Thomas Boschen

Zusi 3 TCP-Client

Bibliothek für CODESYS V3

Thomas Boschen

11.5.2018

INHALT

1. EINLEITUNG

1.1. WAS IST LIBZUSI?

Die Zusi 3 TCP-Client Bibliothek für CODESYS V3 (im Folgenden kurz *libzusi*) enthält alle notwendigen Funktionen um ein Fahrpult unter Verwendung einer CODESYS-Steuerung an den Zusi 3 TCP-Server anzuschließen. Die Bibliothek wurde auf der Plattform „CODESYS Control for Raspberry Pi SL“ entwickelt und getestet, ist jedoch theoretisch auf allen CODESYS V3-Steuerungen lauffähig. Die Funktion unter Twincat 3 ist bisher nicht getestet.

Libzusi ist in erster Linie als Baukastensystem zur Erstellung eines Fahrpult-Clients zu verstehen. Es gibt bereits eine Anzahl fertiger Funktionsbausteine für gängige Führerstandsfunktionen wie Fahr- und Bremsschalter, Sifa, Türen, PZB etc. Ein zentraler, leicht konfigurierbarer Baustein verwaltet die Verbindung zu Zusi. Mit Hilfe der Demo-Projekte lässt sich ohne großen Programmieraufwand die passende Lösung für die individuelle Hardware anpassen und das Bidirektional: es können sowohl Befehle an Zusi gesendet, also auch Meldungen empfangen werden um damit z.B. Leuchtmelder oder Drehspulinstrumente anzusteuern.

1.2. WAS LIBZUSI NICHT IST

Es bietet keine „lauffertige“ Lösung für alle Anwendungen. Libzusi ist geeignet für Anwender mit geringer Programmiererfahrung. Sich zumindest oberflächlich mit dem Programmiersystem CODESYS auseinander zu setzen ist leider unumgänglich. Speziellere Anforderungen, wenn die Standart-Bausteine nicht mehr ausreichen, setzen dann eine Vertiefung der Kenntnisse voraus. Dafür ist libzusi aber quelloffen, dem Erstellen eigener Funktionen steht also nichts im Wege.

2. RECHTLICHES UND KONTAKT

Libzusi ist ein reines Hobby-Projekt, von daher gibt es keinen rechtlichen Anspruch auf technische Unterstützung. Diese wird, sofern möglich, auf freiwilliger Basis geleistet. Erste Anlaufstelle bei Fragen und Problemen ist das Zusi-Forum.
Alternativer Kontakt per E-Mail: libzusi@tbxtools.de.

Des Weiteren gilt zu beachten dass aufgrund des Open-Source Status des Projektes keinerlei Gewährleistungsansprüche geltend gemacht werden können und jede Haftung seitens des Autors ausgeschlossen ist. **Jegliche Nutzung von libzusi erfolgt auf eigenes Risiko.**

3. EINBINDUNG UND MODIFIKATION

Das aktuellste Paket der Bibliothek inklusive Beispielprojekte gibt es hier:

http://www.tbxtools.de/codesys/zusi3tcp/

Installiert wird libzusi mit dem Package Manager. Der Speicherort für die Beispielprojekte kann während der Installation festgelegt werden.

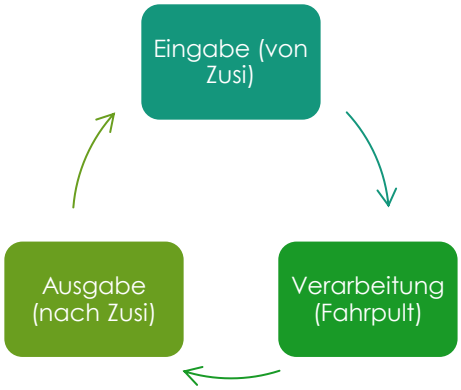
Der komplette Quellcode der Bibliothek kann ebenfalls kostenlos heruntergeladen werden:

http://www.tbxtools.de/codesys/zusi3tcp/lib/

4. ARCHITEKTUR

Libzusi wird als Bibliothek in ein CODESYS V3-Projekt eingebunden. Welche Hardware dabei zum Einsatz kommt ist zunächst unerheblich. Die verwendete Steuerung wird über Ethernet mit dem Zusi-Rechner verbunden. Libzusi empfängt die angeforderten Daten und legt sie in einer eigenen Struktur im dynamischen Speicher der Steuerung ab, wo die einzelnen Werte dann abgefragt werden können.

Abfrage und Verarbeitung der empfangenen Daten erfolgt SPS-typisch nach dem EVA-Prinzip (Eingabe, Verarbeitung, Ausgabe): Bei Taktbeginn werden die empfangenen Daten aufbereitet, dann erfolgen die Fahrpult-spezifischen Verarbeitungen und schließlich werden die gesammelten Eingaben an Zusi zurückgesendet. Dann beginnt der Zyklus erneut.



4.1. DATENEMPFANG

In Zusi 3 kann der Sendeinterval für ausgehende TCP-Daten in Millisekunden eingestellt werden. Voreingestellt ist eine Zeit von 1 Sekunde. Für Fahrpultanwendungen empfiehlt es sich, diese Zeit auf Ca. 100 ms herunterzustellen. Zusätzlich versendet Zusi per Voreinstellung nur Werte die sich geändert haben. Diese Voreinstellung sollte beibehalten werden.

Wie „flüssig“ die Daten im Fahrpultprogramm also empfangen werden hängt somit vom Sendeinterval auf der Zusi-Seite, sowie vom Umfang des Programmes, der Anzahl der angeforderten Daten und der eingestellten mindest-Taktzeit des Tasks ab.

4.2. DATENVERSENDUNG

Befehle an Zusi werden immer zum schnellstmöglichen Zeitpunkt versendet. Während eines Zyklus anfallende Eingabe-Daten der einzelnen Funktionsbausteine werden gesammelt und bei erneutem Aufruf des FB_ZUSITCP versendet (somit einmal pro Zyklus). Wie schnell dies geschieht ist primär vom Programmumfang und der eingestellten Taktzeit abhängig. Üblicherweise liegt die Taktzeit bei kleineren bis mittleren Programmen bei wenigen Millisekunden.

5. WICHTIGER HINWEIS FÜR NEUE PROJEKTE

Beim Anlegen eines neuen Projektes unter Einbindung von libzusi muss in den Applikations-Einstellungen unbedingt die **dynamische Speicherassoziierung** eingeschaltet und eine ausreichend große Menge Speicher eingestellt werden (typischerweise >1000 Bytes).

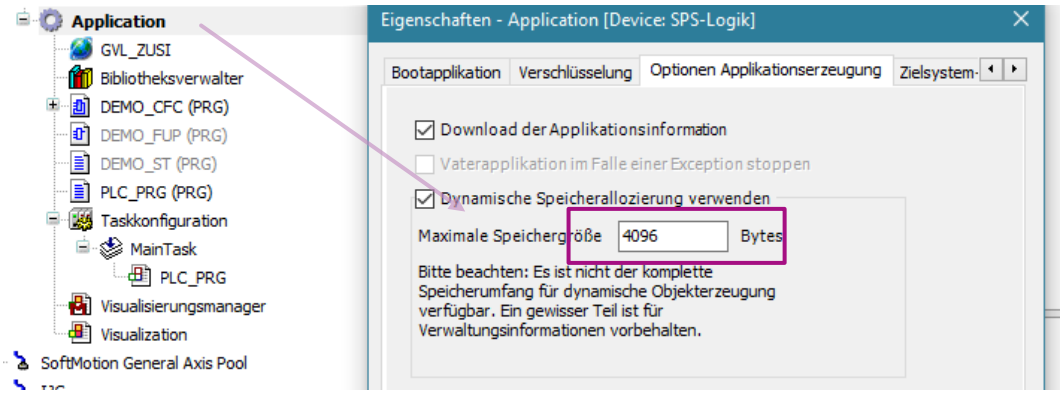


Abbildung 1 Dynamischer Speicher

Das Fehlen dieser Einstellung führt ansonsten beim Kompilieren zu einem Fehler.

6. ERSTE SCHRITTE

Ein guter Einstieg ist das Beispiel-Projekt „libzusi-demo-basics“. Die Demo braucht keine Ein- oder Ausgabehardware und wird über eine Visualisierung bedient. Das Fahrpult-Programm liegt in drei verschiedenen Varianten vor, um zu zeigen, dass die Funktionen Programmiersprachen-Unabhängig sind. Welche Programmiersprache verwendet wird, liegt alleine bei den Präferenzen des Anwenders.

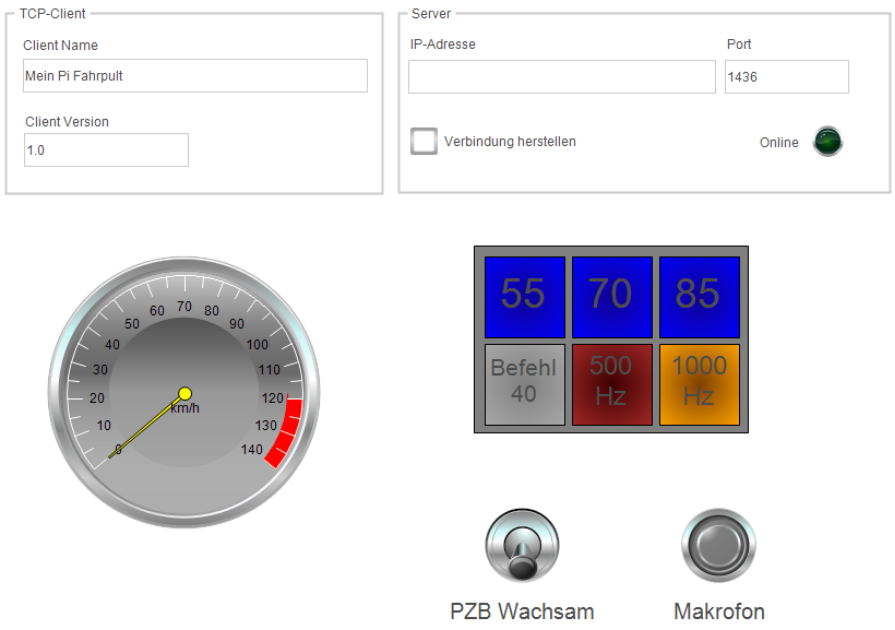


Abbildung 2 Visualisierung

[Video-Tutorial folgt]

7. AUFBAU DER BIBLIOTHEK

Libzusi ist in mehrere Unterordner unterteilt. Im Bibliotheksverwalter kann die detaillierte Dokumentation inklusive Anschluss-Beschreibung eingesehen werden, deshalb hier nur ein grober Überblick:

7.1. FUNKTIONEN

Dieser Ordner enthält Funktionen zur Werte-Abfrage und Umrechnungen.

FUN_ANALOG

Funktion zur einfachen Umrechnung eines Analogkanal-Rohwertes in einen Faktor 0.00 bis 1.00, untere und obere Grenze der Rohwerte müssen definiert sein.

FUN_CABDATA

Abfrage eines einzelnen Wertes im Single (REAL)-Format aus dem Befehlsvorrat, sofern vorher mit NEEDED_DATA angefragt.

FUN_NEEDED_DATA

Hiermit wird beim Login-Vorgang Zusi 3 mitgeteilt welche Daten benötigt werden. Siehe Zusi-Doku Kapitel *11.3.3.3.1 Führerstands-IDs*

FUN_RASTE

Ermöglicht das Umrechnen eines Analog-Faktors von 0.00 bis 1.00 in eine Raste innerhalb eines definierten Bereichs.

7.2. FUNKTIONSBAUSTEINE

Hier liegen alle Funktionsbausteine die für ein Fahrpult-Projekt zur Anwendung kommen könnten.

FB_CABDATA

Wie FUN_CABDATA, aber Instanz gebunden und Abfrage mehrerer Werte gleichzeitig.

FB_INPUT

Senden eines INPUT-Befehls an Zusi. Siehe Zusi-Doku Kapitel *11.3.3.6 Befehl 01 0A – INPUT (Client ! Zusi)*. Dieser Baustein ist Grundlage für u.a. FB_SCHALTER und FB_STICK.

FB_PZB90

Funktionsblock für PZB, ermöglicht Steuerung und Ausgabe.

FB_SCHALTER

Ermöglicht Betätigung eines beliebigen 1/0-Führerstand-Elements, z.B. das Horn. Für Zugbeeinflussung empfiehlt sich die Verwendung der dafür vorgesehenen Bausteine.

FB_SIFA

Funktionsblock für Sicherheitsfahrschalter, ermöglicht Steuerung und Ausgabe.

FB_STICK

Steuerung eines Führerstands-Elementes mit mehr als 2 Stellungen, z.B. Fahrschalter, Führerbremsventil.

FB_TAV

Funktionsblock für Technisches Abfertigungsverfahren, ermöglicht Steuerung und Ausgabe. Dies ist ein Beispiel zur Verwendung des FB_TUEREN mit Tastern zur Türfreigabe.

FB_TUEREN

Funktionsblock für allgemeine Türsteuerung, ermöglicht Steuerung und Ausgabe. Dieser FB kann grundsätzlich für jedes Türsystem eingesetzt werden, siehe z.B. FB_TAV.

FB_ZUSITCP

Kommunikationsbaustein. Siehe Kapitel 8

7.3. INTERN

Dieser Ordner enthält die internen Funktionen von libzusi. Diese können zwar extern verwendet werden, sind aber bisher nicht- oder nicht ausreichend dokumentiert. Allen internen Funktionen sind mit Z3_* benannt.

8. FB_ZUSITCP: VERBINDEN UND DATEN ANFORDERN

Jedes Fahrpult-Projekt muss genau eine Instanz des FB_ZUSITCP enthalten. Dieser verwaltet die Verbindung zum Server, überträgt um empfängt die notwendigen Daten.

8.1. DEM PROJEKT HINZUFÜGEN.

Im Bibliotheksverwalter libzusi hinzufügen (Bibliothek hinzufügen -> Sonstiges -> „*Zusi 3 TCP Client library for Codesys V3*“).

Alle Funktionen und Funktionsbausteine stehen nun im Namensraum „ZUSI“ zur Verfügung.

8.2. BAUSTEIN DEKLARIEREN

Im jeweiligen Programm unter „VAR“ eine Instanz des FB_ZUSI deklarieren:

```
PROGRAM DEMO_FUP
VAR
    (*Instanz des FB_ZUSITCP*)
    tcpclient: zusi.FB_ZUSITCP;
```

8.3. BAUSTEIN VERWENDEN

Dem FB_ZUSI müssen u.a. die IP-Adresse des Zusi-Servers bekannt sein sowie Name und Version des Clients:

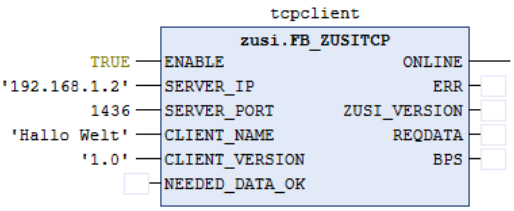


Abbildung 3

8.4. DATEN ANFORDERN

Für den erfolgreichen Login am Server muss vorab festgelegt werden, welche Daten vom Client benötigt werden. Dafür ist die Funktion `FUN_NEEDED_DATA` zuständig.

Zu beachten ist, dass die Funktion nur jeweils zum richtigen Zeitpunkt einmal ausgeführt wird, wenn der Ausgang `REQDATA` des `FB_ZUSITCP` `true` ist. Sind alle Daten angefordert, muss der Eingang `NEEDED_DATA_OK` des `FB_ZUSITCP` auf `true` gesetzt werden.

Angefordert werden können alle Werte aus der Zusidoku Tabelle *11.3.3.3.1 Führerstands-IDs*.

BEISPIEL FUP

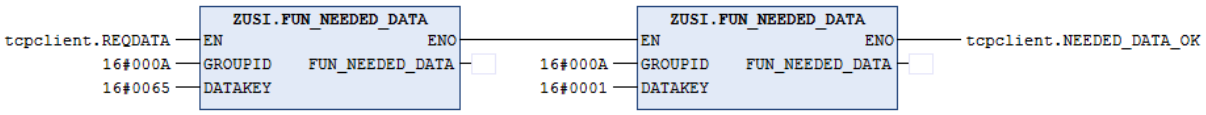


Abbildung 4: FUN_NEEDED_DATA mit EN/ENO

Hier wird die Funktion `FUN_NEEDED_DATA` in EN/ENO-Blöcke verpackt und dadurch nur ausgeführt, wenn `REQDATA` `true` ist. Am Ende der Kette wird `NEEDED_DATA_OK` gesetzt. Danach werden diese Funktionen nicht mehr ausgeführt.

BEISPIEL ST

Etwas weniger Aufwändig ist die Prozedur in der ST-Sprache:

```
IF tcpclient.REQDATA = TRUE THEN
       zusi.FUN_NEEDED_DATA(16#000A, 16#0001);
       zusi.FUN_NEEDED_DATA(16#000A, 16#0065);
       tcpclient(NEEDED_DATA_OK:= TRUE);
END_IF
```

8.5. VERBINDEN / TRENNEN

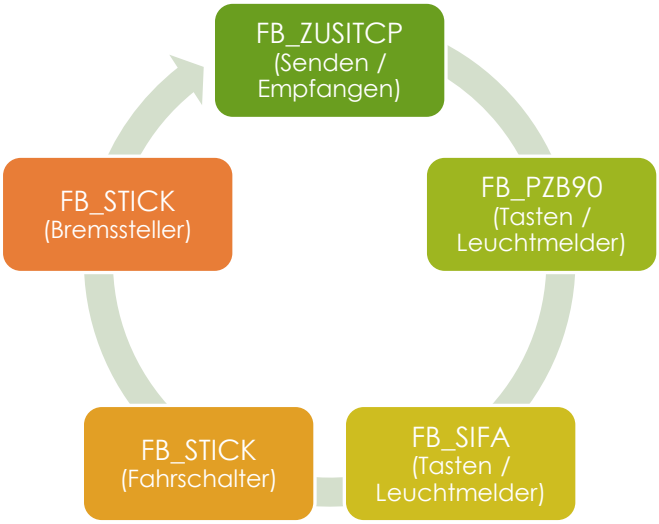
`FB_ZUSITCP` versucht die Verbindung herzustellen, sobald der Eingang `ENABLE` auf `TRUE` gesetzt ist. Sollte die Verbindung nicht sofort zustande kommen, wird automatisch nach ein paar Sekunden ein erneuter Versuch gestartet, bis die Verbindung hergestellt ist oder der Eingang `ENABLE` auf `FALSE` gesetzt wird.

Es gilt zu beachten, dass die Verbindungsherstellung die weitere Ausführung des Programmes so lange blockiert bis die Verbindung besteht oder durch Timeout abgebrochen wurde.

Die bestehende Verbindung kann abgebaut werden, indem der Eingang `ENABLE` auf `FALSE` gesetzt wird. Die Verbindung wird getrennt und der Baustein geht in den Leerlauf über.

8.6. PROGRAMMABLAUF

Der Zyklus eines typischen Fahrpult-Programmes könnte so aussehen:



9. FB_STICK: BEISPIEL HEBEL

In 99% aller Fahrpult-Programme wird es letztendlich um die Einbindung von Hebeln gehen. Hier die Erläuterung wie dies grundsätzlich Funktioniert.

9.1. HARDWAREBESCHALTUNG

Als Beispiel-Hardware sei hier der Industriestandart genommen: Die Analog-Eingabe-Hardware arbeitet mit einer Versorgungsspannung von 24V/DC und einem Messbereich von 0 – 10 V. Am Hebel kommt ein handelsübliches Potentiometer zum Einsatz. Die Beispiel-Beschaltung sieht so aus:

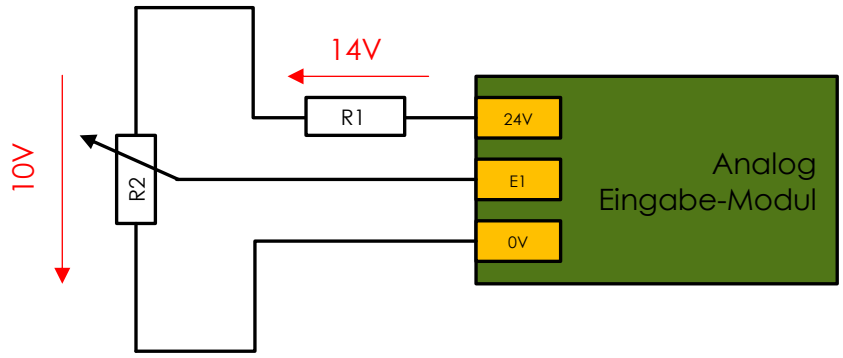


Abbildung 5: Schaltbild Analog-Eingang

Um den Eingang der AI-Hardware nicht zu überlasten kommt Vorwiderstand R1 zum Einsatz, so dass am Poti R2 maximal 10V abfallen. Der Abgriff des Potis wird an den Eingang E1 des AI-Moduls angeschlossen.

9.2. DAS PROGRAMM

Der Rohwert des Analog-Signals auf E1 wird intern in einem Datenwort (%IW1) abgelegt. Der Wert ist bei 0V = 0, bei Vollaussteuerung 10V = 1024.

Hinweis: sollte am Eingang nicht der volle Spannungsbereich zur Verfügung stehen, erreicht der Rohwert einen kleineren Maximalwert. In CODESYS kann der *Online-Konfigurationsbetrieb* genutzt werden um diesen Wert abzulesen. Grundsätzlich sollte immer möglichst der gesamte Messbereich ausgenutzt werden, da ein kleinerer Spannungsbereich zulasten der Genauigkeit ist.

analog1		in1	%IW0	WORD
		in2	%IW1	WORD
		in3	%IW2	WORD
		in4	%IW3	WORD
		in5	%IW4	WORD

Abbildung 6: Variablen-Mapping

Der Rohwert wird jetzt auf die neue Variable `analog1` gemappt.

Es soll ein Fahrschalter mit 33 Rasten angesteuert werden. In das Fahrpult-Programm wird nun eine Instanz des FB_STICK eingebunden (`hebel1`). Dieser benötigt folgende Daten:

- AI_MAX Maximaler Analogwert (1020)*
- AI_MIN Minimaler Analogwert (3)*
- AI_INP Analogsignal (analog1)
- NOTCH_MAX Höchste Fahrstufe (33)
- NOTCH_MIN Kleinste Fahrstufe (0)
- ASSIGN Tastaturzuordnung lt. Zusi-Doku Kapitel 11.3.3.4 (16#0001 Fahrschalter)
- ENABLE Es soll nur gesendet werden, wenn Verbindung besteht.

BEISPIEL FUP

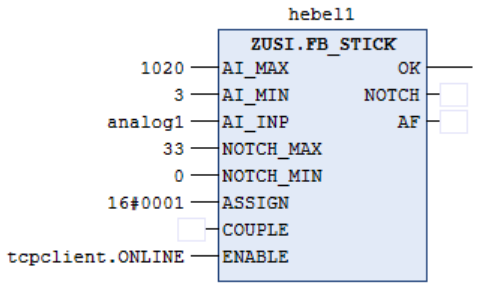


Abbildung 7

BEISPIEL ST

```
hebel1( AI_MAX:= 1020,
        AI_MIN:= 3,
        AI_INP:= analog1,
        NOTCH_MAX:= 33,
        NOTCH_MIN:= 0,
        ASSIGN:= 16#0001,
        ENABLE:= tcpclient.ONLINE );
```

*) Um unterste und oberste Raste überhaupt zu erreichen ist es oft sinnvoll etwas unterhalb / überhalb des maximalen / minimalen Analogwertes zu arbeiten, deshalb 1020 statt 2024 und 3 statt 0.

9.3. TESTEN

Nach dem Download auf die Steuerung kann im Online-Betrieb, auch wenn keine Verbindung zu Zusi besteht, durch drehen am Poti die Ausgabe der richtigen Fahrstufe getestet werden.

10.DATEN ABRUFEN

Neben dem Senden von Befehlen ermöglichst Libzusi das abrufen von Simulator-Daten wie z.B. Geschwindigkeit und Bremsdruck.

Dazu werden folgende Funktion / folgender Funktionsbaustein zur Verfügung gestellt.

10.1. FUN_CABDATA

Die Funktion `FUN_CABDATA` ermöglicht den Abruf eines einzelnen Wertes im REAL-Format (Single-Format in Zusi: einfache Fließkommazahl). Es können nur Daten abgerufen werden die zuvor mit `FUN_NEEDED_DATA` (Kapitel 8.4) angefordert wurden.

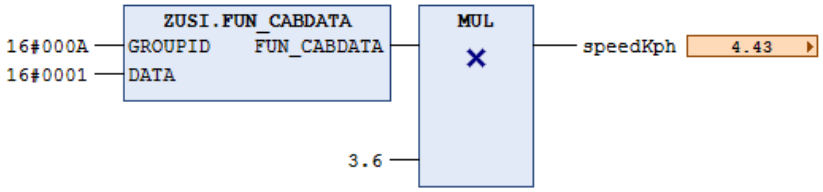


Abbildung 8: Abruf Geschwindigkeit mit Umrechnung in km/h

10.2. FB_CABDATA

Der Funktionsbaustein `FB_CABDATA` ist die Instanz gebundene Entsprechung der `FUN_CABDATA` und ermöglicht den Abruf von bis zu fünf Werten gleichzeitig.

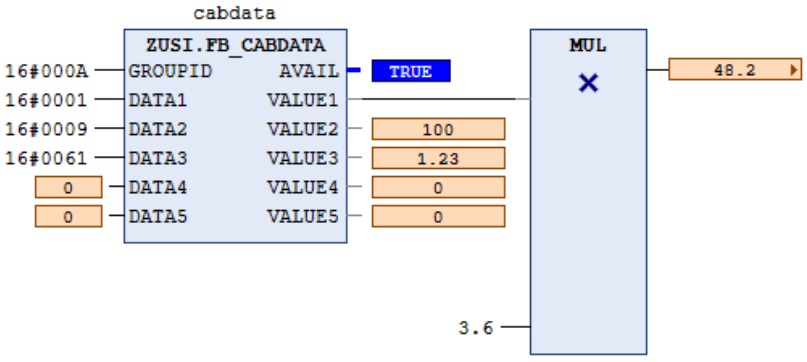


Abbildung 9: Abruf Geschwindigkeit, Zugkraft und Kilometer

Der Ausgang `AVAIL` wird `TRUE` sobald mindestes einer der fünf Werte verfügbar ist.

11.KOMPLEXE FUNKTIONSGRUPPEN

Für Funktionen wie Zugbeeinflussung und Türsteuerung, welche aus komplexen Gruppendaten zusammengesetzt sind (Zusi-Doku ab *11.3.3.3.2 Status Notbremssystem*), stellt Libzusi eigene Funktionsbausteine zur Verfügung. Diese vereinigen Ein- und Ausgabe der jeweiligen Baugruppen (Grundsätzlich für alle Funktionsbausteine gilt: Links Eingabe, Rechts Ausgabe).

11.1. BEISPIEL: FB_PZB90

Der Funktionsbaustein FB_PZB90 dient der Steuerung und Anzeige der Baugruppe PZB90. Damit der Baustein seine Werte abrufen kann, muss die Funktionsgruppe mit NEEDED_DATA angefordert werden (16#0065 „Status Zugbeeinflussung“ -> Siehe Kapitel 8.4).

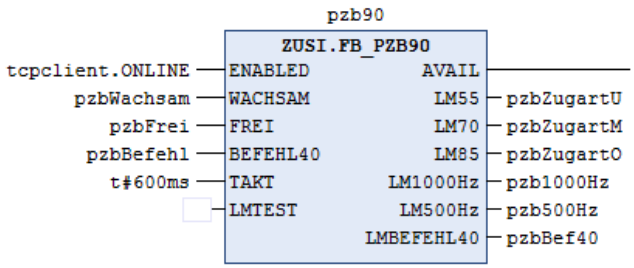


Abbildung 10: FB_PZB90 mit zugewiesenen Variablen

Dem FB werden auf der linken Seite Eingabebefehle in Form von Variablen (aber auch direkte Adressierung möglich) zugewiesen, auf der rechten Seite gibt der FB den Zustand der Leuchtmelder aus.

Die Blinkgeschwindigkeit kann über den Eingang `TAKT` eingestellt werden. Vorgegeben ist ein Takt von 400 ms. Der Eingang `LMTEST` schaltet alle Ausgangs-Signale auf `TRUE` und dient dem Lampentest.

Der Ausgang `AVAIL` wird `TRUE` sobald PZB-Werte verfügbar sind.