms)
DwordSwap	Muss gesetzt werde, wenn Low vor High Word in DWORD gesendet wird (0)
ForceSingleWordWrite	Sollte immer auf 1 gesetzt sein
UseCache	Ist der Wert auf 0 gesetzt, wird das Zusammenfassen von aufeinanderfolgenden Variablen zu blockweisen Abfragen (Caching) deaktiviert. Alle Variablen werden in einzelnen Abfragen geholt. (Default: 1)
MaxElements	Begrenzt die beim Caching in einer Modbus-Nachricht abgefragten Elemente.
swap	Mit dem optionalen Parameter swap kann pro Datenpunkt die Reihenfolge der Bytes innerhalb eines 16 Bit Wertes vertauscht werden. Standardwert=0 (MSB first; kann in diesem Fall weggelassen werden) Wird swap=“1“ definiert, dann werden die Bytes innerhalb eines 16 Bit Registers (auch in DWord Registern) vertauscht (LSB first).

Es kann eine Liste von Variablen definiert werden:

Variablentyp in der SPS (C,I,H,…)

```
<Alarm11 _="C" ind="0x03E3" acc="R" read="1" write="5"/>
```

Jede Zeile definiert einen logischen Namen (Alias, z.B. `Alarm11`) und den Typ der Variable in der Modbus-Steuerung (siehe: Liste der unterstützten Variablen)
Der '`ind`' Parameter definiert die Adresse der Modbus RTU Variable als HEX-Code und der '`acc`' Parameter das Zugriffsrecht. Das Zugriffsrecht kann entweder '**R**' oder '**RW**' für Lese oder Lese-/Schreibzugriff sein, abhängig von der gewählten Variable. Fügt man dem `acc` Attributwert ein **C** hinzu (z.B. '**RWC**'), wird die betreffende Variable nicht sofort abgefragt sondern geprüft, ob die folgende auch noch in der Abfrage mitgelesen werden kann (caching, falls `UseCache=0`).

Der '`def`' Parameter bestimmt den Startwert der Variable. Eine Variable mit Schreibzugriff enthält diesen Startwert bis zum ersten Schreibzugriff. Eine Variable mit Lesezugriff enthält diesen Wert bis das Gerät den tatsächlichen Wert von der SPS erhalten hat. Wenn eine Kommunikation mit der SPS besteht, wird der Startwert beim Starten der Buskommunikation in das SPS-Register geschrieben.

Liste der unterstützten Variablen für Modbus RTU:

Typ	index	access	Kommentar
C	0-65535	RW	Coil (single bit)
I	0-65535	R	Discrete input
R	0-65535	R	Input register (unsigned)
H	0-65535	RW	Holding register (WORD Marker, unsigned)
D	0-65534	RW	Holding Register (DWORD Marker, unsigned)
RI	0-65535	R	Input register (signed integer)
HI	0-65535	RW	Holding register (WORD Marker, signed integer)
DI	0-65534	RW	Holding Register (DWORD Marker, signed integer)
RX	0-65534	R	Input register (DWORD Marker, unsigned)
RY	0-65534	R	Input Register (DWORD Marker, signed)
DF	0-65534	RW	Holding Register (DWORD) Wert wird als float interpretiert (nur sinnvoll, wenn das Modbus-Gerät tatsächlich einen 4-Byte langen float-Wert liefert)
RY	0-65534	R	Input Register (DWORD Marker, signed)
RXF	0-65534	R	Input Register (DWORD) Wert wird als float interpretiert (nur sinnvoll, wenn das Modbus-Gerät tatsächlich einen 4-Byte langen float-Wert liefert)
HS	0-65534	R	Input Register (WORD Marker, unsigned) Der Inhalt von einem oder mehrerer zusammenhängender Input Register wird hier als String interpretiert. Dazu muss noch eine Stringlänge mit size="x" angegegen werden. x muss durch 2 teilbar sein. Nur Kombination mit simpleType="String" verwenden !

Typ	index	access	Kommentar
			Kann auch in Kombination mit dem swap-Parameter verwendet werden.
D2	0-65532	RW	Holding Register (QWORD = 64 Bit, unsigned)
DI2	0-65532	RW	Holding Register (QWORD = 64 Bit, signed)
RX2	0-65532	R	Input Register (QWORD = 64 Bit, unsigned)
RY2	0-65532	R	Input Register (QWORD = 64 Bit, signed)
D2F	0-65532	RW	Holding Register (QWORD = 64 Bit) Wert wird als float interpretiert (nur sinnvoll, wenn das Modbus-Gerät tatsächlich einen 8-Byte langen float-Wert liefert)
RX2F	0-65532	R	Input Register (QWORD) Wert wird als float interpretiert (nur sinnvoll, wenn das Modbus-Gerät tatsächlich einen 8-Byte langen float-Wert liefert)
RB	0-65535	R	Input register (unsigned) => interpretiert als BCD (4 Ziffern)
HB	0-65535	RW	Holding register (WORD Marker, unsigned) => interpretiert als BCD (4 Ziffern)
DB	0-65534	RW	Holding Register (DWORD Marker, unsigned) => interpretiert als BCD (8 Ziffern)
RXB	0-65534	R	Input register (DWORD Marker, unsigned) => interpretiert als BCD (8 Ziffern)
D2B	0-65532	RW	Holding Register (QWORD = 64 Bit, unsigned) => interpretiert als BCD (16 Ziffern)
RX2B	0-65532	R	Input Register (QWORD = 64 Bit, unsigned) => interpretiert als BCD (16 Ziffern)

Wenn nur Lesezugriff ‚R' möglich ist, kann der Parameter ‚acc' weggelassen werden.

Modbus function codes

Das FP IoT Gateway verwendet, abhängig vom Variablentyp, folgende Modbus Function-Codes:

Code (dezimal)	Variablentyp
1 - Read Coil Status	C
2 - Read Input Status	I
3 - Read Holding Registers	H, HI, D, DI, DF, D2, DI2, HB, DB, D2B
4 - Read Input Registers	R, RI, RX, RY, RXF, RX2, RY2, RX2F, RB, RXB, RX2B
5 - Force Single Coil	C
6 – Preset Single Register	H, R, D, HI, RI, DI, HS, DL, DI2, D2F, HB, DB, D2B (wenn ForceSingleWordWrite=1)
15 – Force Multiple Coil	C (wenn size>1)

Über die Variablenattribute "read" und "write" können die Function-Codes geändert werden.

Beispiel für das Auslesen von Strings über den Modbus-Variablentyp HS:

```
<String_1 _="HS" simpleType="String" size="10" swap="1" ind="761" acc="R"/>
```

Im Beispiel werden 5 aufeinanderfolgende Holding Register (jeweils 2 Bytes lang) ab Adresse 761 ausgelesen und als String interpretiert. Nicht druckbare Zeichen werden dabei einfach weggelassen. Innerhalb eines 16 Bit Registers werden die Bytes vertauscht (wegen swap="1").

Fernzugriff

Für den Fernzugriff auf die SPS ist folgender "TransMode"-Befehl notwendig: (siehe TiXML-Reference manual für weitere Informationen)

```
[<TransMode baud="19200" format="8N1" com="COM1"/>]
```

(Verwenden Sie `com="COM2"` wenn die SPS an der RS232 COM2 steckt.)

Beispiel für 64 Bit Zugriffe auf Modbus Register

Ab der Firmware-Version 5.2.6.26 können auch 64 Bit breite Registerwerte verarbeitet werden (dazu werden 4x 16 Bit Register zusammengefasst).
Die neuen 64 Bit Variablentypen heißen D2, DI2, RX2, RY2, D2F, RX2F
Im folgenden Beispiel ist die Nutzung der neuen Datentypen in der External dargestellt:

```
[<SetConfig _="PROCCFG" ver="v">
<External>
<Bus _="ETH" Name="ModbusTCP" protocol="Modbus,TCP" type="Master">
    <Device Name="Device_1" _="1" IP="169.254.9.2" Pollrate="10s" Timeout="300ms"
     DWordInc="2" DwordSwap="0" ForceSingleWordWrite="0" UseCache="1">
        <Dbl_0 _="D2F" simpleType="Double" size="1" swap="0" ind="0" acc="RW"/>
        <Dbl_4 _="RX2F" simpleType="Double" size="1" swap="0" ind="4" acc="R"/>

        <Ui64_8 _="D2" simpleType="Uint64" size="1" swap="0" ind="8" acc="RW"/>
        <Ui64_12 _="RX2" simpleType="Uint64" size="1" swap="0" ind="12" acc="R"/>

        <I64_16 _="DI2" simpleType="Int64" size="1" swap="0" ind="16" acc="RW"/>
        <I64_20 _="RY2" simpleType="Int64" size="1" swap="0" ind="20" acc="R"/>

        <Fl_30 _="DF" simpleType="Float" size="1" swap="0" ind="30" acc="RW"/>
        <Fl_32 _="RXF" simpleType="Float" size="1" swap="0" ind="32" acc="R"/>
    </Device>
</Bus>
</External>
</SetConfig>]
```

Eine dazu passende Logdefinition könnte so aussehen:

```
[<SetConfig _="LOG" ver="y">
<LogDefinition>
<LogFiles>
    <JobReport size="10240"/>
    <Event size="10240"/>
    <Login size="10240"/>
    <IncomingMessage size="10240"/>
    <FailedIncomingCall size="10240"/>
    <SupportLog size="102400"/>
    <Datalogging_0 record="Datalogging_0" size="1024000" contenttype="binary"/>
</LogFiles>
<Records>
    <Datalogging_0>
        <!-- 16 Bit -->
        <H0 Name="H0" _="Uint16" path="/Process/Bus1/Device_0/H0"/>
        <H1 Name="H1" _="Int16" path="/Process/Bus1/Device_0/H1"/>

        <!-- 32 Bit -->
        <D0 Name="D0" _="Uint32" path="/Process/Bus1/Device_0/D0"/>
        <DI0 Name="DI2" _="Int32" path="/Process/Bus1/Device_0/DI0"/>
        <DF2 Name="DF2" _="Float" path="/Process/Bus1/Device_0/DF2"/>
        <RX0 Name="RX0" _="Uint32" path="/Process/Bus1/Device_0/RX0"/>
        <RY0 Name="RY0" _="Int32" path="/Process/Bus1/Device_0/RY0"/>
        <RXF2 Name="RXF2" _="Float" path="/Process/Bus1/Device_0/RXF2"/>
```

```
        <!-- 64 Bit -->
        <D20 Name="D20" _="Uint64" path="/Process/Bus1/Device_0/D20"/>
        <DI20 Name="DI20" _="Int64" path="/Process/Bus1/Device_0/DI20"/>
        <DF20 Name="D2F0" _="Double" path="/Process/Bus1/Device_0/DF20"/>
        <IRX20 Name="RX20" _="Uint64" path="/Process/Bus1/Device_0/IRX20"/>
        <IRY20 Name="RY20" _="Int64" path="/Process/Bus1/Device_0/IRY20"/>
        <IRX2F0 Name="RX2F" _="Double" path="/Process/Bus1/Device_0/IRX2F0"/>
    </Datalogging_0>
</Records>
</LogDefinition>
</SetConfig>]
```

Beispiel für BCD Modbus Variablentypen

Ab der Firmware-Version 5.2.6.26 können Modbus-Registerwerte als BCD-Werte interpretiert werden. Die BCD-Variablentypen heißen RB, HB, DB, RXB, D2B, RX2B.

Im folgenden Beispiel ist die Nutzung der neuen BCD-Datentypen in der External dargestellt:

```
[<SetConfig _="PROCCFG" ver="v">
<External>
<Bus _="ETH" Name="ModbusTCP" protocol="Modbus,TCP" type="Master">
  <Device Name="Device_1" _="1" IP="169.254.9.2" Pollrate="10s" Timeout="300ms"
   DWordInc="2" DwordSwap="0" ForceSingleWordWrite="0" UseCache="1">
      <!-- 16 bit BCD HR -->
      <Var_BCD_HB Name="Var_BCD_HB" _="HB" ind="0" acc="RW" swap="1"/>
      <!-- 16 bit BCD IR -->
      <Var_BCD_RB Name="Var_BCD_RB" _="RB" ind="0" acc="R" swap="1"/>
      <!-- 32 bit BCD HR -->
      <Var_BCD_DB Name="Var_BCD_DB" _="DB" ind="0" acc="RW" swap="1"/>
      <!-- 32 bit BCD IR -->
      <Var_BCD_RXB Name="Var_BCD_RXB" _="RXB" ind="0" acc="R" swap="1"/>
      <!-- 64 bit BCD HR -->
      <Var_BCD_D2B Name="Var_BCD_D2B" _="D2B" ind="0" acc="RW" swap="1"/>
      <!-- 64 bit BCD IR -->
      <Var_BCD_RX2B Name="Var_BCD_RX2B" _="RX2B" ind="0" acc="R" swap="1"/>
    </Device>
</Bus>
</External>
</SetConfig>]
```

Eine dazu passende Logdefinition könnte so aussehen:

```
[<SetConfig _="LOG" ver="y">
<LogDefinition>
<LogFiles>
    <JobReport size="10240"/>
    <Event size="10240"/>
    <Login size="10240"/>
    <IncomingMessage size="10240"/>
    <FailedIncomingCall size="10240"/>
    <SupportLog size="102400"/>
    <Datalogging_0 record="Datalogging_0" size="1024000" contenttype="binary"/>
</LogFiles>

<Records>
    <Datalogging_0>
        <!-- 16 Bit BCD -->
        <Var_BCD_HB _="Uint16" Name="16 bit BCD HR"
                    path="/Process/Bus1/Device_0/Var_BCD_HB"/>
        <Var_BCD_RB _="Uint16" Name="16 bit BCD IR"
                    path="/Process/Bus1/Device_0/Var_BCD_RB"/>
        <!-- 32 Bit BCD -->
        <Var_BCD_DB _="Uint32" Name="32 bit BCD HR"
                    path="/Process/Bus1/Device_0/Var_BCD_DB"/>
```

```
        <Var_BCD_RXB _="Uint32" Name="32 bit BCD IR"
                     path="/Process/Bus1/Device_0/Var_BCD_RXB"/>
        <!-- 64 Bit BCD -->
        <Var_BCD_D2B _="Uint64" Name="64 bit BCD HR"
                     path="/Process/Bus1/Device_0/Var_BCD_D2B"/>
        <Var_BCD_RX2B _="Uint64" Name="64 bit BCD IR"
                     path="/Process/Bus1/Device_0/Var_BCD_RX2B"/>
    </Datalogging_0>
</Records>
</LogDefinition>
</SetConfig>]
```

4.3.1 Modbus automatische Konfiguration

Informationen zur automatischen Modbus Konfiguration finden Sie im Dokument Modbus_AutoKonfiguration.pdf.

4.4 Modbus ASCII Master

Die Modbus ASCII Parametrierung ist ähnlich der Modbus RTU (Kapitel 4.3).
Der Bus `protocol` Parameter muss **“Modbus,ASCII”** lauten.
Die speziellen Modbus RTU Device-Parameter (CharTimeout, Pause, Timeout, DWordInc, DwordSwap, ForceSingleWordWrite, UseCache) sind bei Modbus ASCII nicht gültig.

4.5 Modbus TCP Master

Tixi Geräte ab der "Generation 6" (H600 und Wand.Box Serie) unterstützen Modbus TCP.
Die Modbus TCP Variablen werden in der External Sektion der 'PROCCFG' Datenbank definiert.

Verwende LAN interface

```
<External>
   <Bus _="ETH" protocol="ModbusTCP" type="Master" >
      <Device _=”1” IP="IPAddress" Pollrate=”1s” Timeout=""
       DWordInc="2" DwordSwap="0" ForceSingleWordWrite="0"
       UseCache="1">
         <Input1 _="I" ind="0x03E0" acc="R"/>
         <Coil1 _="C" ind="0x2000" acc=”RW”/>
         <InputReg5 _="R" ind="0x2005" acc="R"/>
         <Register10 _="H" ind="0x200A" swap="1" acc="RW" def="1"/>
         <Register3 _="D" ind="0x4003" acc="RW" def="1"/>
      </Device>
   </Bus>
</External >
```

Set of Variables

Die BUS-Parameter legen den LAN Port, das Busprotokoll “ModbusTCP” und den Modus“Master” fest.

Parameter der Device-Sektion:

`IPAddress` IP-Adresse der Steuerung

Weitere Parameter können in der Device-Sektion agegeben werden, um die Kommunikation mit der Steuerung gezielt zu steuern:

`CharTimeout`	Timeout zwischen den Zeichen (50ms,)
`Pause`	Pause zwischen den Nachrichten (50ms)
`Timeout`	Timeout für Antwort (300ms)
`DwordSwap`	Muss gesetzt werde, wenn Low vor High Word in DWORD gesendet wird (0)

`ForceSingleWordWrite`

Sollte immer auf 1 gesetzt werden

`UseCache` Ist der Wert auf 0 gesetzt, wird das Zusammenfassen von aufeinanderfolgenden Variablen zu blockweisen Abfragen (Caching) deaktiviert. Alle Variablen werden in einzelnen Abfragen geholt. (Default: 1)

`swap` Mit dem optionalen Parameter swap kann pro Datenpunkt die Reihenfolge der Bytes innerhalb eines 16 Bit Wertes vertauscht werden.
Standardwert=0 (MSB first; kann in diesem Fall weggelassen werden)
Wird swap="1" definiert, dann werden die Bytes innerhalb eines 16 Bit Registers (auch in DWord Registern) vertauscht (LSB first).

Für jedes Device können Variablen definiert werden:

Variable type in Modbus RTU protocol (z.B. C,I,R,H,D)

```
<Alarm11 _="C" ind="1" simpleType="Bit" acc="RW" def="0"/>
```

Jede Zeile definiert einen logischen Namen (Alias, z.B. Alarm11) und den Typ der Variable in der Modbus-Steuerung (siehe: Liste der unterstützten Variablen)
Der 'ind' Parameter definiert die Adresse der Modbus TCP Variable und der 'acc' Parameter das Zugriffsrecht. Das Zugriffsrecht kann entweder 'R' oder 'RW' für Lese oder Lese-/Schreibzugriff sein, abhängig von der gewählten Variable.

Der '`def`' Parameter bestimmt den Startwert der Variable. Eine Variable mit Schreibzugriff enthält diesen Startwert bis zum ersten Schreibzugriff. Eine Variable mit Lesezugriff enthält diesen Wert bis das Gerät den tatsächlichen Wert von der SPS erhalten hat. Wenn eine Kommunikation mit der SPS besteht, wird der Startwert beim Starten der Buskommunikation in das SPS-Register geschrieben.

Liste der unterstützten Variablen für Modbus TCP Master:

Typ	index	access	Kommentar
C	0-65535	RW	Coil (single bit)
I	0-65535	R	Discrete input
R	0-65535	R	Input register (unsigned)
H	0-65535	RW	Holding register (WORD Marker, unsigned)
D	0-65534	RW	Holding Register (DWORD Marker, unsigned)
RI	0-65535	R	Input register (signed integer)
HI	0-65535	RW	Holding register (WORD Marker, signed integer)
DI	0-65534	RW	Holding Register (DWORD Marker, signed integer)
RX	0-65534	R	Input register (DWORD Marker, unsigned)
RY	0-65534	R	Input Register (DWORD Marker, signed)
DF	0-65534	RW	Holding Register (DWORD) Wert wird als float interpretiert (nur sinnvoll, wenn das Modbus-Gerät tatsächlich einen 4-Byte langen float-Wert liefert)
RY	0-65534	R	Input Register (DWORD Marker, signed)
RXF	0-65534	R	Input Register (DWORD) Wert wird als float interpretiert

Typ	index	access	Kommentar
			(nur sinnvoll, wenn das Modbus-Gerät tatsächlich einen 4-Byte langen float-Wert liefert)
HS	0-65534	R	Input Register (WORD Marker, unsigned) Der Inhalt von einem oder mehrerer zusammenhängender Input Register wird hier als String interpretiert. Dazu muss noch eine Stringlänge mit size="" angegegen werden. Size muss durch 2 teilbar sein. Nur Kombination mit simpleType="String" verwenden ! Kann auch in Kombination mit dem swap-Parameter verwendet werden.
D2	0-65532	RW	Holding Register (QWORD = 64 Bit, unsigned)
DI2	0-65532	RW	Holding Register (QWORD = 64 Bit, signed)
RX2	0-65532	R	Input Register (QWORD = 64 Bit, unsigned)
RY2	0-65532	R	Input Register (QWORD = 64 Bit, signed)
D2F	0-65532	RW	Holding Register (QWORD = 64 Bit) Wert wird als float interpretiert (nur sinnvoll, wenn das Modbus-Gerät tatsächlich einen 8-Byte langen float-Wert liefert)
RX2F	0-65532	R	Input Register (QWORD) Wert wird als float interpretiert (nur sinnvoll, wenn das Modbus-Gerät tatsächlich einen 8-Byte langen float-Wert liefert)
RB	0-65535	R	Input register (unsigned) => interpretiert als BCD (4 Ziffern)
HB	0-65535	RW	Holding register (WORD Marker, unsigned) => interpretiert als BCD (4 Ziffern)
DB	0-65534	RW	Holding Register (DWORD Marker, unsigned) => interpretiert als BCD (8 Ziffern)
RXB	0-65534	R	Input register (DWORD Marker, unsigned) => interpretiert als BCD (8 Ziffern)
D2B	0-65532	RW	Holding Register (QWORD = 64 Bit, unsigned) => interpretiert als BCD (16 Ziffern)
RX2B	0-65532	R	Input Register (QWORD = 64 Bit, unsigned) => interpretiert als BCD (16 Ziffern)

Wenn nur Lesezugriff ‚R' möglich ist, kann der Parameter ‚acc' weggelassen werden.

Modbus function codes

Das FP IoT Gateway verwendet, abhängig vom Variablentyp, folgende Modbus Function-Codes:

Code (dezimal)	Variablentyp
1 - Read Coil Status	C
2 - Read Input Status	I
3 - Read Holding Registers	H, HI, D, DI, DF, HS
4 - Read Input Registers	R, RI, RX, RY, RXF
5 - Force Single Coil	C
6 – Preset Single Register	H, R, D, HI, RI, DI (wenn ForceSingleWordWrite=1)
15 – Force Multiple Coil	C (wenn size>1)
16 - Preset Multiple Registers	H, R, D, HI, RI, DI

Über die Variablenattribute “read” und “write” können die Function-Codes geändert werden. Register werden standardmäßig über FC16 geschrieben. Wenn FC6 benötigt wird, kann dieser mit “ForceSingleWordWrite” aktiviert werden.

Beispiel

Abfrage eines Modbus TCP Gerätes auf Adresse 193.101.167.29, Port 503 jede Sekunde.

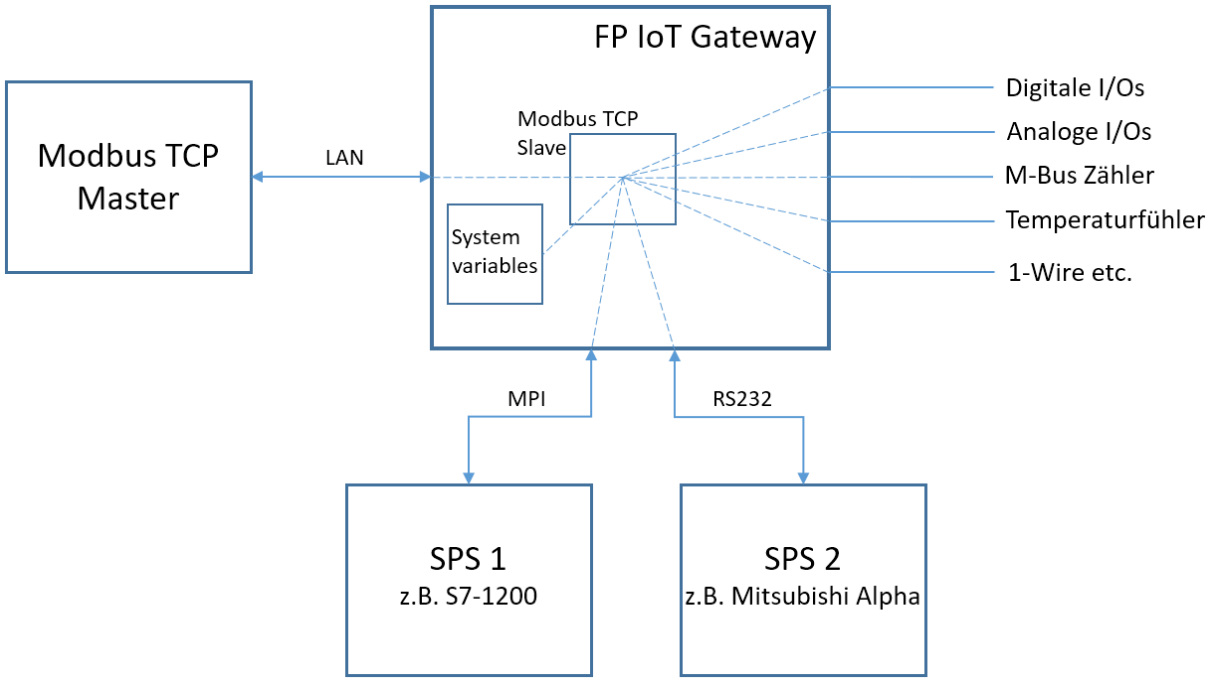
```
<Device Name="Device_31" NameUser="Software-Slave-HR" _="1"
 IP="193.101.167.29" port="503" Pollrate="1s" DWordInc="2"
 DwordSwap="0" ForceSingleWordWrite="1" UseCache="1">
   <Variable_3100 Name="H0" _="H" simpleType="Uint16" ind="0" acc="RW"/>
   <Variable_3101 Name="H1" _="H" simpleType="Uint16" ind="1" acc="RW"/>
   <Variable_3102 Name="H2" _="H" simpleType="Uint16" ind="2" acc="RW"/>
   <Variable_3103 Name="H3" _="H" simpleType="Uint16" ind="3" acc="RW"/>
   <Variable_3104 Name="H4" _="H" simpleType="Uint16" ind="4" acc="RW"/>
   <String_1 Name="String" _="HS" simpleType="String" ind="10" size="6"
             swap="1" acc="R"/>
</Device>
```

Neue Variablentypen für 64 Bit und BCD-Werte

Die Verwendung der neuen 64 Bit Variablentypen D2, DI2, RX2, RY2, D2F, RX2F und der BCD-Variablentypen RB, HB, DB, RXB, D2B, RX2B entspricht der Beschreibung im Modbus RTU Protokoll (siehe Kapitel 4.3). Diese Variablentypen sind ab Firmware Version 5.2.6.26 nutzbar.

4.6 Modbus TCP Slave

Ab Firmware Version 5.2.7.22 kann ein Modbus TCP Slave konfiguriert werden.
Der ModBus TCP Slave ermöglicht es, von einem anderen Leitsystem aus beliebige im FP IoT Gateway definierte Prozessdaten zu lesen und zu schreiben. Die vom Slave zur Verfügung gestellten Datenpunkte werden dabei als Speicherzellen genutzt oder optional internen Busvariablen, Prozessvariablen oder anderen System Properties über einen Referenzpfad zugeordnet. Der Slave kopiert dann die zugeordneten Variablen in seine interne Variablenstruktur und stellt sie dem Master zum Lesen und Schreiben zur Verfügung. Damit können z.B. Datenpunkte einer am FP IoT Gateway angeschlossenen Steuerung (z.B. SPS von Siemens, ABB o.ä.) oder eine im FP IoT Gateway definierte Prozessvariable vom Modbus TCP Master gelesen und geschrieben werden.



Die Modbus Variablen sind in der External-Gruppe der 'PROCCFG' Datenbank gespeichert:

```
<External>                                    Verwende LAN interface
  <Bus _="ETH" name="Bus1" protocol="ModbusTCP" type="Slave" >
    <Device _="1" Name="Device_0" Pollrate="10s">
      <Input1 _="I" ind="1000" def="0" path="/Process/PV/I1"/>
      <Coil1 _="C" ind="2000" def="1" path="/Process/Bus1/D1/C1"/>
      <InputReg5 _="R" ind="3000" def="1"/>
      <Register10 _="H" ind="4000" def="20"/>
      <Register3 _="D" ind="5000" def="41"/>
    </Device>                                          Variablenliste
 </Bus>
</External >
```

Parameter der Device-Sektion:

`Pollrate` Aktualisierungsintervall der Slave-Variablen (nur bei Nutzung von path=)

Wird ein Referenzpfad verwendet, dann werden die Slave-Variablen im Zyklus der Pollrate lesend und schreibend aktualisiert. Bei einer Pollrate von 10s werden also alle 10 Sekunden die Werte aus dem referenzierten Pfad in die Slave-Variable übernommen bzw. wird der vom Master in die Slave-Variable geschriebene Wert in die referenzierte Variable geschrieben.

Wichtige Einschränkungen:

1. Der TCP-Port ist fest auf 502 eingestellt.
2. Es ist nur ein Slave definierbar.
3. Es kann nur ein Master gleichzeitig auf den Slave zugreifen.

Die TCP-Verbindung wird vom Slave nach 180s Inaktivität getrennt.

Parameter der Variablen-Sektion:

Es kann eine Liste von Variablen definiert werden: Variablentyp des Slave (C,I,H,…)

```
<Coil1 _="C" ind="1000" def="0" path="/Process/Bus1/D1/C1"
simpleType="Bit" />
```

Jede Zeile definiert einen logischen Namen (Alias, z.B. `Coil1`) und den Typ der Variable im Modbus-Slave (siehe Tabelle: Liste der unterstützten Variablen für Modbus TCP Slave).

`Coil1` Name der Variable. Variablentypen siehe "Liste der unterstützten Variablen.."

Es gibt 4 Basistypen:
C = Coil, I = Discrete Input, R = Input Register, H = Holding Register

Von den Basistypen R und H sind weitere abgeleitete Typen definiert:

Input Register: R, RI, RX, RY, RX2, RY2, RXF, RX2F
Holding Register: H, HI, D, DI, D2, DI2, DF, D2F

`ind` Index für den Zugriff durch den Master.

Der Index ist innerhalb der Basistypen der in der Tabelle „Liste der unterstützten Variablen für Modbus TCP Slave" angegebenen Grenzen frei wählbar, müssen aber innerhalb der Basistypen eindeutig sein und sollten aufeinander folgen.

`def` Startwert der Variable

Für jede Variable muss zwingend ein Default-Wert definiert werden.
Wenn kein Defaultwert definiert wurde, dann ist der Defaultwert = 0.

Die Defaultwerte werden unter folgenden Umständen verwendet:

- Beim Start des Slave-Busses, wenn für die Slave-Variable keine Referenzvariable verwendet wurde.
- Wenn für die Slave-Variable eine Referenz konfiguriert wurde und die Referenz nicht auflösbar ist.

`path` (optional): Referenzpfad

Ein Referenzpfad kann ein beliebiger Zugriffspfad auf interne Datenstrukturen sein, z.B. Prozessvariablen (`/Process/PV/…`), Busvariablen von angeschlossenen Steuerungen (`BusX/DeviceY/…`), allgemeine System Properties (`/GSM/…` etc.), Onboard-Schnittstellen (`/Process/MB/IO/…`) oder Schnittstellen von Erweiterungsmodulen (`/Process/Cxxx/…`).
Wird ein Referenzpfad verwendet, dann werden die Slave-Variablen im Zyklus der Pollrate lesend und schreibend aktualisiert. d.h. wenn sich der Wert der über den Referenzpfad angegebenen Variable ändert, wird auch die Slave-Variable aktualisiert (bei Slave-Lesevariablen) bzw. der Wert der über den Referenzpfad angegebenen Variable geschrieben (bei Slave-Schreibvariablen), wenn die Slave-Variable vom Modbus-TCP-Master aus geschrieben wird.

Der Anwender muss sich für den Modbus Slave ein Schema überlegen, welche Register und in welcher Reihenfolge die Register vom Master abrufbar sind.

Bitte beachten:
Die Slave-Variablen können auch über einen Set-Befehl z.B. in einem EventHandler, einer Berechnung innerhalb der ProcessVars oder über die Befehlskonsole geschrieben werden. Ist der optionale path-Parameter für eine Slave-Variable definiert und die Slave-Variable wird über einen Set-Befehl geschrieben, dann wird beim nächsten Pollzyklus die Slave-Variable wieder mit dem Wert der Referenzvariablen aktualisiert.

Liste der unterstützten Variablen für Modbus TCP Slave:

Typ	index	access	simpleType	Kommentar
C	0-65535	RW	Bit	Coil (single bit)
I	0-65535	R	Bit	Discrete input
R	0-65535	R	Uint16	Input register (unsigned)
H	0-65535	RW	Uint16	Holding register (WORD Marker, unsigned)
D	0-65535	RW	Uint32	Holding Register (DWORD Marker, unsigned)
RI	0-65535	R	Int16	Input register (signed integer)
HI	0-65535	RW	Int16	Holding register (WORD Marker, signed integer)
DI	0-65535	RW	Int32	Holding Register (DWORD Marker, signed integer)
RX	0-65535	R	Uint32	Input register (DWORD Marker, unsigned)
RY	0-65535	R	Int32	Input Register (DWORD Marker, signed)
DF	0-65535	RW	Float	Holding Register (DWORD) Wert wird als float interpretiert
RXF	0-65535	R	Float	Input Register (DWORD) Wert wird als float interpretiert
D2	0-65535	RW	Uint64	Holding Register (QWORD = 64 Bit, unsigned)
DI2	0-65535	RW	Int64	Holding Register (QWORD = 64 Bit, signed)
RX2	0-65535	R	Uint64	Input Register (QWORD = 64 Bit, unsigned)
RY2	0-65535	R	Int64	Input Register (QWORD = 64 Bit, signed)
D2F	0-65535	RW	Double	Holding Register (QWORD = 64 Bit) Wert wird als double interpretiert
RX2F	0-65535	R	Double	Input Register (QWORD) Wert wird als double interpretiert

Die access-Parameter in der Tabelle beziehen sich auf den Master und ergeben sich implizit aus dem Variablentyp. Sie können daher weggelassen werden. Ein Coil ist z.B. vom Master aus immer lesbar und schreibbar, ein Discret Input Register nur lesbar.

Für die Registertypen DF und RXF muss als simpleType zwingend Float verwendet werden.
Für die Registertypen D2F und RX2F muss als simpleType zwingend Double verwendet werden.

Modbus function codes aus Sicht des Modbus Masters
Der Modbus TCP Master kann je nach Variablentyp folgende Modbus Function-Codes verwenden:

Code (dezimal)	Variablentyp
1 - Read Coil Status	C
2 - Read Input Status	I
3 - Read Holding Registers	H, HI, D, DI, D2, DI2, DF, D2F
4 - Read Input Registers	R, RI, RX, RY, RX2, RY2, RXF, RX2F
5 - Force Single Coil	C
6 – Preset Single Register	H, HI, D, DI, D2, DI2, DF, D2F
15 – Force Multiple Coils	C (wenn size>1)
16 - Preset Multiple Registers	H, HI, D, DI, D2, DI2, DF, D2F

Datenformat der Slave-Variablen beim Zugriff über Modbus TCP

Der Zugriff auf die Slave-Variablen erfolgt im Big-Endian-Format.

Beispiel fuer die Abfrage einer 32-Bit Variable vom Typ "D", Uint32:

```
<HR2_UI32 _="D" ind="4002" def="305419896" />
```

Anfrage 2 Holding Register von Adresse `4002` (`0x0fa0`):

```
00 01 00 00 00 06 00 03 0F A2 00 02
```

Antwort, 4 Datenbyte `0x12, 0x34, 0x56, 0x78` (dezimal `305419896`):

```
00 01 00 00 00 07 00 03 04 12 34 56 78
```

Regeln für die External-Konfiguration des Modbus TCP Slave

Folgende Regeln bei der Erstellung der External-Konfiguration sind zwingend zu beachten:

1. Variablen eines Basistypen liegen virtuell direkt aufeinanderfolgend in der Reihenfolge nach der sie in der External definiert werden (es gibt keine Lücken).
2. Jeder Index ("ind") muss pro Basistyp einmalig sein.
3. Die Indizes sollten aufsteigend in der External definiert werden (Übersichtlichkeit). Schrittweite kann 1 sein - unabhängig von der wahren Größe der Variablen.
4. Beim ModBus-Zugriff auf mehrere Register wird über die Startadresse die entsprechende Start-Variable ausgewählt, die angeforderten Register werden aus dieser gefüllt, und weitere angeforderte Register werden dann mit den in der External nachfolgend definierten Variablen gleichen Basis-Typs gefüllt. Die Indizes der folgenden Variablen spielen dabei keine Rolle, die Reihenfolge in der External ist entscheidend.
   Sollten nicht genug Folge-Register existieren, wird mit Fehler geantwortet!
   (-> Lesen/Schreiben von mehr Registern als definiert sind: Fehler).
   Beim Schreiben mehrerer Variablen werden die Ziel-Register beschrieben, für deren Variablen komplette Quelldaten vorliegen (für eine 64-Bit-Variable müssen also 4 Registerwerte geschrieben werden). Die letzte Variable würde im Fehlerfall nicht verändert werden.
5. Die Startvariable muss existieren, d.h. die Startadresse muss in der External definiert sein, sonst Fehlermeldung.

Modbus TCP Slave Referenzkonfiguration

Die folgende Modbus TCP Slave Referenzkonfiguration demonstriert die Nutzung aller verfügbaren Variablentypen. Die im letzten Teil gezeigten Beispiele mit Referenzpfad setzen voraus, dass die Referenzpfade auch verfügbar sind.

```
[<SetConfig _="PROCCFG" ver="y">
<External>

<!-- Modbus Slave Definition aller funktionablen Registertypen -->
<!-- Die angegebenen Zugriffsarten beziehen sich auf den Master -->

<!-- fixer ip-port=502 -->
<Bus _="ETH" Name="Bus1" protocol="ModbusTCP" type="Slave">
  <Device _="0" Name="Device_0" Pollrate="10s">

  <!-- coil, Modbus-Access: RW, Read Coil Status(1), Force Single Coil (5),
       Force Multiple Coils (15) -->
    <!-- Coil (Bit), acc="RW", simpleType="Bit" -->
    <COIL0 _="C" ind="1000" def="0" />
```

```
  <!-- discrete input, Modbus-Access: R only, Read Input Status (2) -->
  <DIN0 _="I" ind="2000" def="1" /> <!-- Discrete Input (Bit), acc="R",
              simpleType="Bit" -->

<!-- input register, Modbus-Access: R only, Read Input Registers (4) -->

  <!-- Input Register (Word), unsigned, acc="R", simpleType="Uint16" -->
  <IR0_UI16 _="R" ind="3000" def="3000" />

  <!-- Input Register (Word), signed, acc="R", simpleType="Int16" -->
  <IR1_I16 _="RI" ind="3001" def="-3001" />

  <!-- Input Register (DWord), unsigned, acc="R", simpleType="Uint32" -->
  <IR2_UI32 _="RX" ind="3002" def="3002" />

  <!-- Input Register (DWord), signed, acc="R", simpleType="Int32" -->
  <IR3_I32 _="RY" ind="3003" def="-3003" />

  <!-- Input Register (QWord), unsigned, acc="R", simpleType="Uint64" -->
  <IR4_UI64 _="RX2" ind="3004" def="3004" />

  <!-- Input Register (QWord), signed, acc="R", simpleType="Int64" -->
  <IR5_I64 _="RY2" ind="3005" def="-3005" />

  <!-- Input Register (DWord), Interpretation als float, acc="R",
       braucht simpleType="Float" -->
  <IR10_RXF _="RXF" ind="3500" def="3.45"   simpleType="Float" />

  <!-- Input Register (QWord), Interpretation als float, acc="R",
       braucht simpleType="Double" -->
  <IR11_RX2F _="RX2F" ind="3501" def="4.56" simpleType="Double" />

<!-- holding register, Modbus-Access: RW, Read Holding Registers (3),
     Preset Single Register (6), Preset Multiple Registers (16) -->

  <!-- Holding Register (Word), unsigned,acc="RW",simpleType="Uint16" -->
  <HR0_UI16 _="H" ind="4000" def="4660" />

  <!-- Holding Register (Word), signed, acc="RW", simpleType="Int16" -->
  <HR1_I16 _="HI" ind="4001" def="-4001" />

  <!-- Holding Register (DWord), unsigned, "RW", simpleType="Uint32" -->
  <HR2_UI32 _="D" ind="4002" def="305419896" />

  <!-- Holding Register (DWord), signed, acc="RW", simpleType="Int32" -->
  <HR3_I32 _="DI" ind="4003" def="-4003" />

  <!-- Holding Register (QWord), unsigned, "RW", simpleType="Uint64" -->
  <HR4_UI64 _="D2" ind="4004" def="81985529216486895"/>

  <!-- Holding Register (QWord), signed, acc="RW", simpleType="Int64" -->
  <HR5_I64 _="DI2" ind="4005" def="-4005" />

  <!-- Holding Register (DWord), Interpretation als float, acc="RW",
       braucht simpleType="Float" -->
  <HR10_DF _="DF" ind="4500" def="1.23"  simpleType="Float" />

  <!-- Holding Register (QWord), Interpretation als float, acc="RW",
       braucht simpleType="Double" -->
  <HR11_D2F _="D2F" ind="4501" def="2.34"   simpleType="Double" />
```

```
    <!-- Beispiele mit Referenz zu einer internen Variable in path -->
    <!-- wenn Referenz nicht erreichbar, dann Defaultwert (def) -->

    <!-- Input Register (Word), unsigned, acc="R", simpleType="Uint16" -->
    <IR6_UI16 _="R" ind="3006" def="3006"
              path="/Process/C092/AI_AAAATPPSSB/P9" />

    <!-- Input Register (Word), unsigned, acc="R", simpleType="Uint16" -->
    <IR7_UI16 _="R" ind="3007" def="999" path="/Process/PV/MyVar1" />

    <!-- Holding Register (Word), signed, "RW", simpleType="Int16" -->
    <HR20_I16 _="HI" ind="4600" def="-2000"
               path="/Process/Bus1/Device_0/HR1_I16" />

  </Device>
</Bus>
</External>
</SetConfig>]
```

4.7 M-Bus

M-Bus (Meter-Bus) ist ein Feldbus Protokoll, um Energie- und Klimageräte auf effiziente Art und Weise zu überwachen, auch wenn mehrere Geräte am FP IoT Gateway angeschlossen sind.

Für den einfachen Anschluss von M-Bus Geräten sind spezielle M-Bus Geräte mit Pegelwandler verfügbar. M-Bus Geräte können ebenso über einen externen M-Bus/RS232 Konverter (Pegelwandler) angeschlossen werden.

Die M-Bus Implementierung arbeitet als Bus-Master mit der Standard-Baudrate 2400 (konfigurierbar), 8 Datenbits, 1 Stopbit, gerader Parität und ohne Handshake.

Für jedes M-BUS Gerät muss ein 'Device'-Eintrag erstellt werden, welcher mindestens die Primäradresse (`PrimaryAddr,` dezimal) oder die Sekundäradresse (`SecondaryAddr,` 8 dezimals) oder Fabrikationsadresse (`FabricationAddr,` 8 dezimals) des Gerätes sowie den Abfragezyklus enthalten muss.

Die M-Bus Variablen sind in der External Gruppe der 'PROCCFG' Datenbank registriert.

```
<External>
   <Bus _=”COM3” protocol=”meterbus” type="Master" baud="Baudrate"
    format="Format" Timeout="Timeout">
      <Device _=”1” PrimaryAddr=”123” SecondaryAddr="12345678"
       Pollrate=”1s”>
         <SecondaryAddr _="ident" acc="R"/>
         <Var01 ind=”1”/>
         <Var02 ind=”2”/>
      </Device>
      <Device _=”2” SecondaryAddr=”12345678” Pollrate=”60s”>
         <Var01 ind=”1”/>
         <Time _=”DateTime”/>
      </Device>
   </Bus>
</External>
```

Verwende M-BUS Schnittstelle an COM3

Gerät 1

Gerät 2

Beschreibung der möglichen Attribute (kursiv dargestellt):

baud	Baudrate. Standard=2400 Baud. Mögliche Werte: 300, 2400, 9600
format	Datenformat. Standard=8E1
Timeout	maximale Wartezeit zwischen den Abfragen
PrimaryAddr	Primäradresse
SecondaryAddr	Sekundäradresse (immer 8-stellig)
Pollrate	Abfragerate

Wenn alle Geräte auf dem M-Bus die gleiche Primäradresse haben, können die Geräte nur über die Sekundäradresse addressiert werden. In diesem Fall darf die Primäradresse NICHT in der Device-Definition verwendet werden !

Optionale Parameter sind "`ManufactoryCode`" (3 ASCII Zeichen), "`Generation`" (hex) und "`Medium`" (hex) welche als weitere Unterscheidungsmerkmale für Geräte mit gleicher Adresse verwendet werden können.

Es kann eine Liste von Variablen definiert werden:

```
<Var01 ind=”1”/>
```

Jede Zeile definiert einen logischen Namen (Alias, z.B. Var01) und der "ind"-Parameter die Position der Variable im M-BUS Telegramm.

Ausgabe der Gerätekennungen

Die Gerätekennungen „Primäradresse“, „Sekundäradresse“, „Herstellercode“ und weitere können bei der Abfrage des Gerätes z.B. für das Logging mit ausgegeben werden.
Dazu müssen spezielle Einträge in der Variablenliste vorgenommen werden:

```
<PrimaryAddr _="primary"/>
<SecondaryAddr _="ident"/>
<Manufacturer _="manufacturer"/>
<State _="stat"/>
<Medium _="medium"/>
<Version _="version"/>
```

(verfügbar ab FW 5.1.6.6)

Spezielle Initialisierungen

Time:
Um die FP IoT Gateway RTC Zeit an ein M-BUS Gerät zu übertragen, muss die Variable “Time” (nicht lesbar) im Device-Abschnitt des Gerätes definiert werden:

```
<Time _=”DateTime”/>
```

Reset:
Zu Beginn der Kommunikation kann ein “Reset Code” zum M-BUS Gerät gesendet werden. Die notwendige Variable “ResCode” ist nicht lesbar und enthält den Reset-Code als Parameter “def” (Wert=0-255):

```
<ResCode _="Reset" def="114"/>
```

Rohdaten Initialisierung:
Zu Beginn der Kommunikation wird ein benutzerdefiniertes Datagram (Parameter def) zum M-BUS Gerät gesendet:

```
<Init _="Raw" def="7304FD0834120000"/>
```

Die def Zeichenkette muss in hex Bytes angegeben werden. Das erste Byte ist das CI-Feld gefolgt von den Rohdaten ohne Prüfsumme.

VIF – Value Information Field: Medium/Einheit:
Beim Lesen des M-BUS Variablen kann das Gerät die Informationen des VIF bzgl. Medium und Einheit und weitere Werte ausgeben. Weitere Informationen: siehe Kapitel 5.

Datenlogging:

Beim Logging von M-Bus Werten werden, um keine M-Bus Daten zu verlieren, im Binärlogfile standardmäßig 37Byte pro Wert beansprucht (32 Byte String + 5 Byte Datentyp).
Sollen reine M-Bus Zahlenwerte geloggt werden, kann im Logfilerecord durch Angabe des Attribut size der verwendete Speicher auf die tatsächlich notwendige Größe reduziert werden.
Bei reinen Zahlenwerten empfehlen wir eine Größe von 13 Byte (8 Byte Wert + 5 Byte Datentyp), z.B.:

```
<Datalogging_0>
 <Variable_0 _="meterbus" path="/Process/Bus1/Dev_0/Variable_0"/>
 <Variable_1 _="meterbus" size="13" path="/Process/Bus1/Dev_0/Variable_1"/>
</Datalogging_0>
```

Weitere Informationen zum Datenlogging und zur Berechnung der Logfilegröße finden sie im TiXML-Recerence-Manual.

Fernzugriff

Für den Fernzugriff auf die Geräte ist folgender “TransMode”-Befehl notwendig: (siehe TiXML-Reference manual für weitere Informationen)

FP IoT Gateway M-BUS Schnittstelle:

```
[<TransMode baud="2400" format="8E1" com="COM2"/>]
```

4.7.1 M-Bus Scan

FP IoT Gateways verfügen über einen Mechanismus, um alle am M-Bus angeschlossenen Geräte automatisch zu erkennen (Scan). Folgende Voraussetzungen gelten:

- alle Zähler werden über Sekundäradresse angesprochen
- alle Zähler verwenden eine einheitliche Einstellungen für Baudrate und Datenformat

Folgender Befehl startet einen Scan auf dem M-Bus:

```
[<ScanDevices _="COM3" protocol="meterbus" type="Master" ver="v"/>]
```

Das Ergebnis eines Scans liefert das FP IoT Gateway eine Liste aller gefundenen Zähler und deren Variablen zurück. Je nach Firmwareversion des FP IoT Gateways liefert der Scan eine unterschiedliche Anzahl von Werten zurück.

Geräte-Infos:

Wert	Beschreibung
PrimaryAddr	Primäradresse
SecondaryAddr	Sekundäradresse
Manufacturer	Herstellercode
Version	Version
Medium	Medium
State	Status

Variablen-Infos:

Wert	Beschreibung
ind	Index im M-Bus Telegramm
vib	Value information block
value	Wert
storage	storage number
tariff	Tarif-Info
subunit	subunit
Function	Function code

Optional können ein Reset-Code und eine einzelne Sekundäradresse angegeben werden:

```
[<ScanDevices _="COM3" protocol="meterbus" type="Master"
 SecondaryAddr="12345678" Reset="192" ver="v"/>]
```

Beispiel

Qundis WTT16 Knoten (Device 1) mit 1 angeschlossenem Heizkostenverteiler (Device 2):

```
<ScanDevices>
  <Device  PrimaryAddr="2"  SecondaryAddr="10307448"  Manufacturer="LSE"
    Version="30"  Medium="Bus/System"  State="0">
  <Var01  ind="1"  vib="On Time [h]"  value="6032"
    storage="0"  tariff="0"  subunit="0"  function="0"/>
  <Var02  ind="2"  vib="Time Point [Date+Time]"  value="2016/07/12,14:18"
    storage="0"  tariff="0"  subunit="0"  function="0"/>
  <Var03  ind="3"  vib="Model / Version"  value="3444564688926"
    storage="0"  tariff="0"  subunit="0"  function="0"/>
  <Var04  ind="4"  vib="Parameter Set Ident."  value="WTT16"
    storage="0"  tariff="0"  subunit="0"  function="0"/>
  <Var05  ind="5"  vib="Time Point [Date]"  value="19127/15/31"
    storage="0"  tariff="0"  subunit="0"  function="3"/>
  <Var06  ind="6"  vib="Bus Address"  value="258"
    storage="0"  tariff="0"  subunit="0"  function="0"/>
  <Var07  ind="7"  vib="% BATT"  value="99"
    storage="0"  tariff="0"  subunit="0"  function="0"/>
</Device>

<Device  PrimaryAddr=""  SecondaryAddr="90510717"  Manufacturer="LSE"
    Version="51"  Medium="Heat_Cost_Alloc."  State="0">
  <Var01  ind="1"  vib="Heat Cost Allocator"  value="289"
    storage="0"  tariff="0"  subunit="0"  function="0"/>
  <Var02  ind="2"  vib="Heat Cost Allocator"  value="12"
    storage="1"  tariff="0"  subunit="0"  function="0"/>
  <Var03  ind="3"  vib="Time Point [Date]"  value="2015/11/30"
    storage="1"  tariff="0"  subunit="0"  function="0"/>
  <Var04  ind="4"  vib="Time Point [Date]"  value="19127/15/31"
    storage="0"  tariff="0"  subunit="0"  function="3"/>
  <Var05  ind="5"  vib="Time Point [Date+Time]"  value="2016/07/12,10:36"
    storage="0"  tariff="0"  subunit="0"  function="0"/>
  <Var06  ind="6"  vib="Size of Storage"  value="8"
    storage="8"  tariff="0"  subunit="0"  function="0"/>
  <Var07  ind="7"  vib="Time Point [Date]"  value="2016/06/30"
    storage="15"  tariff="0"  subunit="0"  function="0"/>
  <Var08  ind="8"  vib="Storage Interval [months]"  value="1"
    storage="8"  tariff="0"  subunit="0"  function="0"/>
  <Var09  ind="9"  vib="Heat Cost Allocator"  value="289"
    storage="15"  tariff="0"  subunit="0"  function="0"/>
  <Var10  ind="10"  vib="Heat Cost Allocator"  value="289"
    storage="14"  tariff="0"  subunit="0"  function="0"/>
  <Var11  ind="11"  vib="Heat Cost Allocator"  value="276"
    storage="13"  tariff="0"  subunit="0"  function="0"/>
  <Var12  ind="12"  vib="Heat Cost Allocator"  value="259"
    storage="12"  tariff="0"  subunit="0"  function="0"/>
  <Var13  ind="13"  vib="Heat Cost Allocator"  value="226"
    storage="11"  tariff="0"  subunit="0"  function="0"/>
  <Var14  ind="14"  vib="Heat Cost Allocator"  value="207"
    storage="10"  tariff="0"  subunit="0"  function="0"/>
  <Var15  ind="15"  vib="Heat Cost Allocator"  value="63"
    storage="9"  tariff="0"  subunit="0"  function="0"/>
  <Var16  ind="16"  vib="Heat Cost Allocator"  value="12"
    storage="8"  tariff="0"  subunit="0"  function="0"/>
  <Var17  ind="17"  vib="Model / Version"  value="2645700378675"
    storage="0"  tariff="0"  subunit="0"  function="0"/>
  <Var18  ind="18"  vib="Bus Address"  value="258"
    storage="0"  tariff="0"  subunit="0"  function="0"/>
</Device>
```

Hinweis:
Der Scan kann unter Umständen sehr lange dauern, wenn sehr viele M-Bus Geräte am Bus angeschlossen sind (bis zu mehrere Stunden).

4.7.2 Starten der M-Bus Auslesung

Die angeschlossenen M-Bus Zähler können mit dem folgenden Kommando gezielt ausgelesen werden:

```
[<StartBusPolling _=“COM-Port“/>]
```

Busname = `COM3 [| COM1 | COM2]`

Sobald das Kommando gestartet wurde, kann man dies an der entsprechenden System Property sehen:

`/Process/COM1PollActive 0` = nicht aktiv `1` = Polling läuft
`/Process/COM2PollActive 0` = nicht aktiv `1` = Polling läuft
`/Process/COM3PollActive 0` = nicht aktiv `1` = Polling läuft

Beispiel

Start des M-Bus Polling auf COM3:

```
[<StartBusPolling _=”COM3”/>]
```

Dabei wird die folgende System Property angelegt:

`/Process/COM3MBusPollActive` `0` = nicht aktiv `1` = MBus Polling läuft

4.8 CAN-Bus

Noch nicht freigegeben.

4.9 CS-Protokoll (EN 62056-21 Mode C / EN 61107)

FP IoT Gateways unterstützen das CS-Protokoll nach EN 62056-21 Mode C (nur Datenabfrage).
Dieses Protokoll wird häufig bei Stromzählern verwendet.

Die Variablen sind in der External Gruppe der 'PROCCFG' Datenbank registriert.

Verwende Schnittstelle COM2

```
<External>
   <Bus _="COM2" Name="BusName" protocol="DIN1107,CSData"
    type="Master" baud="Speed" handshake="HALF" format="DataFormat"
    BusPollrate="BusPollrate TUnit">
      <Device _="1" Address="xxxxxx" Pollrate="1s" PreSend="PreSend"
       PreDelay="PreDelay" OpMode="OpMode" Options="Options">
         <Manufacturer _="MC" acc="R" simpleType="String"/>
         <Identification _="ID" acc="R" size="16"
          simpleType="String" format=";%% "/>
         <DevID_OBISCode _="OBIS" obis="0.0.0" acc="R" size="5"
          simpleType="String" format=";%% "/>
         <DevID_DevID _="Text" obis="0.0.0" acc="R" size="8"
          simpleType="String" format=";%% "/>
         <DevID_DevIDDW _="DWord" obis="0.0.0" acc="R"
          simpleType="Uint32" format=";%% "/>
         <Cnt_OBISCode _="OBIS" obis="3.8.1" acc="R" size="5"
          simpleType="String" format=";%% "/>
         <Cnt_CounterDbl _="Count" obis="3.8.1" acc="R"
          simpleType="Double" format="F09,3;%% "/>
         <Cnt_Counter _="DWord" obis="3.8.1" acc="R"
          simpleType="Uint32" format="F09,3;%% "/>
         <Cnt_Unit _="Text" obis="3.8.1" acc="R" size="3"
          subind="1" simpleType="String" format=";%% "/>
         <Cnt_Date _="Text" obis="1.6.1" acc="R" size="8"
          simpleType="String" format=";(%%)"/>
         <OBIS_DevIDCode _="OBISLine" ind="1" acc="R" size="15"
          simpleType="String" format=";%% "/>
         <OBIS_Error _="OBISLine" ind="0" acc="R" size="15"
          simpleType="String" format=";%% "/>
         <Obis_180 _="Count" ProgCmd="R1" simpleType="Double"
           obis="1.8.0" subindVal="0" acc="R"/>
      </Device>
   </Bus>
</External>
```

Die Zähler werden an eine RS485-Schnittstelle an das FP IoT Gateway angeschlossen.
Alle rot dargestellten XML-Attribute sind erforderliche Attribute mit konstanten Werten.

4.9.1 Bus-Konfiguration

Beschreibung der konstanten (rot gekennzeichneten) Attribute:

protocol	Definiert das Busprotokoll: "DIN1107,CSData"
type	Definiert die Rolle des FP IoT Gateways im Bus als "Master"
handshake	Bestimmt die Art der Flusskontrolle zwischen FP IoT Gateway und Zähler. Bei Nutzung des 2-Draht RS485 Busses muss eine Halbduplex-Verbindung konfiguriert werden, daher "HALF"
COM	erforderlich Schnittstelle des FP IoT Gateways, an dem der Bus angeschlossen wird.

Werden 2 Busse definiert, müssen sich die Schnittstellen unterscheiden.

Wert	Beschreibung
COM2	RS 485 Schnittstelle auf COM2
COM*X*	Weitere RS485-Schnittstelle (falls im FP IoT Gateway vorhanden) *X* = Nummer der Schnittstelle

Übersicht der möglichen Parameter (kursiv geschriebene Werte):

BusName optional, Standard ist: "" (leer). Maximal 20 alphanumerische Zeichen, darf nicht mit einer Zahl beginnen.

Definiert den Namen der TiXML-Attributgruppe, die den Bus in der "Process"-Attributgruppe repräsentiert. Der Gruppenname ist der erste Teil einer Gerätevariablen-Adresse:

Process/***Bus***/*Device/Variable*

Die kursiv geschriebenen Namen werden durch die Prozesskonfiguration definiert. Der fett geschriebene Name wird durch dieses Attribut definiert.

Wert	Beschreibung
"" (leer)	(Standardwert) Auto-Name. Die Attributgruppe bekommt einen automatischen Namen zugewiesen: Process/Busn/... n automatisch generierte Indexnummer
BusName (nicht leer)	Die Attributgruppe erhält den angegebenen Namen: Process/MeinName/... Beispiel: <Bus Name="CS" ...> Process/CS/...

Speed optional, Standard: 9600
Gültige Werte: 300, 600, 1200, 2400, 4800, **9600**, 19200, 38400, 115200

DataFormat optional, Standard: 7E1
Bestimmt das Datenformat auf der seriellen Schnittstelle.
Syntax: **DataBitsParityBitsStopBits**

DataBits: 8 = 8 Datenbits; 7 = 7 Datenbits
ParityBits: N = kein Paritätsbit; E = gerade Parität; O = ungerade Parität
StopBits: 1 = 1 Stopbit; 2 = 2 Stopbits

BusPollrate optional, Standard: "200ms"

Zeit in Zeiteinheiten (siehe TUnit), die das FP IoT Gateway mindestens zwischen zwei Bus-Abfragen auf dem Bus wartet. Damit lässt sich ein Blockieren des Prozessors bei "leerer" Buskonfiguration verhindern.

Wertebereich:
0ms ... 1073741823ms
0s ... 1073740s
0m ... 17894m
0h ... 297h

TUnit Zeiteinheit (ms, s, m, h)

4.9.2 Device-Konfiguration

Die Gerätekonfiguration wird durch ein "Device" XML Element innerhalb des Bus Elements definiert und durch folgende XML-Attribute beschrieben:

```
<External>
   <Bus ...>
      <Device _="ID" Name="Alias" Address="Address" PollRate="Rate
       TUnit" Opmode="OpMode" Options="Options">
              <<< Konfiguration der Geräte-Variablen >>>
      </Device>
   </Bus>
</External>
```

Für jeden Zähler, der an der Schnittstelle (COM2 oder COMX) angeschlossen ist und mit dem das FP IoT Gateway kommunizieren soll, muss ein "Devcice"-Eintrag angelegt werden.

Beschreibung der *(kursiv und fett geschriebenen)* Attributwerte*:*

ID erforderlich, Zahlen, 0 ..255

Bus-ID des Zählers. Die Adresse ist im Prinzip nicht erforderlich, da im CS-Protokoll keine laufenden Nummern für das Gerät benötigt werden. Sie dient der Unterscheidung, falls mehrere Devices konfiguriert werden.

Address optional, Standard:““(leer)
String alphanumerisch und Leerzeichen, max. 30 Zeichen

Kommunikationsadresse des Zählers (in der Regel die Seriennummer). Die Adresse ist nur erforderlich, wenn mehrere Zähler an einem Bus angeschlossen sind. Ist die Adresse leer (Standard) wird jedes Gerät angesprochen (siehe EN 62056-21:2002).

Alias optional, Standardwert: “” (leer), max. 20 alphanumerische Zeichen, darf nicht mit einer Zahl beginnen.

Definiert den Namen der TiXML Attributgruppe, die das Gerät in der „Bus“ Attributgruppe repräsentiert. Der Gruppenname ist der zweite Teil einer Gerätevariablen- adresse:

Process/*Bus*/***Device***/*Variable*

Die kursiv geschriebenen Namen werden durch die Prozesskonfiguration definiert. Der fett geschriebene Name wird durch dieses Attribut definiert.

Wert	Beschreibung
“” (leer)	(Standartwert) Auto-Name. Die Attributgruppe erhält einen automatisch generierten Namen zugewiesen: Process/*BusName/*D*Id…* *Id* Wert des Attributs “_”
DeviceName (nicht leer)	Die Attributgruppe erhält den angegebenen Namen: Process/*Bus/DeviceName…* Beispiel: <Bus Name=”CS” ….> <Device Name=”Meter1” ….> Process/CS/Meter1/…

Rate optional, Standard: „15m“, gültige Werte, siehe unten
Pollrate in Zeiteinheiten (siehe TUnit), mit der das FP IoT Gateway das Gerät abfragt. Wertebereiche siehe "BusPollrate"

TUnit Zeiteinheit (ms, s, m, h)

PreSend sendet den angegeben Wert (0-255) binär vor Beginn der Kommunikation mit dem Zähler

PreDelay Wartezeit in ms nach Senden des unter PreSend angegeben Wertes

OpMode Betriebsmodus; Standard: „data“ (kann dann weggelassen werden)
„prog“ = Programmiermodus. Dieser Modus ist für spezielle Kommandos Wie z.B. „R1“ erforderlich.

Options Optionen (optional; Standard = keine Optionen)
„RemainConnected“ = In diesem Modus wird die Verbindung zum Zähler bei jedem Durchlauf aufrecht erhalten. Dadurch können die Zählerwerte schneller abgefragt werden. Dieser Modus wird oft in Verbindung mit dem OpMode „prog“ und R1-Kommandos verwendet.
Achtung: Im Modus „RemainConnected“ darf nur ein einziger Zähler innerhalb des Busses angeschlossen werden !

4.9.3 Konfiguration der Gerätevariablen

Die Konfiguration einer Gerätevariablen wird durch ein XML-Element innerhalb des „Device“ Elements definiert zu dem die Gerätevariable gehört. Folgende XML-Attribute bestimmen die Konfiguration einer Variablen:

```
<External>
  <Bus …>
     <Device …>
        <ValueName  _="MC"  format="Format"/>
        <ValueName  _="ID" sizeID="IDSize" format="Format"/>
        <ValueName  _="OBIS" obis="OBIS" size="Size" indVAL="OBISInd"
          format="Format" />
        <ValueName  _="Text" obis="OBIS" size="Size" indVAL="ValInd"
          subindTXT="TxtSubInd" format="Format" />
        <ValueName  _="Count" obis="OBIS" precision="Precision"
         indVAL="ValInd" subindVAL="ValSubInd" format="Format" />
        <ValueName  _="DWord" obis="OBIS" indVAL=" ValInd "
         subindVAL="ValSubInd" exp="Exp" multip="Factor" format="Format" />
        <ValueName  _="Byte" obis="OBIS" indVAL=" ValInd "
        subindVAL="ValSubInd " exp="Exp" multip="Factor" format="Format" />
        <ValueName  _="OBISLine"  indVAL="LineInd" size="Size"
         format="Format" />
        <ValueName _="Count" ProgCmd="R1" simpleType="Double"
         obis="OBIS" subindVal="0" />
     </Device>
  </Bus>
</External>
```

Jedes Element beschreibt einen Geräteparameterwert, der vom Zähler gelesen werden kann, um z.B. Alarmnachrichten zu generieren oder den Gerätestatus zu loggen.

Es gibt acht Klassen von Gerätevariablen. Eine Klasse fasst alle Geräteparameter zusammen, die gleiche Eigenschaften (Speicherung, Kodierung, Lesen oder Schreiben möglich) besitzen.
Die Klassenzugehörigkeit einer Gerätevariablen ist durch den Wert des Typ-Attributs „_“ (rot hervorgehoben) definiert:

Variablenklasse	Beschreibung

Text	Parameterwert dargestellt durch einen ASCII Text String im OBIS-Format.
	Der Wert einer “Text” Gerätevariable wird im FP IoT Gateway als ein C-String (simpleType=”**String**”) gespeichert. Die „Text“-Parameter besitzen eine maximale Länge, die durch das „size“ Attribut festgelegt wird.
	Das Attribut „**obis**“ kennzeichnet die Zeile im OBIS Format aus, der der Wert gelesen wird.
	Durch das Attribut „**indVAL**“ wird festgelegt welcher Wert (Klammerausdruck) aus der Zeile gelesen werden soll.
	Besteht ein Wert aus mehreren Teilen z.B. Zählerwert und Einheit (sind durch *-Zeichen getrennt) wird mit dem Attribut „subind“ der entsprechende Teil des Wertes adressiert. Ist der „**subindTXT**“ Wert 255 (Standardwert) wird der ganze Klammerausdruck gelesen, ohne Beachtung der Werteteile.
	Beispiel
	Lesen der Geräteidentifikation
	`<DevID_DevID _="Text" obis="1-0:0.0.9*255" indVAL="0" subindTXT="0" size="16" />`
	
	Vom Zähler gesendete Nutzdaten:
	1-0:0.0.9*255(2210200300001000A) ← anhand der OBIS Kennzahl selektierte Zeile
	1-0:1.8.0*255(141002.0000*kWh)
	1-0:96.5.5*255(00) ← Wert der „Text“ Variablen
	
	Beispiel
	Lesen der Zählwerteinheit
	`<Cnt_Unit _="Text" obis="1-0:1.8.0*255" size="3" indVAL="0" subindTXT="1"/>`
	
	Vom Zähler gesendete Nutzdaten:
	1-0:0.0.9*255(221020030000100A)
	1-0:1.8.0*255(141002.0000*kWh) ← anhand der OBIS Kennzahl selektierte Zeile
	1-0:96.5.5*255(00) ← Wert der „Text“ Variablen
	
	Beispiel
	Lesen des gesammten Zählwertes mit Einheit als Text
	`<Cnt_Unit _="Text“ obis="1-0:1.8.0*255" size="3" indVAL="0" subindVAL="255" />`

Variablenklasse	Beschreibung
	Vom Zähler gesendete Nutzdaten: 1-0:0.0.9*255(221020030000100A) 1-0:1.8.0*255(141002.0000*kWh) 1-0:96.5.5*255(00) anhand der OBIS Kennzahl selektierte Zeile Wert der „Text“ Variablen *Beispiel* Lesen des Zählwerte Zeitstempels als Text `<Cnt_DateRAW _="Text" obis="1-0:1.8.0*255" size="10" indVAL="1" subind="255"/>` Vom Zähler gesendete Nutzdaten: 1-0:0.0.9*255(221020030000100A) 1-0:1.8.0*255(141002.0000*kWh)(0504010000) 1-0:96.5.5*255(00) anhand der OBIS Kennzahl selektierte Zeile Wert der „Text“ Variablen

OBIS	OBIS Code dargestellt durch einen ASCII Text String im OBIS-Format.
	Der Wert einer “OBIS” Gerätevariable wird im FP IoT Gateway als ein C-String (simpleType=”**String**”) gespeichert. Die „OBIS“-Parameter besitzen eine maximale Länge, die durch das „size“ Attribut festgelegt wird.
	Das Attribut „**obis**“ kennzeichnet die Zeile im OBIS Format aus, der der Wert gelesen wird.
	Durch das Attribut „**ind**“ wird festgelegt welcher Teil des OBIS Codes gelesen werden soll (Standard ist 0 = kompletter OBIS Code).
	
	Beispiel
	Lesen des ganzen OBIS Codes
	`<DevID_OBIS _="OBIS" obis="1-0:0.0.9*255" indVAL="0" size="16" />`
	
	Vom Zähler gesendete Nutzdaten:
	
	1-0:0.0.9*255(221020030000100A)
	1-0:1.8.0*255(141002.0000*kWh)
	1-0:96.5.5*255(00)
	
	anhand der OBIS Kennzahl selektierte Zeile
	Wert der „OBIS“ Variablen
	
	Beispiel
	Lesen des (optionalen) ersten Teils(Medium)
	`<Cnt_Medium _="OBIS" obis="1-0:1.8.0*255" size="3" indVAL="1"/>`
	
	Vom Zähler gesendete Nutzdaten:
	
	1-0:0.0.9*255(221020030000100A)
	1-0:1.8.0*255(141002.0000*kWh)
	1-0:96.5.5*255(00)
	
	anhand der OBIS Kennzahl selektierte Zeile
	Wert der „OBIS“ Variablen

Variablenklasse	Beschreibung

	Beispiel
	Lesen des mittleren (obligatorischen) Teils
	`<Cnt_OBISID _="OBIS" obis="1-0:1.8.0*255" size="5" indVAL="2" />`
	
	Vom Zähler gesendete Nutzdaten:
	1-0:0.0.9*255(221020030000100A)
	1-0:1.8.0*255(141002.0000*kWh)
	1-0:96.5.5*255(00)
	anhand der OBIS Kennzahl selektierte Zeile
	Wert der „OBIS“ Variablen
	
	Beispiel
	Lesen des dritten(optionalen)Teils (Vorwert)
	`<Cnt_VV _="OBIS" obis="1-0:1.8.0*255" size="3" indVAL="3" />`
	
	Vom Zähler gesendete Nutzdaten:
	1-0:0.0.9*255(221020030000100A)
	1-0:1.8.0*255(141002.0000*kWh)(0504010000)
	1-0:96.5.5*255(00)
	anhand der OBIS Kennzahl selektierte Zeile
	Wert der „OBIS“ Variablen

Count	Zählerwert dargestellt durch eine Festkommazahl im OBIS Format. Der Wert einer “Count” Gerätevariable wird im FP IoT Gateway als eine Double Precision Floatingpoint (simpleType=”**Double**”) Zahl gespeichert. **Hinweis:** Count Variablen sollen zur Speicherung von Zählerständen dienen. Die Zählerstände besitzen bis zu 10 Stellen. Eine solche Zahl lässt sich nicht in einem DWORD Integerwert speichern (maximal 9 Stellen). Double Werte erreichen die geforderte Genauigkeit (maximal 16 Stellen). Da für weitere Berechnungen im FP IoT Gateway nur eine Integer Arithmetik zur Verfügung steht, muss der Double Wert mit dem „**precision**“ Attribut in einen Integerwert umgewandelt werden. Dabei können Genauigkeitsverluste auftreten. Das Attribut „**obis**“ kennzeichnet die Zeile im OBIS Format aus, der der Wert gelesen wird. Durch das Attribut „**indVAL**“ wird festgelegt, welcher Wert (Klammerausdruck) aus der Zeile gelesen werden soll (Standard ist 0). Besteht ein Wert aus mehreren Teilen, z.B. Zählerwert und Einheit, wird mit dem Attribut „**subindVAL**“ der entsprechende Teil des Wertes adressiert. Der Standardwert ist 0, so dass ohne Angabe von „ind“ und „subind“ immer der Zählerwert gelesen wird.

Variablenklasse	Beschreibung
	Beispiel Lesen des Zählerwertes. `<Cnt_Counter _="Count" obis="1-0:1.8.0*255" indVAL="0" subindVAL="0" />` Vom Zähler gesendete Nutzdaten: 1-0:0.0.9*255(221020030000100A) 1-0:1.8.0*255(141002.0000*kWh) 1-0:96.5.5*255(00) anhand der OBIS Kennzahl selektierte Zeile Wert der „Count“-Variablen
Byte	Parameterwert dargestellt durch eine ASCII codierte Integerzahl (maximal 3 Ziffern) im OBIS Format. Der Wert einer “Byte” Gerätevariable wird im FP IoT Gateway als eine Bytezahl (simpleType=”**Uint8**”) gespeichert. Das Attribut „**obis**“ kennzeichnet die Zeile im OBIS Format aus, der der Wert gelesen wird. Durch das Attribut „**indVAL**“ wird festgelegt, welcher Wert (Klammerausdruck) aus der Zeile gelesen werden soll (Standard ist 0). Besteht ein Wert aus mehreren Teilen, z.B. Zählerwert und Einheit, wird mit dem Attribut „**subindVAL**“ der entsprechende Teil des Wertes adressiert. Der Standardwert ist 0, so dass ohne Angabe von „indVAL“ und „subindVAL“ immer der Zahlenwert gelesen wird. *Beispiel* Lesen des Statuswertes als Bytewert. `<State_State _="Byte" obis="1-0:96.5.5*255" />` Vom Zähler gesendete Nutzdaten: 1-0:0.0.9*255(221020030000100A) 1-0:1.8.0*255(141002.0000*kWh) 1-0:96.5.5*255(00) anhand der OBIS Kennzahl selektierte Zeile Wert der „Byte“ Variablen
DWord	Parameterwert dargestellt durch eine ASCII codierte Integerzahl (maximal 10 Ziffern) im OBIS Format. Der Wert einer “Byte” Gerätevariable wird im FP IoT Gateway als eine Double-Word-Zahl (simpleType=”**Uint32**” siehe Kapitel 2.1) **gespeichert.** Das Attribut „**obis**“ kennzeichnet die Zeile im OBIS Format aus, der der Wert gelesen wird. Durch das Attribut „**indVAL**“ wird festgelegt welcher Wert (Klammerausdruck) aus der Zeile gelesen werden soll (Standard ist 0). Besteht ein Wert aus mehreren Teilen, z.B. Zählerwert und Einheit, wird mit dem Attribut „**subindVAL**“ der entsprechende Teil des Wertes adressiert. Der Standardwert ist 0, so dass ohne Angabe von „indVAL“ und „subindVAL“ immer der Zahlenwert gelesen wird.

Variablenklasse	Beschreibung
	Beispiel Lesen des Zählwerte Zeitstempels als Zahlenwert `<Cnt_Date _="Text" obis="1-0:1.8.0*255" indVAL="1" />` Vom Zähler gesendete Nutzdaten: 1-0:0.0.9*255(221020030000100A) 1-0:1.8.0*255(141002.0000*kWh)(0504010000) 1-0:96.5.5*255(00) anhand der OBIS Kennzahl selektierte Zeile Wert der „DWord“ Variablen
OBISLine	Speichert eine Zeile des OBIS Formats als ASCII Text. Diese Variablen Klasse soll es ermöglichen den vom Zähler gesendeten OBIS-Text im Original an ein Abrechnungssystem weiterzuleiten. Die „OBISLine“-Parameter besitzen eine maximale Länge, die durch das „size“ Attribut festgelegt wird. Durch das Attribut „indVAL“ wird die Nummer der Zeile definiert, die gespeichert werden soll. *Beispiel* Lesen der OBIS Zeile mit dem Zählwert `<OBIS_Cnt _="OBISLine" size="40" indVAL="1" />` Vom Zähler gesendete Nutzdaten: 1-0:0.0.9*255(221020030000100A) 1-0:1.8.0*255(141002.0000*kWh) 1-0:96.5.5*255(00) anhand des „ind“ Attributes selektierte Zeile
MC	**Manufacturer Code** dargestellt durch drei ASCII Zeichen im CS-Protokoll. Der Wert einer “MC” Gerätevariable wird im FP IoT Gateway als ein C-String (simpleType=”**String**”) gespeichert. Die Stringlänge ist fix auf **3 Zeichen** eingestellt. Der Wert einer „MC“ Gerätevariablen wird auf folgende Weise aus dem CS-Protokoll Telegramm ermittelt: Startzeichen / Baudrate / Trennung / Ende /XXX5NNNNNNNNNNNNNNN*CRLFCRLF(Nutzdaten)*!*CRLF* Wert der „MC“ Variablen (3 ASCII Zeichen) Identifikation (max. 16 ASCII Zeichen)

Variablenklasse	Beschreibung
ID	**Identification** dargestellt durch maximal 16 ASCII Zeichen im CS-Protokoll. Der Wert einer "ID" Gerätevariable wird im FP IoT Gateway als ein C-String (simpleType="**String**" siehe Kapitel 2.1) **gespeichert.** Die „ID"-Parameter besitzen eine maximale Länge, die durch das „size" Attribut festgelegt wird. Der Wert einer „ID" Gerätevariablen wird auf folgende Weise aus dem CS-Protokoll Telegramm ermittelt: Startzeichen Baudrate Trennung Ende **/XXX5**NNNNNNNNNNNNNNNN*CRLFCRLF(Nutzdaten)!CRLF* Wert der „MC" Variablen (3 ASCII Zeichen) Identifikation (max. 16 ASCII Zeichen)
R1	Spezielles Abfragekommando zum Auslesen einzelner Werte über OBIS-Kennzahl. Beschleunigt bei Auslesung nur weniger Werte die Datenabfrage erheblich. Dafür wird der OpMode="prog" benötigt.

Jede Klasse hat ihr eigenes Set an Attributen.

Beschreibung der (kursiv und fett geschriebenen) Attributwerte:

ValueName Name der Gerätevariablen maximal 30 Zeichen, nur alphanumerische Zeichen, keine Umlaute, darf nicht mit einer Zahl beginnen. Bildet den letzten Teil der Gerätevariablen-Adresse:

Process/*Bus*/*Device*/***VariableName***

Die kursiv geschriebenen Namen werden durch die Prozesskonfiguration definiert. Der fett geschriebene Name wird durch diesen Wert definiert.

Beispiel

```
<Bus Name="CS" … >
   <Device Name="Meter1" … >
      <Cnt_Counter…../>
```

Process/ CS /Meter1/Cnt_Counter

OBISInd optional, Standardwert: 0 Dezimalzahl (0...255)
Kennzeichnet den **Index eines OBIS Kennzahl Teils** in einer OBIS-Format Zeile.

Die Kennzahl hat folgenden Aufbau:

Medium A	-	Kanal B	:	Messgröße C	*	Messart D	*	Tarif E	*	Vorwert F	Daten

Die Felder A+B und F sind optional.
Damit besteht die Kennzahl aus drei Teilen, die mit diesem Attribut adressiert werden können. Der Standardwert 0 liefert die komplette (empfangene) Kennzahl.

Beispiel

Einzelne Teile

1-0:1.8.0*255(141002.0000*kWh)(0504010000)

Wert ind="2"

Wert ind="1"

Die Zählung beginnt mit dem ersten Teil bei 1.

Ganze Kennzahl

1-0:1.8.0*255(141002.0000*kWh)(0504010000)

Wert ind="0"

ValInd optional, Standardwert: 0 Dezimalzahl (0...255)

Kennzeichnet den **Index eines Werts** (Klammerausdruck) in einer OBIS-Format Zeile.

Beispiel

1-0:1.8.0*255(141002.0000*kWh)(0504010000)

OBIS Kennzahl — Wert ind="0" — Wert ind="1"

Die Zählung beginnt mit dem ersten Klammerausdruck bei 0.

ValSubInd optional, Standardwert: 0 Dezimalzahl (0...255)

Kennzeichnet den **Index eines Werteteils** (innerhalb eines Klammerausdrucks) in einer OBIS-Format Zeile. Ein Wert kann aus mehreren Teilen bestehen, die durch das „*"-Zeichen oder durch ein Leerzeichen getrennt sind. Diese Teile können mit diesem Index einzeln adressiert werden.

Beispiel

Wert subind="0"

1-0:1.8.0*255(141002.0000*kWh)(0504010000)

OBIS Kennzahl — Wert subind="0" — Wert subind="1"

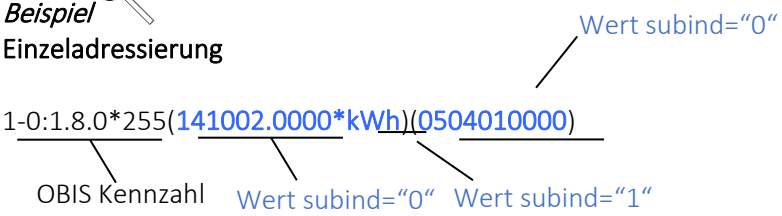
Die Zählung beginnt mit dem ersten Teil eines Wertes (Klammerausdruck) bei 0.

TxtSubInd optional, Standardwert: 255 Dezimalzahl (0...255)

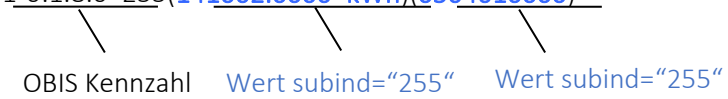
Kennzeichnet den **Index eines Werteteils** (innerhalb eines Klammerausdrucks) in einer OBIS-Format Zeile oder den **gesamten Klammerausdruck**. Ein Wert kann aus mehreren Teilen bestehen, die durch das „*"-Zeichen oder durch ein Leerzeichen getrennt sind. Diese Teile können mit diesem Index einzeln adressiert oder durch den Index **255 zusammengefasst** werden.

Beispiel

Einzeladressierung



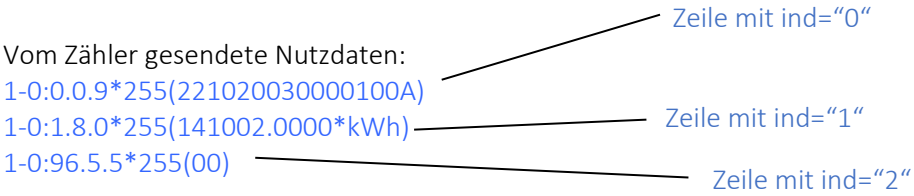
Gesamtadressierung



Die Zählung beginnt mit dem ersten Teil eines Wertes (Klammerausdruck) bei 0.

LineInd optional, Standardwert: 0, Zahlenwert 0...255
Bedeutung nur für Variablen der Klasse **OBISLine.** Beschreibt den Index der Zeile in den Nutzdaten des CS-Protokolls.

Beispiel



Die Zählung beginnt mit der ersten Zeile in den Nutzdaten bei 0.

Size erforderlich, 0...100
Bedeutung nur für Variablen der Klasse **Text.** Beschreibt die maximale Anzahl der ASCII Zeichen im OBIS-Format.

IDSize Optional, Standardwert ist „16", 1...16
Bedeutung nur für Variablen der Klasse **ID.** Beschreibt die maximale Anzahl der ASCII Zeichen im CS-Protokoll.

OBIS erforderlich, OBIS-Kennzahl Suchmaske
OBIS Kennzahl zur Adressierung der Zeile im OBIS-Format.

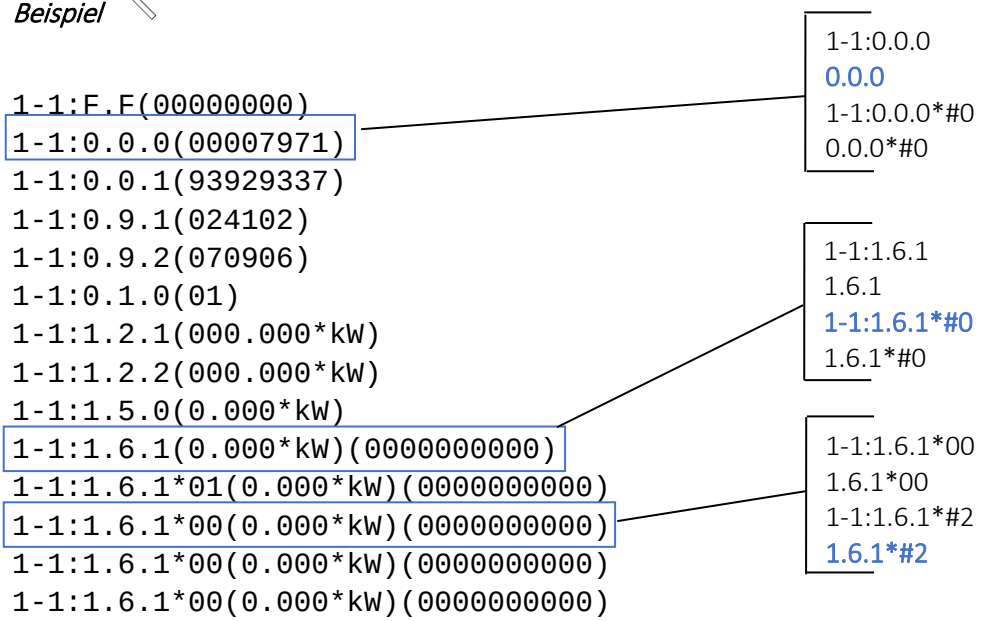
Die Kennzahl hat folgenden Aufbau:

Medium A	-	Kanal B	:	Messgröße C	.	Messart D	.	Tarif E	*	Vorwert F	Daten

Die Felder A+B und F sind optional.
Zur Ermittlung der referenzierten Zeile wird die vom Zähler gesendete Kennzahl mit der im „obis" Attribut angegeben Kennzahl verglichen. Beide Kennzahlen sind gleich, wenn sie in den Feldern C,D und E übereinstimmen. Es kann weiterhin durch Angabe weiterer Felder die Auswahl präzisiert werden.
Wird kein Vorwert (Sektion F) angegeben, so wird der erste Wert gefunden, der mit dem restlichen Angaben korreliert.

Mit **#0** als Vorwert wird der erste Wert gefunden, der keinen Vorwert besitzt, jedoch bei dem die restlichen Angaben korrelieren.
Mit ***#n,*** wobei ***n*** eine Zahl zwischen 1 und 255 ist, findet den Wert mit dem n-ten Vorwert.

Beispiel



Links: Datenausgabe des Zählers. Die gefundenen Zeilen sind umrahmt.
Rechts: OBIS-Kennzahl Suchmaske. Die besten Suchmasken sind fett geschrieben.

Format Optional, Standartwert ist „“ (keine Formatierung).
Die Formatoptionen kennzeichnen die Standardformatierung des Wertes, wie sie bei Ausgaben z.B. in E-Mails oder beim Get Befehl (siehe "TiXML Reference Manual") erfolgt.

Factor optional, Standardwert: 1/1, Format: siehe unten
Der vom Zähler empfangene Wert (valueDevice) wird mit diesem Faktor multipliziert :

valueTDG = Factor * valueDevice
valueDevice = 1/Factor * valueTDG

Der Faktor wird als Bruch geschrieben, z.B.: „1/1000“ oder „3600/1“, Zähler und Nenner müssen positive Ganzzahlen sein und dürfen nicht 0 sein.

Exp optional, Standardwert: 0, erlaubte Werte: siehe untenstehende Tabelle
Exponent der Basis 10, der die Genauigkeit einer Festkommazahl definiert.
Der im FP IoT Gateway gespeicherte Wert wird mit 10 exp(*Exp*) (nach Anwendung des Faktors), zum Variablenwert multipliziert .

valueVariable = 10^{Exp} * valueGateway.

Der Exponent definiert damit die Position des Kommas in der Festkommazahl.

Folgende Werte sind möglich:

Exp Wert	Beschreibung
-6	Genauigkeit = 0,000001
-5	Genauigkeit = 0,00001
-4	Genauigkeit = 0,0001
-3	Genauigkeit = 0,001
-2	Genauigkeit = 0,01
-1	Genauigkeit = 0,1
0	Genauigkeit = 1 (Standard)
1	Genauigkeit = 10
2	Genauigkeit = 100
3	Genauigkeit = 1000
4	Genauigkeit = 10000
5	Genauigkeit = 100000
6	Genauigkeit = 1000000

Precision optional, Standradwert:0, gültige Werte: siehe folgende Tabelle

Der im FP IoT Gateway gespeicherte Wert wird mit 10 exp(*Exp*) multipliziert, um den Wert in die Integerdarstellung zu wandeln. Die Integerdarstellung wird bei der Berechnung der Prozessvariablen z.B. mit den Befehlen (GT, LT etc.) verwendet. Er gibt somit die Genauigkeit bei der Berechnung der Prozessvariablen an und ist Abhängig von der Anwendung zu definieren.

Integerdarstellung = Ganzzahl(10^{Exp} * valueParameter).
Die Werte des entsprechen der folgenden Tabelle.

Precision Wert	Beschreibung
-6	Faktor = 0,000001
-5	Faktor = 0,00001
-4	Faktor = 0,0001
-3	Faktor = 0,001
-2	Faktor = 0,01
-1	Faktor = 0,1
0	Faktor = 1 (Standard)
1	Faktor = 10
2	Faktor = 100
3	Faktor = 1000
4	Faktor = 10000
5	Faktor = 100000
6	Faktor = 1000000

4.9.4 Fehlerwert einer Variablen

Jede konfigurierte Gerätevariable hat einen zugehörigen Fehlerzustandswert.
Der Fehlerzustandswert enthält einen zweigliedrigen Fehlercode, der den genauen Fehlerzustand und die Fehlerquelle beschreibt. Der Fehlerzustandswert kann verwendet werden, um Konfigurationsfehler aufzudecken oder um Alarme oder Logeinträge zu erzeugen, wenn ein Kommunikationsfehler entsteht.

Sollte eine Variable nicht gelesen werden können, weil Kommunikations- oder Anwendungsfehler auftreten, wird der Fehlerzustandswert mit dem entsprechenden Fehlercode versehen und der Variablenwert auf „**undefiniert**“ gesetzt.

Um den Fehlerzustandswert einer Variablen zu lesen, wird eine Erweiterung des „Get“ Kommandos verwendet:

Befehl:
`[<Get _="VPath" AddInfo="AddInfo" />]`

Übersicht der möglichen Parameter (kursiv geschriebene Werte):

Vpath: Pfad zur Adressierung des Parameters.

AddInfo:
Error ..... Gibt Fehlerzustand des Wertes zurück.

AddInfo wird bei Werten, die keine Parameter von externen Geräten sind, ignoriert.
Es wird der Wert des Parameters zurückgegeben.

Antwort:
`[<Get _="ErrorClass,ErrorValue" />]`

ErrorClass: Fehler Klasse
0 Kein Fehler
1 von FP IoT Gateway erkannte Fehler

ErrorValue: Fehlerwert

ErrorClass	ErrorValue	Bedeutung
0	0	Kein Fehler
1	2	Wertezeile ist nicht im Telegramm enthalten. Das tritt ein, wenn: a) Keine Zeile mit der in der Konfiguration einer Gerätevariablen definierten OBIS - Kennzahl (siehe Attribut „**obis**“) gefunden wurde. oder b) Wenn keine Zeile in dem in der Konfiguration einer Gerätevariablen definierten Zeilenindex (siehe Attribut „**ind**“ bei der Variablenklasse „OBISLine“) empfangen wurde

ErrorClass	ErrorValue	Bedeutung
1	3	Wert ist nicht im Telegramm enthalten. Das tritt ein, wenn zwar die Zeile gefunden wurde, jedoch der per Attribut „**ind**“ und „**subind**“ adressierte Wert (siehe Konfiguration der Gerätevariablen) nicht gefunden wurde oder nicht gewandelt werden konnte.
1	4	Ungültiges Telegram. Es wurde ein Telegramm empfangen, das nicht die erwartete Struktur hat oder dessen Block Control Code falsch ist.
1	5	Timeout, es wurde kein vollständiges Telegramm nach der letzten Anfrage empfangen.

Beispiel für die Verwendung des R1-Befehls

```
<Device Name="Device_0" NameUser="MT174" _="1" OpMode="prog"
  Options="RemainConnected" Pollrate="1s" Address="12345678"
  PreSend="0" PreDelay="180">

    <Variable_0 Name="Uhrzeit" _="Text" size="6" ProgCmd="R1"
     obis="0.9.1" acc="R"/>

    <Obis_000 Name="Nummer" _="Text" ProgCmd="R1" simpleType="String"
     obis="0.0.0" size="8" acc="R"/>

    <Obis_092 Name="Datum" _="Text" ProgCmd="R1" simpleType="Uint32"
    obis="0.9.2" size="12" subindVAL="0" acc="R"/>

    <Obis_091 Name="Uhrzeit" _="DWord" ProgCmd="R1"
    simpleType="Uint32" obis="0.9.1" subindVAL="0" acc="R"/>

    <Obis_170 _="Count" ProgCmd="R1" simpleType="Double" obis="1.7.0"
    subindVAL="0" acc="R"/>

    <Obis_180 _="Count" ProgCmd="R1" simpleType="Double" obis="1.8.0"
    subindVAL="0" acc="R"/>

    <Obis_280 _="Count" ProgCmd="R1" simpleType="Double" obis="2.8.0"
    subindVAL="0" acc="R"/>

    <Obis_270 _="Count" ProgCmd="R1" simpleType="Double" obis="2.7.0"
    subindVAL="0" acc="R"/>

</Device>
```

4.10 D0-Protokoll (EN 62056-21 Mode D)

FP IoT Gateways unterstützen das CS-Protokoll nach EN 62056-21 Mode D.

Dieses Protokoll wird häufig bei Stromzählern verwendet (eHZ = Elektronischer Haushaltszähler).

Die Bus-Definition und die Variablen-Definition sind fast vollständig identisch zum CS-Protokoll (siehe Kapitel 4.9), deshalb werden hier nur die Unterschiede dokumentiert.

External Gruppe der 'PROCCFG' Datenbank:

```
<External>
   <Bus _="COM2" Name="BusName" protocol="DIN1107,D0" type="Master"
    baud="Speed" handshake="HALF" format="DataFormat"
    BusPollrate="BusPollrate TUnit">
```

Unterschiede

- Protokoll "DIN1107,D0"
- es kann nur jeweils ein Zähler pro serielle Schnittstelle angeschlossen werden (kein Bus!)

4.11 SML-Protokoll

FP IoT Gateways der 8.Generation (z.B. HE852) unterstützen das SML-Protokoll für die LMN Schnittstelle für moderne Messeinrichtungen. Die LMN-Schnittstelle arbeitet mit einer Baudrate von 921600 bps.
Die Variablen sind in der External Gruppe der 'PROCCFG' Datenbank registriert.

Verwende Schnittstelle COM2

```
<External>
   <Bus _="COM2" Name="BusName" product="LMN" family="LMN"
    protocol="LMNProt" type="Master" baud="Speed" handshake="HALF"
    format="DataFormat">
      <Device _="1" Name="Alias " MS-ID="MS_ID" Pollrate="1s">
         <Obis_180 _="Count" obis="1.8.0" simpleType="Double" acc="R" />
         <Obis_280 _="Count" obis="2.8.0" simpleType="Double" acc="R" />
         <Obis_3270 _="Count" obis="32.7.0" simpleType="Double" acc="R" />
      </Device>
   </Bus>
</External>
```

Die Zähler werden an eine RS485-Schnittstelle an das Tixi G8-Gerät angeschlossen.
Alle rot dargestellten XML-Attribute sind erforderliche Attribute mit konstanten Werten.

4.11.1 Bus-Konfiguration

Beschreibung der konstanten (rot gekennzeichneten) Attribute:

protocol Definiert das Busprotokoll: "LMNProt"

type Definiert die Rolle des FP IoT Gateways im Bus als "Master"

handshake Bestimmt die Art der Flusskontrolle zwischen FP IoT Gateway und Zähler. Bei Nutzung des 2-Draht RS485 Busses muss eine Halbduplex-Verbindung konfiguriert werden, daher "HALF"

COM erforderlich
Schnittstelle des FP IoT Gateways, an dem der Bus angeschlossen wird.
Werden 2 Busse definiert, müssen sich die Schnittstellen unterscheiden.

Wert	Beschreibung
COM1	RS 485 Schnittstelle auf COM1
COM*X*	Weitere RS485-Schnittstelle (falls im FP IoT Gateway vorhanden) *X* = Nummer der Schnittstelle, z.B. 2

Übersicht der möglichen Parameter (kursiv geschriebene Werte):

BusName optional, Standard ist: "" (leer). Maximal 20 alphanumerische Zeichen, darf nicht mit einer Zahl beginnen.

Definiert den Namen der TiXML-Attributgruppe, die den Bus in der "Process"-Attributgruppe repräsentiert. Der Gruppenname ist der erste Teil einer Gerätevariablen-Adresse:

Process/***Bus***/*Device/Variable*

Die kursiv geschriebenen Namen werden durch die Prozesskonfiguration definiert. Der fett geschriebene Name wird durch dieses Attribut definiert.

Wert	Beschreibung
"" (leer)	(Standardwert) Auto-Name. Die Attributgruppe bekommt einen automatischen Namen zugewiesen: Process/Busn/... n automatisch generierte Indexnummer
BusName (nicht leer)	Die Attributgruppe erhält den angegebenen Namen: Process/MeinName/... Beispiel: <Bus Name="LMN" ...> Process/LMN/...

Speed optional, fester Wert: 921600

DataFormat fester Wert: 8N1
Bestimmt das Datenformat auf der seriellen Schnittstelle.
Syntax: DataBitsParityBitsStopBits

DataBits: 8 = 8 Datenbits; 7 = 7 Datenbits
ParityBits: N = kein Paritätsbit; E = gerade Parität; O = ungerade Parität
StopBits: 1 = 1 Stopbit; 2 = 2 Stopbits

Handshake fester Wert bei RS485-Schnittstellen: HALF

4.11.2 Device-Konfiguration

Die Gerätekonfiguration wird durch ein "Device" XML Element innerhalb des Bus Elements definiert und durch folgende XML-Attribute beschrieben:

```
<External>
   <Bus ...>
      <Device _="ID" Name="Alias" MS-ID="MS_ID" PollRate="Rate
       TUnit" >
               <<< Konfiguration der Geräte-Variablen >>>
      </Device>
   </Bus>
</External>
```

Für jeden Zähler, der an der Schnittstelle (COM1 oder COM2) angeschlossen ist und mit dem das FP IoT Gateway kommunizieren soll, muss ein "Devcice"-Eintrag angelegt werden.

Beschreibung der *(kursiv und fett geschriebenen)* Attributwerte*:*

ID erforderlich, Zahlen, 0 ..255

Bus-ID des Zählers.
Sie dient der Unterscheidung, falls mehrere Devices konfiguriert werden.

MS_ID erforderlich, Messstellen-Nummer, die auf den Zählern aufgedruckt ist. Ziffern, 8-stellig Kommunikationsadresse des Zählers.

Alias optional, Standardwert: “” (leer), max. 20 alphanumerische Zeichen, darf nicht mit einer Zahl beginnen.

Definiert den Namen der TiXML Attributgruppe, die das Gerät in der „Bus“ Attributgruppe repräsentiert. Der Gruppenname ist der zweite Teil einer Gerätevariablen- adresse:

Process/*Bus*/***Device***/*Variable*

Die kursiv geschriebenen Namen werden durch die Prozesskonfiguration definiert. Der fett geschriebene Name wird durch dieses Attribut definiert.

Wert	Beschreibung
“” (leer)	(Standartwert) Auto-Name. Die Attributgruppe erhält einen automatisch generierten Namen zugewiesen: Process/*BusName/*D*Id…* *Id* Wert des Attributs “_”
DeviceName (nicht leer)	Die Attributgruppe erhält den angegebenen Namen: Process/*Bus/DeviceName…* Beispiel: <Bus Name=”LMN” ….> <Device Name=”Meter1” ….> Process/LMN/Meter1/…

Rate optional, Standard: „15m“, gültige Werte, siehe unten
Pollrate in Zeiteinheiten (siehe TUnit), mit der das FP IoT Gateway das Gerät abfragt.
Wertebereiche siehe "BusPollrate"

TUnit Zeiteinheit (ms, s, m, h)

4.11.3 Konfiguration der Gerätevariablen

Die Konfiguration einer Gerätevariablen wird durch ein XML-Element innerhalb des „Device“ Elements definiert zu dem die Gerätevariable gehört. Folgende XML-Attribute bestimmen die Konfiguration einer Variablen:

```
<External>
  <Bus …>
    <Device ...>
      <ValueName  _="Bool"   obis="OBIS" format="Format" />
      <ValueName  _="Count"  obis="OBIS" format="Format" exp="Exp"
                             multip="Factor" />
      <ValueName  _="SCount" obis="OBIS" format="Format" exp="Exp"
                             multip="Factor"/>
      <ValueName  _="ASCII"  size="Size" obis="OBIS" format="Format" />
    </Device>
  </Bus>
</External>
```

Jedes Element beschreibt einen Geräteparameterwert, der vom Zähler gelesen werden kann, um z.B. Alarmnachrichten zu generieren oder den Gerätestatus zu loggen.

Es gibt vier Klassen von Gerätevariablen. Eine Klasse fasst alle Geräteparameter zusammen, die gleiche Eigenschaften (Speicherung, Kodierung, Lesen oder Schreiben möglich) besitzen.

Die Klassenzugehörigkeit einer Gerätevariablen ist durch den Wert des Typ-Attributs „_“ (rot hervorgehoben) definiert:

Variablenklasse	Beschreibung
Bool	Parameterwert dargestellt durch einen Boolean-Wert Der Wert einer “Bool” Gerätevariable wird im FP IoT Gateway als ein Bit (simpleType=”**Bit**”) gespeichert. Das Attribut „**obis**“ kennzeichnet die OBIS-Kennzahl. *Beispiel* **Lesen der Einstellung „Ausgabe eines hersteller-spezifischen Datensatzes auf der INFO-Schnittstelle“** `<Fbz_0078 _="Bool" obis="94.49.1*2" />`
Count	Zählerwert dargestellt durch eine Festkommazahl im OBIS Format. Der Wert einer “Count” Gerätevariable wird im FP IoT Gateway als eine vorzeichenlose Double Precision Floatingpoint (simpleType=”**Double**”) Zahl gespeichert. **Hinweis:** Count Variablen sollen zur Speicherung von Zählerständen dienen. Die Zählerstände besitzen bis zu 10 Stellen. Eine solche Zahl lässt sich nicht in einem DWORD Integerwert speichern (maximal 9 Stellen). Double Werte erreichen die geforderte Genauigkeit (maximal 16 Stellen). Da für weitere Berechnungen im FP IoT Gateway nur eine Integer Arithmetik zur Verfügung steht, muss der Double Wert mit dem „**precision**“ Attribut in einen Integerwert umgewandelt

Variablenklasse	Beschreibung
	werden. Dabei können Genauigkeitsverluste auftreten. Das Attribut „**obis**“ kennzeichnet die OBIS Kennzahl. *Beispiel* **Lesen des Zählerstands zur Wirkarbeit in Richtung +A** `<Fbz_0098 _="Count" obis="1.8.0*255" />`
SCount	Zählerwert dargestellt durch eine Festkommazahl im OBIS Format. Der Wert einer “Count” Gerätevariable wird im FP IoT Gateway als eine vorzeichenbehaftete Double Precision Floatingpoint (simpleType=”**Double**”) Zahl gespeichert. **Hinweis:** Count Variablen sollen zur Speicherung von Zählerständen dienen. Die Zählerstände besitzen bis zu 10 Stellen. Eine solche Zahl lässt sich nicht in einem DWORD Integerwert speichern (maximal 9 Stellen). Double Werte erreichen die geforderte Genauigkeit (maximal 16 Stellen). Da für weitere Berechnungen im FP IoT Gateway nur eine Integer Arithmetik zur Verfügung steht, muss der Double Wert mit dem „**precision**“ Attribut in einen Integerwert umgewandelt werden. Dabei können Genauigkeitsverluste auftreten. Das Attribut „**obis**“ kennzeichnet die OBIS-Kennzahl. *Beispiel* **Lesen der momentanen Wirkleistung** `<Fbz_0103 _="SCount" obis="1.7.0*255" />`
ASCII	Parameterwert dargestellt durch einen ASCII String. Das Attribut „**size**“ legt die Länge der Zeichenkette in Bytes fest. Das Attribut „**obis**“ kennzeichnet die OBIS-Kennzahl. *Beispiel* **Lesen der Hersteller-Kennung** `<Fbz_0084 _="ASCII" size="20" obis="96.50.1*1" />`

Jede Klasse hat ihr eigenes Set an Attributen.

Beschreibung der (kursiv und fett geschriebenen) Attributwerte:

`ValueName` Name der Gerätevariablen maximal 30 Zeichen, nur alphanumerische Zeichen, keine Umlaute, darf nicht mit einer Zahl beginnen. Bildet den letzten Teil der Gerätevariablen-Adresse:

Process/*Bus*/*Device*/***VariableName***
Die kursiv geschriebenen Namen werden durch die Prozesskonfiguration definiert.

Der fett geschriebene Name wird durch diesen Wert definiert.

Beispiel

```
<Bus Name="LMN" … >
   <Device Name="Meter1" … >
      <Cnt_Counter…../>
```

Process/LMN/Meter1/Cnt_Counter

OBIS erforderlich, OBIS-Kennzahl
OBIS Kennzahl zur Adressierung des Datenpunktes im verkürzten OBIS-Format.

Die Kennzahl hat folgenden Aufbau:

Die Felder A+B müssen weggelassen werden und werden intern automatisch auf „1-0“ gesetzt. Das Feld F ist optional.
Wird das Feld F weggelassen, wird es intern durch 255 (=FF) ersetzt.

Size erforderlich, 0...100
Bedeutung nur für Variablen der Klasse **ASCII.** Beschreibt die maximale Anzahl der ASCII Zeichen im OBIS-Format.

Format Optional, Standartwert ist „“ (keine Formatierung).
Die Formatoptionen kennzeichnen die Standardformatierung des Wertes, wie sie bei Ausgaben z.B. in E-Mails oder beim Get Befehl (siehe "TiXML Reference Manual") erfolgt.

Factor optional, Standardwert: 1/1, Format: siehe unten
Der vom Zähler empfangene Wert (valueDevice) wird mit diesem Faktor multipliziert :

valueTDG = Factor * valueDevice
valueDevice = 1/Factor * valueTDG

Der Faktor wird als Bruch geschrieben, z.B.: „1/1000“ oder „3600/1“, Zähler und Nenner müssen positive Ganzzahlen sein und dürfen nicht 0 sein.

Exp optional, Standardwert: 0, erlaubte Werte: siehe untenstehende Tabelle
Exponent der Basis 10, der die Genauigkeit einer Festkommazahl definiert.
Der im FP IoT Gateway gespeicherte Wert wird mit 10 exp(*Exp*) (nach Anwendung des Faktors), zum Variablenwert multipliziert .

valueVariable = 10^{Exp} * valueGateway.

Der Exponent definiert damit die Position des Kommas in der Festkommazahl.

Folgende Werte sind möglich:

Exp Wert	Beschreibung
-6	Genauigkeit = 0,000001
-5	Genauigkeit = 0,00001
-4	Genauigkeit = 0,0001
-3	Genauigkeit = 0,001
-2	Genauigkeit = 0,01
-1	Genauigkeit = 0,1

Exp Wert	Beschreibung
0	Genauigkeit = 1 (Standard)
1	Genauigkeit = 10
2	Genauigkeit = 100
3	Genauigkeit = 1000
4	Genauigkeit = 10000
5	Genauigkeit = 100000
6	Genauigkeit = 1000000

Precision optional, Standardwert:0, gültige Werte: siehe folgende Tabelle

Der im FP IoT Gateway gespeicherte Wert wird mit 10 exp(*Exp*) multipliziert, um den Wert in die Integerdarstellung zu wandeln. Die Integerdarstellung wird bei der Berechnung der Prozessvariablen z.B. mit den Befehlen (GT, LT etc.) verwendet. Er gibt somit die Genauigkeit bei der Berechnung der Prozessvariablen an und ist abhängig von der Anwendung zu definieren.

Integerdarstellung = Ganzzahl(10^{Exp} * valueParameter).
Die Werte des entsprechen der folgenden Tabelle.

Precision Wert	Beschreibung
-6	Faktor = 0,000001
-5	Faktor = 0,00001
-4	Faktor = 0,0001
-3	Faktor = 0,001
-2	Faktor = 0,01
-1	Faktor = 0,1
0	Faktor = 1 (Standard)
1	Faktor = 10
2	Faktor = 100
3	Faktor = 1000
4	Faktor = 10000
5	Faktor = 100000
6	Faktor = 1000000

Beispielkonfiguration

```
[<SetConfig _="PROCCFG" ver="y">
<External>

<Bus Name="Bus1" _="COM2" family="LMN" Product="LMN" protocol="LMNProt"
     Mem="120000" baud="921600" handshake="HALF" type="Master" format="8N1"
     AddProperties="Name,TimeStamp">

  <Device Name="MT176" _="1" MS-ID="68575884" Pollrate="5s">
    <Obis_180 _="Count" obis="1.8.0" acc="R" def="0" format=";%% kWh" />
    <Obis_280 _="Count" obis="2.8.0" acc="R" def="" format=";%% kWh" />
    <Obis_1670 _="SCount" obis="16.7.0" acc="R" def="0" format=";%% kWh" />
    <Obis_3270 _="Count" obis="32.7.0" acc="R" def="0" format=";%% V" />
    <Obis_944912 _="Bool" obis="94.49.1*2" acc="R" def="0"/>
    <Obis_965011 _="ASCII" size="10" obis="96.50.1*1" acc="R" def="0"/>
    <Obis_944901 _="Count" obis="94.49.0*1" acc="R" exp="-3" def="0"/>
  </Device>

  <Device Name="MT631" _="2" MS-ID="68956562" Pollrate="2s">
    <Obis_180 _="Count" obis="1.8.0" acc="R" def="0" format=";%% kWh" />
    <Obis_280 _="Count" obis="2.8.0" acc="R" def="0" format=";%% kWh" />
    <Obis_1670 _="SCount" obis="16.7.0" acc="R" def="-1" format=";%% kWh"/>
```

```
    <Obis_3270 _="Count" obis="32.7.0" acc="R" def="0" format=";%% V" />
    <Obis_944912 _="Bool" obis="94.49.1*2" acc="R" def="0"/>
    <Obis_965011 _="ASCII" size="5" obis="96.50.1*1" acc="R" def="0"/>
    <Obis_944901 _="Count" obis="94.49.0*1" acc="R" multip="1/10" def="0"/>
    <Obis_944911 _="Bool" obis="94.49.1*1" acc="RW" def="0"/>
  </Device>

</Bus>

</External>
</SetConfig>]
```

Release-Notes der aktuellen Beta-Version

Folgende Hinweise gelten für die Beta-Version v6.0.0.26 (Stand: 15.11.2019):

- Folgende OBIS-Kennzahlen wurden für den MT176 erfolgreich getestet:
  94.49.0*1, 94.49.1.*1, 94.49.1.*2, 94.49.1.*3, 94.49.1.*4, 94.49.1.*9, 94.49.1.*10
  96.50.1*1, 1.8.0*255, 1.8.1*255, 32.7.0*255, 52.7.0*255, 72.7.0*255, 16.7.0*255
- Folgende OBIS-Kennzahlen wurden für den MT631 erfolgreich getestet:
  94.49.0*1, 94.49.1.*1, 94.49.1.*2, 94.49.1.*3, 94.49.1.*9, 94.49.1.*10
  96.50.1*1, 1.8.0*255, 1.8.1*255, 32.7.0*255, 52.7.0*255, 72.7.0*255, 16.7.0*255
- Aufgrund eines gravierenden Fehlers der beiden Zählertypen erfolgt die Abfrage von Parametern (OBIS-Kennzahlen) einzeln je Parameter, da die Zähler bei der schnellen Abfrage von mehreren Parametern in kurzer Zeit in einen Zustand kommen, in dem eine Kommunikation über die LMN-Schnittstelle nicht mehr möglich ist. Dies hat erhebliche Auswirkungen auf die Abfragerate (je mehr Parameter abgefragt werden, desto länger dauert die gesamte Abfrage)
- Die Firmware erlaubt das Einspielen neuer Abfrage-Parameter (OBIS-Kennzahlen) über einen SetConfig-Befehl. Die Dokumentation dieser Schnittstelle wird noch genauer dokumentiert.

4.12 1-Wire

Der 1-Wire Bus ermöglicht den Anschluss von bis zu 30 Sensoren, family=10 oder family=28.
Es ist empfohlen, extern gespeiste Sensoren zu nutzen (Sensoren mit 3 Anschlüssen: GND, VDD, Data).
Der parasitär versorgte Betrieb wird nicht empfohlen, da der Bus in dieser Betriebsart bei Buslängen über 10m nicht mehr zuverlässig funktioniert.

Die Konfiguration erfolgt über die "External"-Datenbank:

```
[<SetConfig _="PROCCFG" ver="v">
<External>

  <Bus Name="Bus0" _="1Wire" protocol="1Wire" type="master"
   StrongPullup="enable">

    <Device Name="Device_0" _="0" family="28" serial="000000fbbb1d">
      <Temperature_0 Name="Temp_0" _="DW" simpleType="Int32"/>
    </Device>
    <Device Name="Device_1" _="1" family="28" serial="000000fc8b96">
      <Temperature_1 Name="Temp_1" _="DW" simpleType="Int32"/>
    </Device>

  </Bus>
</External>
</SetConfig>]
```

Als Busprotokoll `"1Wire"`, `type="master"` angeben.
Die einzelnen Sensoren werden über Device-Einträge konfiguriert. Dabei muss die Familie (`family=""`)und die Seriennummer (`serial=""`)angegeben werden.

Hinweis:
Die meisten 1-Wire Sensoren geben die Temperatur in 1/1000 °C aus. Ab Firmware Version 5.2.1.6 kann über den Parameter **`StrongPullup`** die Stromversorgung parasitär gesteuerter Sensoren gesteuert werden (`enable` = ein, `disable` = aus).
Im parasitären Betrieb sollte **`StrongPullup`** immer auf `enable` gesetzt werden. Damit kann die Zuverlässigkeit der Kommunikation mit den Sensoren verbessert werden.
Im gespeisten Modus hat der Parameter keine Auswirkungen.

4.12.1 Scannen des 1-Wire Busses

Um zu ermitteln, welche Sensoren am Bus erkannt wurden, kann der TiXML-Befehl `ScanDevices` verwendet werden:

```
[<ScanDevices _="1Wire" protocol="1Wire"/>]
```

Der Befehl gibt alle erkannten Sensoren in einer Listenansicht zurück:

```
<ScanDevices>
  <Device Family="10" Serial="000802bdfa08" Value="11240" ExtPower="1" />
  <Device Family="28" Serial="00000556b8d5" Value="15629" ExtPower="1" />
</ScanDevices>
```

Ab Firmware Version 5.2.1.6 zeigt das `ScanDevices` Kommando den Temperaturmesswert des Sensors (`Value`) sowie die Betriebsart (`ExtPower` 0 = parasitär, 1 = gespeist) an.

Die von `ScanDevices` zurückgegebenen Werte können dann in die External übernommen werden.

```
 [<SetConfig _="PROCCFG" ver="v">
<External>
  <Bus Name="Bus0" _="1Wire" protocol="1Wire" type="master">
    <Device Name="Device_0" _="0" family="10" serial="000802bdfa08">
      <Temperature_0 Name="Temp_0" _="DW" simpleType="Int32"/>
    </Device>
    <Device Name="Device_1" _="1" family="28" serial="00000556b8d5">
      <Temperature_1 Name="Temp_1" _="DW" simpleType="Int32"/>
    </Device>
  </Bus>
</External>
</SetConfig>]
```

Beispiel (Process-Zweig mit einem konfigurierten 1-Wire-Fühler)

```
<Device_0>
  <DeviceState _="1" />
  <ChangeToggle _="0" />
  <Temp_0 _="23456" />
  <ExternalPower _="1" />
</Device_0>
```

Der Parameter **`ExternalPower`** (0 = parasitär, 1 = gespeist) wird ab FW Version 5.2.1.6 angezeigt.

4.13 Aurora-Protokoll für ABB Wechselrichter

Das Aurora-Protokoll ist ein Kommunikationsprotokoll für ABB-Wechselrichter.
Es handelt sich um ein einfaches serielles Protokoll, welches die RS485-Schnittstelle nutzt.
Zur Zeit wird eine Teilmenge des Protokolls im „Normal mode“ unterstützt.

Liste der unterstützten Aurora-Befehle (Normal mode)

Funktion	Beschreibung
CMD50	Request the state of the system modules
CMD58	Version Reading
CMD59	Request measurement to the DSP, Typ 3 "Grid Power (W)”
CMD63	Serial Number Reading
CMD78	Cumulated energy readings

Die Konfiguration erfolgt über die "External"-Datenbank und ist spezifisch für jeden Aurora-Befehl.

Busdefinition

```
<Bus Name="Bus1" _="COM2" family="ABB" Product="Aurora"
     protocol="Aurora,Normal" Mem="120000" baud="19200" handshake="HALF"
     type="Master" format="8N1">
```

4.13.1 CMD50 (”Request state of the system modules“)

Das Kommando 50 ermöglicht das Abfragen von Informationen der System-Module des Wechselrichters.
Bei dieser Funktion werden mehrere Parameter gelesen:

```
Global state, Inverter state, DC/DC Channel 1 state,
DC/DC Channel 2 state, Alarm state
```

Die Auswahl der Parameter in der Variablendefinition erfolgt über den Index (`ind`).

Variablendefinition

```
<GlobState _="CMD50" ind="ByteIndex" simpleType="Uint8" acc="R"/>
```

`CMD50` = Kommando 50 zum Auslesen von Stati der System-Module

```
ByteIndex = 0 | 1 | 2 | 3 | 4
0 = Global state, 1 = Inverter state, 2 = DC/DC Channel 1 state,
3 = DC/DC Channel 2 state, 4 = Alarm state
```

Die vom Kommando zurückgelieferten Werte haben je nach ByteIndex unterschiedliche Bedeutung.
Näheres dazu finden Sie in der Aurora Protokollbeschreibung.

Auszug aus „UAP_00855_AuroraCommunicationProtocol_5_1_6_PUBLIC.pdf“:

Global State	Beschreibung
0	Sending parameters
1	Wait Sun/Grid
2	Checking Grid
3	Wait Sun/Grid
4	Checking Grid
5	Wait Sun/Grid
6	Checking Grid
7	Wait Sun/Grid
8	Checking Grid
9	Wait Sun/Grid
10	Checking Grid
…	…
151	Global Setting Not Valid
198	System Configuration Error
200	DSP Programming

DcDc State	Beschreibung
0	DcDc Off
1	Ramp Start
2	MPPT
3	Not used
4	Input OC
5	Input UV
6	Input OV
7	Input Low
8	No Parameters
9	Bulk OV
10	Communication Error
11	Ramp Fail
12	Internal Error
…	…
255	DcDc Dsp not programmed

Inverter State	Beschreibung
0	Stand By
1	Checking Grid
2	Run
3	Bulk OV
4	Out OC
5	IGBT Sat
6	Bulk UV
7	Degauss Error
8	No Parameters
9	Bulk Low
10	Grid OV
46	Grid Fail
47	Input OC
…	…
255	Inverter Dsp not programmed

Alarm State	Beschreibung	Code
0	No alarm	
1	Sun low	W001
2	Input OC	E001
3	Input UV	W002
4	Input OV	E002
5	Sun low	W001
6	No parameters (DSP)	E003
7	Bulk OV	E004
8	Internal error	E005
9	Output OC	E006
10	IGBT Sat	E007
…	…	
157	System frozen	W058
158	Output power OL	W059
159	Wrong wiring	E089

Beispiel *(Auslesen aller Stati vom Wechselrichter über CMD50)*

```
<GlobState      _="CMD50" ind="0" simpleType="Uint8" acc="R"/>
<InvState       _="CMD50" ind="1" simpleType="Uint8" acc="R"/>
<DCDCChn1State  _="CMD50" ind="2" simpleType="Uint8" acc="R"/>
<DCDCChn2State  _="CMD50" ind="3" simpleType="Uint8" acc="R"/>
<AlarmState     _="CMD50" ind="4" simpleType="Uint8" acc="R"/>
```

Abfrage der Process-Daten

```
<Get _="/Process/Aurora/ABB_1/GlobState" ver="y" />
<Get _="/Process/Aurora/ABB_1/InvState" ver="y" />
<Get _="/Process/Aurora/ABB_1/DCDCChn1State" ver="y" />
<Get _="/Process/Aurora/ABB_1/DCDCChn2State" ver="y" />
<Get _="/Process/Aurora/ABB_1/AlarmState" ver="y" />
```

Ergebnis

```
<GlobState _="1" />
<InvState _="0" />
<DCDCChn1State _="0" />
<DCDCChn2State _="7" />
<AlarmState _="0" />
```

4.13.2 CMD58 (“Version Reading“)

Bei dieser Funktion werden mehrere Parameter gelesen:
`Model identifier, actual supported grid-standard, transformer, type`

Die Auswahl der Parameter in der Variablendefinition erfolgt über den Index (`ind`).

Variablendefinition

```
<Model _="CMD58" ind="ByteIndex" simpleType="String" size="1" acc="R"/>
```

`CMD58 =` Kommando 58 zum Auslesen von Versionsinformationen

```
ByteIndex = 1 | 2 | 3 | 4
 1 = model, 2 = grid, 3 = transformer, 4=type
```

Die vom Kommando zurückgelieferten Werte haben je nach ByteIndex unterschiedliche Bedeutung. Näheres dazu finden Sie in der Aurora Protokollbeschreibung.

Auszug aus „UAP_00855_AuroraCommunicationProtocol_5_1_6_PUBLIC.pdf“:

Model (Auszug)

Model char	Model code	Indoor/Outdoor Type
'i'	105	PVI-2000
'o'	111	PVI-2000-OUTD
'I'	73	PVI-3600
'O'	79	PVI-3600-OUTD
'5'	53	PVI-5000-OUTD
'6'	54	PVI-6000-OUTD
'P'	80	3-phase-interface 3G74
'C'	67	PVI-CENTRAL-50 module
'4'	52	PVI-4.2-OUTD
'3'	51	PVI-3.6-OUTD
'2'	50	PVI-3.8-OUTD
…	…	
'9'	57	UNO-3.6-TL-OUTD
	236	UNO-3.8-TL-OUTD
'x'	120	UNO-4.2-TL-OUTD

Grid (Auszug)

Grid char	Grid code	Grid Standard
'A'	65	USA – UL1741
'E'	69	Germany – VDE0126
'S'	83	Spain – DR 1663/2000
'I'	73	Italy – ENEL DK 5950
'U'	85	UK – UK G83
'K'	75	Australia – AS 4777
'F'	70	France – VDE French Model
'R'	82	Ireland – EN50438
'B'	66	Belgium – VDE Belgium model
'O'	79	Korea
'G'	71	Greece – VDE Greece model
…	…	…
'!'	33	Hawaii – 208 V Single Ph.
'”'	34	Hawaii – 240 V Split Ph.
'#'	35	Hawaii – 277 V Single Ph.

Transformer

Transformer	Beschreibung
'N'	Transformerless version
'T'	Transformer version
't'	Transformer HF version
'x'	Dummy transformer type

Type

Transformer	Beschreibung
'N'	Photovoltaic version
'W'	Eolic version
'x'	Dummy inverter type
'n'	Photovoltaic version + ESS (energy storage system)
'w'	Eolic version + ESS (energy storage system)

Beispiel *(Auslesen aller Versionsinformationen vom Wechselrichter über CMD58)*

```
<Model        _="CMD58" ind="1" simpleType="String" size="1" acc="R"/>
<Grid         _="CMD58" ind="2" simpleType="String" size="1" acc="R"/>
<Transformer  _="CMD58" ind="3" simpleType="String" size="1" acc="R"/>
<Type         _="CMD58" ind="4" simpleType="String" size="1" acc="R"/>
```

Abfrage der Process-Daten

```
<Get _="/Process/Aurora/ABB_1/Model" ver="y" />
<Get _="/Process/Aurora/ABB_1/Grid" ver="y" />
<Get _="/Process/Aurora/ABB_1/Transformer" ver="y" />
<Get _="/Process/Aurora/ABB_1/Type" ver="y" />
```

Ergebnis

```
<Model _="4" />       -> PVI-4.2-OUTD
<Grid _="E" />        -> Germany - VDE0126
<Transformer _="T"    -> transformer version
<Type _="N" />        -> photovoltaic version
```

4.13.3 CMD59 (“Request Measurement to the DSP“)

Das Kommando 59 ermöglicht das Auslesen von DSP-Messwerten. Über einen type-Parameter kann eine Vielzahl von DSP-Werten ausgelesen werden.

Variablendefinition
```
<DSPMeas _="CMD59" type="DSPType" global="1" simpleType="float" acc="R"/>
```

`CMD59` = Kommando 59 zum Auslesen von Messwerten

`DSPType = 1 .. 203` (Bedeutung der type-Variablen siehe Aurora-Spezifikation)

Die vom Kommando zurückgelieferten Werte haben je nach DSPType unterschiedliche Bedeutung. Näheres dazu finden Sie in der Aurora Protokollbeschreibung.

Auszug aus „UAP_00855_AuroraCommunicationProtocol_5_1_6_PUBLIC.pdf“:

DSPType	Messwert	Unterstützte Geräte	Bemerkungen
1	Grid Voltage (V)	Alle	Wenn 3-Phasig = Mittelwert
2	Grid Current (A)	Alle	Wenn 3-Phasig = Mittelwert
3	Grid Power (W)	Alle	Wenn 3-Phasig = Mittelwert
4	Frequency (Hz)	Alle	Wenn 3-Phasig = Mittelwert
…	…	…	…
201	Self Consumption Total	REACT	New architecture ABB inverter
202	Self Sufficiency Today	REACT	New architecture ABB inverter
203	Self Sufficiency Total	REACT	New architecture ABB inverter

Achtung:
Die Wechselrichtertypen "PVI-CENTRAL-350 Liquid Cooled AC GATHERING" werden nicht unterstützt.

Beispiel *(Auslesen der GridPower vom Wechselrichter über CMD59)*
```
<GridPower _="CMD59" type="3" global="1" simpleType="float" acc="R"/>
```

Abfrage der Process-Daten
```
<Get _="/Process/Aurora/ABB_1/GridPower" ver="y" />
```

Ergebnis (Wert in [W])
```
<GridPower _="159.23" />
```

4.13.4 CMD63 (“Serial Number Reading“)

Das Kommando 63 liest die Seriennummer des Wechselrichters aus.

Variablendefinition
```
<SerialNumber _="CMD63" simpleType="String" size="6" acc="R"/>

CMD63 = Kommando 63 zum Auslesen der Seriennummer
```

Abfrage der Process-Daten
```
<Get _="/Process/Aurora/ABB_1/SerialNumber" ver="y" />
```

Ergebnis
```
<SerialNumber _="123456" />
```

4.13.5 CMD78 (“Cumulated Energy Readings“)

Das Kommando 78 liest die kumulierten Energiewerte in [Wh] verschiedener Perioden aus.

Variablendefinition

```
<CumEnergyRead _="CMD78" type="Type" global="1" simpleType="Uint32" acc="R"/>
```

`CMD78 =` Kommando 78 zum Auslesen von Messwerten

`Type = 0 .. 204` (Bedeutung der Type-Variablen siehe Aurora-Spezifikation)

Die vom Kommando zurückgelieferten Werte haben je nach Type unterschiedliche Bedeutung. Näheres dazu finden Sie in der Aurora Protokollbeschreibung.

Auszug aus „UAP_00855_AuroraCommunicationProtocol_5_1_6_PUBLIC.pdf“:

Slot 0:

Type	Beschreibung
0	Daily Energy
1	Weekly Energy
2	Not used
3	Monthly Energy
4	Yearly Energy
5	Total Energy (total lifetime)
6	Partial Energy (cumulated since last reset)

Slot 1:

Type	Beschreibung
100	Daily Energy
101	Weekly Energy
102	Not used
103	Monthly Energy
104	Yearly Energy
105	Total Energy (total lifetime)
106	Partial Energy (cumulated since last reset)

Slot 2:

Type	Beschreibung
107	Daily Energy
108	Weekly Energy
109	Not used
110	Monthly Energy
111	Yearly Energy
112	Total Energy (total lifetime)
113	Partial Energy (cumulated since last reset)

Slot 3: Type = 114 - 120

Slot 4: Type = 121 – 127

Slot 5: Type = 121 – 127

. . .

Slot 14: Type = 191 - 197

Slot 15: Type = 198 - 204

Beispiel: Variablendefinition (alle Variablen für Slot 0)

```
<DailyEnergy    _="CMD78" type="0" global="1" simpleType="Uint32" acc="R"/>
<WeeklyEnergy   _="CMD78" type="1" global="1" simpleType="Uint32" acc="R"/>
<MonthlyEnergy  _="CMD78" type="3" global="1" simpleType="Uint32" acc="R"/>
<YearlyEnergy   _="CMD78" type="4" global="1" simpleType="Uint32" acc="R"/>
<TotalEnergy    _="CMD78" type="5" global="1" simpleType="Uint32" acc="R"/>

CMD63 = Kommando 63 zum Auslesen der Seriennummer
```

Abfrage der Process-Daten (Slot 0)

```
<Get _="/Process/Aurora/ABB_1/DailyEnergy" ver="y" />
<Get _="/Process/Aurora/ABB_1/WeeklyEnergy" ver="y" />
<Get _="/Process/Aurora/ABB_1/MonthlyEnergy" ver="y" />
<Get _="/Process/Aurora/ABB_1/YearlyEnergy" ver="y" />
<Get _="/Process/Aurora/ABB_1/TotalEnergy" ver="y" />
```

Ergebnis (alle Werte in [Wh])

```
<DailyEnergy _="1821" />
<WeeklyEnergy _="5290" />
<MonthlyEnergy _="19467" />
<YearlyEnergy _="252178" />
<TotalEnergy _="1325678" />
```

5 Formatieren von SPS Variablenwerten

Ohne Formatierung werden die Variablen so dargestellt, wie sie von der SPS übermittelt werden. Das FP IoT Gateway kann diese Werte in Zahlenformate ändern und boolsche Variablenwerte durch Zeichenketten ersetzen. Die Formatierte Variable wird in E-Mails eingebunden und bei Get-Befehlen (ohne eigenes Formatattribut) ausgegeben.

Formatieren von SPS-Variablenwerten	
Syntax	*Bei der Variablendefinition:* **<Variable ...simpleType="Uint8" exp="2"...** **format="*Elements;Text*"/>** *Bei der Variablenwertabfrage:* **<Get ... format="*Elements;Text*"/>** *Beim Setzen des Variablenwertes:* **<Set ... format="*Elements*"/>**
Beschreibung:	Der Parameter `format` besteht aus zwei Teilen, die durch ein Semikolon getrennt sind: 1.Teil: Enthält Format-Elemente, die die Aus- oder Eingabe von Zahlen beschreiben. Abgesehen vom Tausendertrennzeichen „T“ und dem Zahlenformat „F“ sind die Formatelemente nicht miteinander kombinierbar. Die Position des Tausendertennzeichenelementes im Format-Befehl ist beliebig. Die Formatanwendung hängt vom verwendeten Typ der Variablen ab. Nicht alle Formatierungen sind für alle Typen gleichermaßen geeignet. Für die Eignung eines Formatierungselements ist der im Attribut „simpleType“ der Variablendefinition angegebene Basistyp der Variablen ausschlaggebend. Daher werden hier für jedes Formatelement die dafür geeigneten Basistypen angegeben. Der erste Teil kann auch leer sein. Dann werden der Variablenwerte in in seinem nativen Format ausgegeben. 2.Teil: Enthält einen Text der zusammen mit dem Wert der Variablen ausgegeben wird. Der Variablenwert kann innerhalb dieses Textes in dem durch den ersten Teil definierten Format ausgeben werden. Dazu wird die Position des Variablenwertes durch einen Platzhalter (%%) dargestellt. Für bestimmte Variablen können noch weitere, zugehörige Variablenwerte (z.B. die phyikaische Größe und die Einheit) in den Text durch Platzhalter eingesetzt werden. Der zweite Teil kann auch weggelassen werden. Dann braucht auch kein Semikolon vorangestellt werden. *Beispiel* Beide Teile “T’F+9,2 ;Radius %% cm“ Nur 1.Teil “R16“ Nur 2. Teil “Text mit:%% als Wert“

Formatieren von SPS-Variablenwerten	
Format Elemente (Teil 1):	
? – logische Alternative	**?string1,string2** Dieser Befehl ersetzt die beiden Werte boolscher Variablen durch Zeichenketten. Wenn die Variable ungleich 0 ist, wird string1 ausgegeben, andernfalls string2. *Anwendbar für folgende simpleType Werte:* Uint8, Uint16, Uint32, Int8, Int16, Int32 mit exp= "0"und Bit
Beispiel 	**<Variable _="F" simpleType="Uint8" exp="0" … format="?open,closed"/>** <Get _="/Process/COM2/D1/Variable"/> FP IoT Gateway antwortet: <Get _="open"/> bei Wert 1
* - Auswahl Alternative	***Value1:Text1*Value2:Text2**:Text3** Dieser Befehl wird verwendet um Variablenwerte durch vordefinierte Zeichenketten zu ersetzen. Wenn der Wert gleich Value1 ist wird Text1 ausgegeben, wenn der Wert gleich Value2 ist, wird Text2 ausgegeben, in allen anderen Fällen Text3. * Trennzeichen für zu erkennenden Wert ** Trennzeichen für alle anderen Werte Die Anzahl der Werte ist unbegrenzt. *Anwendbar für folgende simpleType Werte:* Uint8, Uint16, Uint32, Int8, Int16, Int32 und exp= "0"
Beispiel 	**<Variable _="R" simpleType="Uint8" exp="0" … format="*0:low*1:medium*2:high**:faulty"/>** <Get _="/Process/COM2/D1/Variable"/> FP IoT Gateway antwortet: <Get _="low"/> bei Wert 0 <Get _="medium"/> bei Wert 1 <Get _="high"/> bei Wert 2 <Get _="faulty"/> bei Wert 7
R/r - Basis	**Rn/rn** Der Befehl definiert die Basis n des auszugebenden Wertes. n = 2 Binärausgabe (z.B. 01101010) n = 8 Oktalausgabe (z.B. 21057) n = 10 Dezimalausgabe (Standard, z.B. 1234) n = 16 Hexadezimalausgabe (z.B. AE03) Mit der Groß- oder Kleinschreibung wird die Ausgabe von Buchstaben bei der Hexadezimaldarstellung gesteuert: R Es werden Großbuchstaben versendet (z.B. AE03) r Es werden Kleinbuchstaben verwendet (z.B. ae03) *Anwendbar für folgende simpleType Werte:* Uint8, Uint16, Uint32, Int8, Int16, Int32 und exp= "0"

Formatieren von SPS-Variablenwerten	
Beispiel 	Wert in Großbuchstaben: **<Variable _="R" simpleType="Uint8" exp="0" ... format="R16"/>** <Get _="/Process/COM2/D1/Variable"/> FP IoT Gateway antwortet (Variablenwert=90): <Get _="5A"/> Wert in Kleinbuchstaben: **<Variable _="R" simpleType="Uint8" exp="0"... format="r16"/>** <Get _="/Process/COM2/D1/Variable"/> FP IoT Gateway antwortet (Variablenwert=90): <Get _="5a"/>
T - Tausendertrennzeichen	**T*n*** Definiert das Trennzeichen, welches an jeder tausender Stelle der Ausgabe erscheint. n= , Komma als Tausendertrennzeichen (z.B. 12,345,678) n= . Punkt als Tausendertrennzeichen (z.B. 12.345.678) n= ` Hochkomma als Tausendertrennzeichen (z.B. 12`345`678) n= leer Kein Tausendertrennzeichen (Standard) Hinweis: Mit Formatelement "F" kombinierbar, kann aber auch einzeln verwendet oder weggelassen werden. *Anwendbar für folgende simpleType Werte:* Uint8, Uint16, Uint32, Int8, Int16, Int32, Float, Double
Beispiel 	**<Variable _="R" simpleType="Uint32" exp="0"... format="T. "/>** <Get _="/Process/COM2/D1/Variable"/> FP IoT Gateway antwortet (Variablenwert=98765): <Get _="98.765"/>
F - Zahlenformat	**F Vorzeichen Leerstellen Feldbreite Dezimalzeichen Dezimalstellen** Dieser Befehl definiert das Format einer Zahl. Er enthält verschiedene Elemente, welche in dieser Reihenfolge anzugeben sind: Vorzeichen: Legt fest, ob ein Vorzeichen ausgegeben werden soll + Das Vorzeichen wird immer ausgegeben (z.B. "+12.3" , "-12.3") - Es wird nur ein negatives Vorzeichen ausgegeben (z.B. "12.3" , "-12.3") Leerstellen: Legt fest, wie Leerstellen aufgefüllt werden(nur bei Verwendung von Feldbreite) 0 Leere Stellen werden mit 0 aufgefüllt (z.B. 0066.3) leer Leere Stellen werden werden nicht aufgefüllt (z.B. 66.3)

Formatieren von SPS-Variablenwerten

Feldbreite: Gibt die maximale Feldgröße **inklusive** Vorzeichen und Trennzeichen an. Ohne diese Angabe ist das Feld unbegrenzt, und es werden keine Leerzeichen aufgefüllt.
Geben Sie hier immer ausreichend große Werte an, sonst wird die ausgegebene Zahl links abgeschnitten.

Dezimalzeichen: Dieses Zeichen wird als Dezimaltrenner verwendet (optional)

,	ein Komma als Trennzeichen
.	ein Punkt als Trennzeichen (Default)

Dezimalstellen: Bestimmt die Anzahl der Nachkommastellen.
Kann weggelassen werden, wenn kein Dezimalzeichen angegeben wurde.

Hinweis:
Mit Formatelement "T" kombinierbar, kann aber auch einzeln verwendet oder weggelassen werden.

Anwendbar für folgende simpleType Werte:
Uint8, Uint16, Uint32, Int8, Int16, Int32, Float, Double

Beispiel

Der Wert der Variable ist in allen Beispielen "12345":

Vorzeichen:

```
<Variable _="F" simpleType="Float"… format="F+"/>

<Get _="/Process/COM2/D1/Variable"/>
```

FP IoT Gateway antwortet (Wert = 123,45):

```
<Get _="+123.45"/>
```

Feldbreite, Leerstellen:

```
<Variable _="R" simpleType="Uint32" exp="-3"...
format="F09"/>

<Get _="/Process/COM2/D1/Variable"/>
```

FP IoT Gateway antwortet (Wert = 123,456):

```
<Get _="00123.456"/>
```

Feststellenzahl , Dezimalzeichen, Dezimalstellen, Länge :

```
<Variable ="R" simpleType="Int32" exp="-3"…
format="T'F+9,2"/>

<Get _="/Process/COM2/D1/Variable"/>
```

FP IoT Gateway antwortet (Wert = 3123,456):

```
<Get _="+3'123,45"/>

<Variable_="R" simpleType="Int32" exp="0"… format="T'F+9.2"/>

<Get _="/Process/COM2/D1/Variable"/>
```

FP IoT Gateway antwortet (Wert = -3123456):

Formatieren von SPS-Variablenwerten	

	`<Get _="-312'345"/>`
	
	Gleitkommazahl, Dezimalzeichen, Dezimalstellen, Länge:
	`<Variable _="F" simpleType="Float"… format="T'F+9.2"/>`
	
	`<Get _="/Process/COM2/D1/Variable"/>`
	FP IoT Gateway antwortet (Wert = 3123,456):
	`<Get _="+3'123.45"/>`

Text (Teil 2): **%%**	Dieser Platzhalter kennzeichnet die Position des Variablenwertes im Ausgabetext. Dieser Teil geht für alle Datentypen. Für „String" ist er die einzige Formatierungoption.

Beispiel	**`<Variable _="R" simpleType="Int32" exp="-2" … format="F+;Temp: %%°C"/>`**
	
	`<Get _="/Process/COM2/D1/Variable"/>`
	FP IoT Gateway antwortet (wert ist 123,45):
	`<Get _="Temp: +123.45°C"/>`

%M% – M-BUS Medium (VIF) **%U% – M-BUS Einheit (VIF)**	Diese Platzhalter geben die im M-BUS Value Information Field übergebenen Daten für Medium und Einheit aus. Weitere M-Bus Platzhalter aus den Data Information Fields (DIF) (ab FW 5.1.6.8): **%N% – Storage number (DIF)** **%T% – Tariff (DIF)** **%S% – Subunit (DIF)** **%F% – Function field (DIF)** Um diese DIF-Platzhalter nutzen zu können, müssen in der LOG-Definition die betreffenden Variablen mit einer Größe von size="**17**" konfiguriert werden: `<D_XX_V_1 _="meterbus" path="/Process/MBus/D_XX/D_XX_V_1" size="17" format=";%%,%U%,%N%,%T%,%S%"/>`

Beispiel	`<Var01` **`simpleType="Int32" exp="-2"... format=";Medium:%M% Wert=%% %U%"`**`/>`
	
	`<Get _="/Process/COM3/D1/Var01"/>`
	FP IoT Gateway antwortet (Variablenwert=25,30, Wärmezähler Volumen):
	`<Get _=" Medium:Heat 0 Volume Flow Wert=25.30 l/h"/>`

6 Verwenden der SPS-Variablen im FP IoT Gateway

Die SPS-Variablen können im FP IoT Gateway genauso verwendet werden, wie die FP IoT Gateway eigenen Ein- und Ausgänge, z.B. um Alarme auszulösen, Daten zu loggen oder für Fernwirken.

6.1 Standardadressierung

Nach dem Definieren der SPS-Variablen im FP IoT Gateway können diese über den Prozesspfad angesprochen werden:

Beispiel (RS232-2, Station 0, Variable Alarm11):

```
<L _="&#xae;/Process/COMx/D0/Alarm11"/>
```

Die Variablen sind im Pfad der Erweiterungskarte (z.B. COM1, COM2, COM3) und der Stationsnummer (Device ID) der SPS (z.B. D0, D1, D2,…) aufgeführt und können dort über Ihren Aliasnamen angesprochen werden.

6.2 Adressierung über Busname und Stationsname

Durch die Verwendung von Busnamen und Stationsnamen können die Variablen unabhängig von der Schnittstelle und Stationsnummer im System adressiert werden. Dadurch kann die SPS-Schnittstelle oder Stationsnummer an zentraler Stelle geändert werden, ohne das gesammte Projekt ändern zu müssen.
Der COM-Port kann dabei ab Firmware 1.80 entsprechend der Beschriftung mit COM1 bis COM3 definiert werden.

Beispiel

```
<External>
   <Bus _="COM2" Name="MyPLC" protocol="Mitsubishi,Alpha2"
type="Master"
    baud="9600">
      <Device _="0" Name="AlphaXL" Pollrate="1s">
         <Input1 _="I" ind="1"/>
      </Device>
   </Bus>
</External>
```

Der Input1 kann im System nun wie folgt angesprochen werden:

```
<L _="&#xae;/Process/MyPLC/AlphaXL/Alarm11"/>
```

Diese Adressierung wäre auch dann noch gültig, wenn in der SPS-Definition die Schnittstelle "COM2" und/oder die DeviceID "0" geändert würde.

6.3 Überwachung der SPS-Kommunikation

Für jedes parametrierte Gerät werden automatisch Systemvariablen eingefügt.

DeviceState:

```
[<Get _="/Process/COM?/D?/DeviceState"/>]
```

Diese Variable zeigt den aktuellen Zustand der Kommunikation an:
SPS-Antwortet: DeviceState=1
keine Antwort: DeviceState=0

ChangeToggle:

```
[<Get _="/Process/COM?/D?/ChangeToggle"/>]
```

Wenn das Gerät bei einem Pollzyklus Wertänderungen in der SPS erkannt hat, wechselt diese Bitvariable ihren Zustand.

Beide Variablen können somit auch in den EventStates z.B. als Alarm- oder Logtrigger verwendet werden.

Active:

```
[<Get _="/Process/COM?/D?/Active"/>]
```

Über diese schreibbare Variable lässt sich die Kommunikation zur SPS unterbrechen:
`[<Set _="/Process/COM?/D?/Active" value="0"/>]` stoppt die Kommunikation.
`[<Set _="/Process/COM?/D?/Active" value="1"/>]` startet die Kommunikation (default).

Index