FAKULTÄT FÜR
ELEKTROTECHNIK UND
INFORMATIONSTECHNIK

INSTITUT FÜR
AUTOMATISIERUNGSTECHNIK

Implementing PROFIBUS Support for FORTE

Thomas Hadlich

Technical Report
(Draft)

IFAT-LIA 1/2014

22.08.2014

Universität Magdeburg
Fakultät für Elektrotechnik und Informationstechnik
Institut für Automatisierungstechnik
Postfach 4120, D-39016 Magdeburg
Germany

1 General

When a fieldbus communication shall be integrated into the 4DIAC-suite, two basic questions need to be answered:

- How will the communication be integrated into the 1499-Fieldbuss-Network?
- How will the communication between the control and the I/Os be configured?

As explained in [Zoi14a], an layered approach should be used for implementing support for communication [Zoi14b].

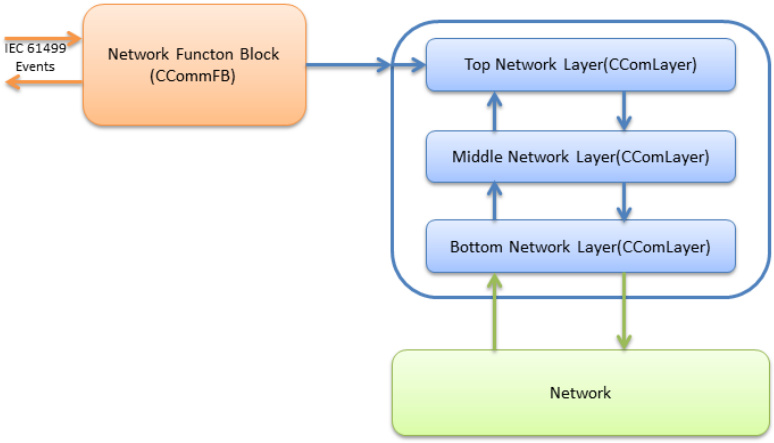

Figure 1 – Network communications [Zoi14a]

With PROFIBUS (as with most fieldbuses), the communication needs to be configured in order to work properly.

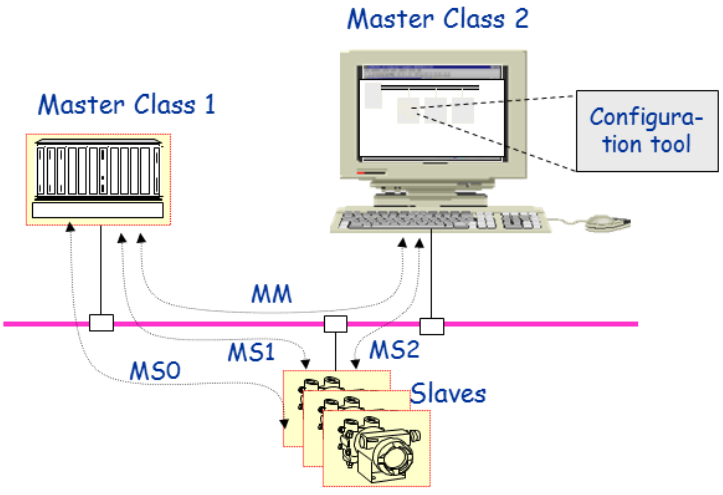

Figure 2 – Overview of PROFIBUS components (based on [Die12])

The PROFIBUS protocol is based on master/slave communication. There are two types of PROFIBUS master: Master Class 1 (with cyclic data transfer) and Master Class 2 (with acyclic data transfer). The cyclic data transfer, provides a reliable, high prior, deterministic way of communicating data. A control typically acts as Master Class 1, using the cyclic data transfer to communicate with the devices, which

act as slave. Be aware that the figure describes an IEC61311 environment, where I/O communication may be provided natively by the PLC, with an IEC61499 environment the communication is provided by SIFB [Die12].

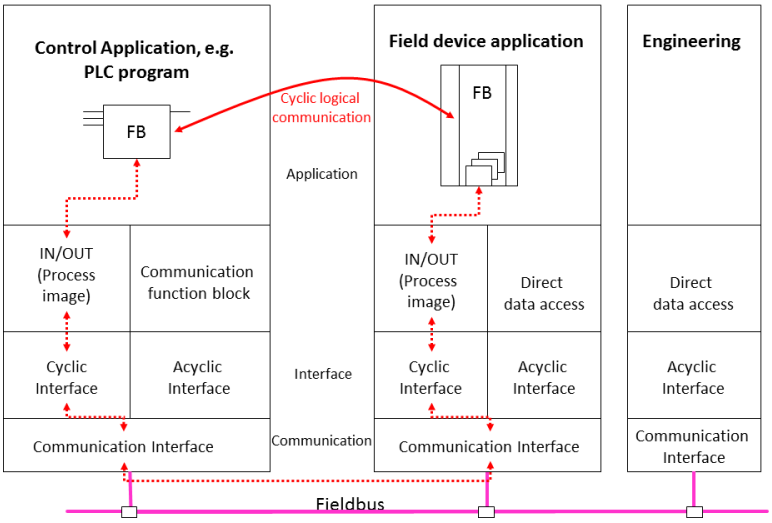

Figure 3 – Cyclic communication(based on [Die12])

An engineering workstation, which is used to configure the devices, uses acyclic data transfer with direct access to all configuration data.

A PROFIBUS Master Class 1 needs at least following configuration:

- General
- Station address of the master (typically an low address, e.g. 0 or 1)
- Baud rate for the communication
- different time settings (based on the station address and on the baud rate)
- for each slave
- Station address of the slave (range goes up to 125, 126 is for devices that are not configured yet)
- module configuration
- configuration of I/O data (based on module configuration)
- different further settings (e.g. watch dog timer)

This means, when integrating PROFIBUS into the FORTE runtime, a communication stack needs to be integrated, which allows to communicate a number of variables. It will be necessary to configure the stack for communication to multiple devices with different I/Os. A central configuration for the communication is needed and the communication access needs to go through a central communication hub.

2 Approach

The approach proposed in this document is to use an OPC server for integration of the fieldbus. The integration of OPC servers into FORTE is described in [Zoi14c]. FORTE has been enabled to access OPC servers as a client.

Within the IEC61499-Application the 'CLIENT_X_X' Function Blocks are used for communication

with the OPC-Server. The number of inputs and outputs of the Client-FB depends on how many values shall be communicated with the server.

For example, a *CLIENT_1* Function Block is added to the *Application* in order to read one value and to write one value.The input values are set as shown below, and map it to the hardware.

```
QI = 1

ID = opc[localhost;isPro.MultiServer.1;2000;0.01; dp://brd0.seg0.dev15/Outareai0d17;
                                                  dp://brd0.seg0.dev15/Inareai1d17]

SD_1 = <input-value>
```

The ID input has the following syntax:

```
opc[host;serverName;updateRate;deadBand;fbInputItems;fbOutputItems]
```

- **host**: address to computer with OPC server
- **serverName**: OPC server name, e.g. Matrikon.OPC.Simulation
- **updateRate**: update frequency in milliseconds
- **deadBand**: dead band used for update of value (same unit as value)
- **fbInputItems/fbOutputItems**: items to be added, items should be separated with a comma

Please note that a different syntax is used than in [Zoi14c]. The syntax was changed in order to enable compatibility with the OPC server.

A standard mechanism for integrating fieldbus communication is defined by the DriveServer specification. There is a set of OPC servers that supports the DriveServer communication server specification [DRI01, RTH01].

For this project the main features of the DriveServer specification for communication servers are:

- the names of items may be defined with a syntax, that specifies the address of the data, the length of the data, as well as the data type
- the name space of the server is extendable (within limits)
- the approach works for I/O data as well as for configuration data

3 is Pro MultiServer

is Pro MultiServer is a product of ifak system and supports PROFIBUS Master Class 1 communication as well as Master Class 2 communication [ifa08].

Since target of the project is to control an automated system, the server will be configured to use Master Class 1 communication. For this, the server is configured in a preparation step with the information on the devices (for instance with address and general size of I/O). After the server is configured with device information, the OPC server may be started. The OPC server automatically starts the (cyclic) communication to the devices. When the OPC Client connects to the server, it may specify items that correspond to the data in the I/O-Area of the device.

The syntax of the OPC Server for accessing an I/O-value (represented as an OPC item) is:

```
<protocol>://brd<interface>.seg<seqment>.dev<slave>/<data kind>i<position>d<datatype>[s<length>]
```

- **<protocol>** - denotes the Profibus-protocl to be used ('dp' for cyclic communication, 'dp2' for acyclic communication)
- **<interface>** - number(zero-based), denotes the Profibus-interface card of the PC
- **<seqment>** - number(zero-based), denotes the Profibus-Port of the inferface card
- **<slave>** - number, the station address of the slave
- **<data kind>** - denotes the type of data that shall be communicated
    - 'inarea' for input data,
    - 'outarea' for output data,
    - 'par' for configuration parameter)

- **<position>** - number(zero-based), denotes the position of the data in the respective data area
- **<datatype>** - denotes the OPC data type
- **<length>** - denotes the length of data

For example:

dp://brd0.seg0.dev15/Inareai1d17

describes the access via the first Profibus interface of the PC, the first port of the interface, access to slave with station address 15, input data of the slave, the second byte of the data, to be represented as datatype Byte (this data should be mapped to output of the FB)

dp://brd0.seg0.dev15/Outareai0d17

describes the access via the first Profibus interface of the PC, the first port of the interface, access to slave with station address 15, output data of the slave, the first byte of the data, to be represented as datatype Byte (this data should be mapped to input of the FB)

4 Implementation

The structure of the software is shown in Figure 4.

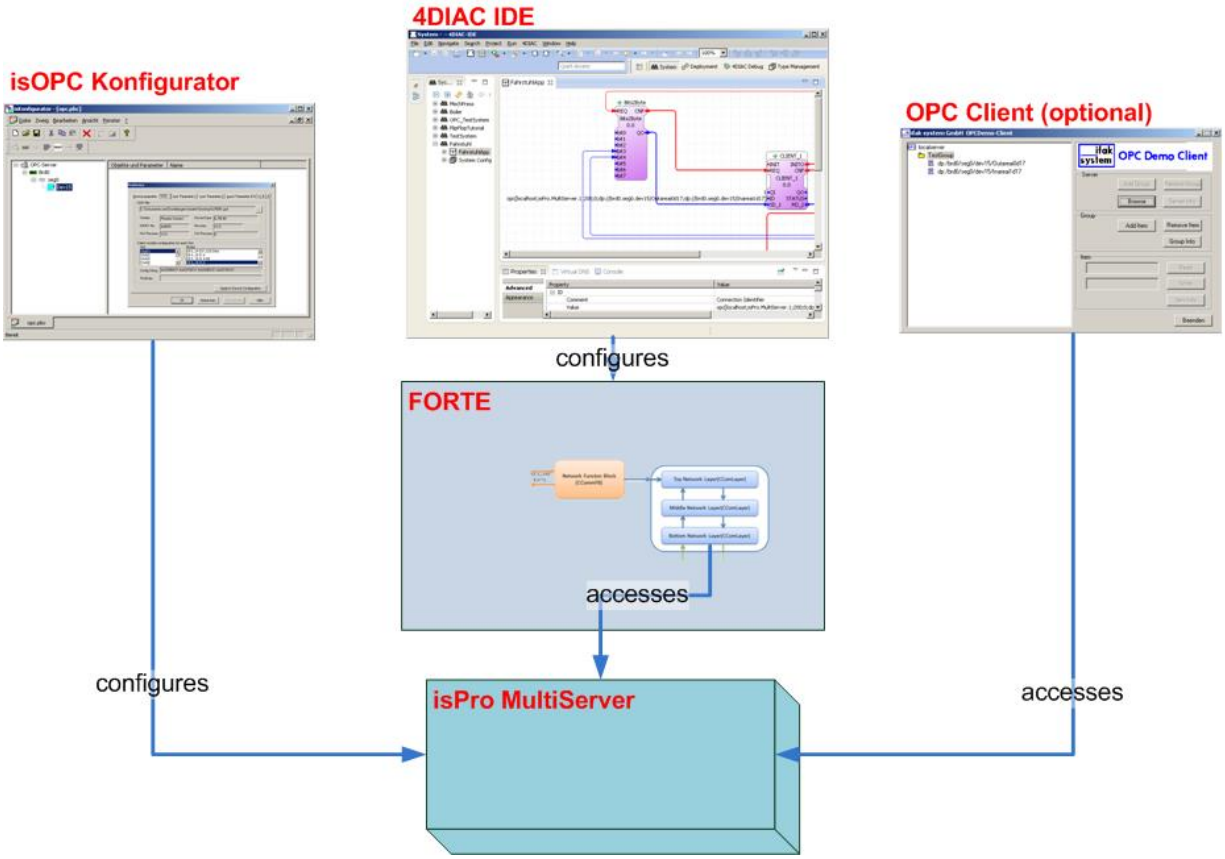

Figure 4 – SW-Structure

The isOPC Konfigurator is used to configure a device with its address and I/O configuration. For this configuration the GSD of the device may be used (see Figure 5).

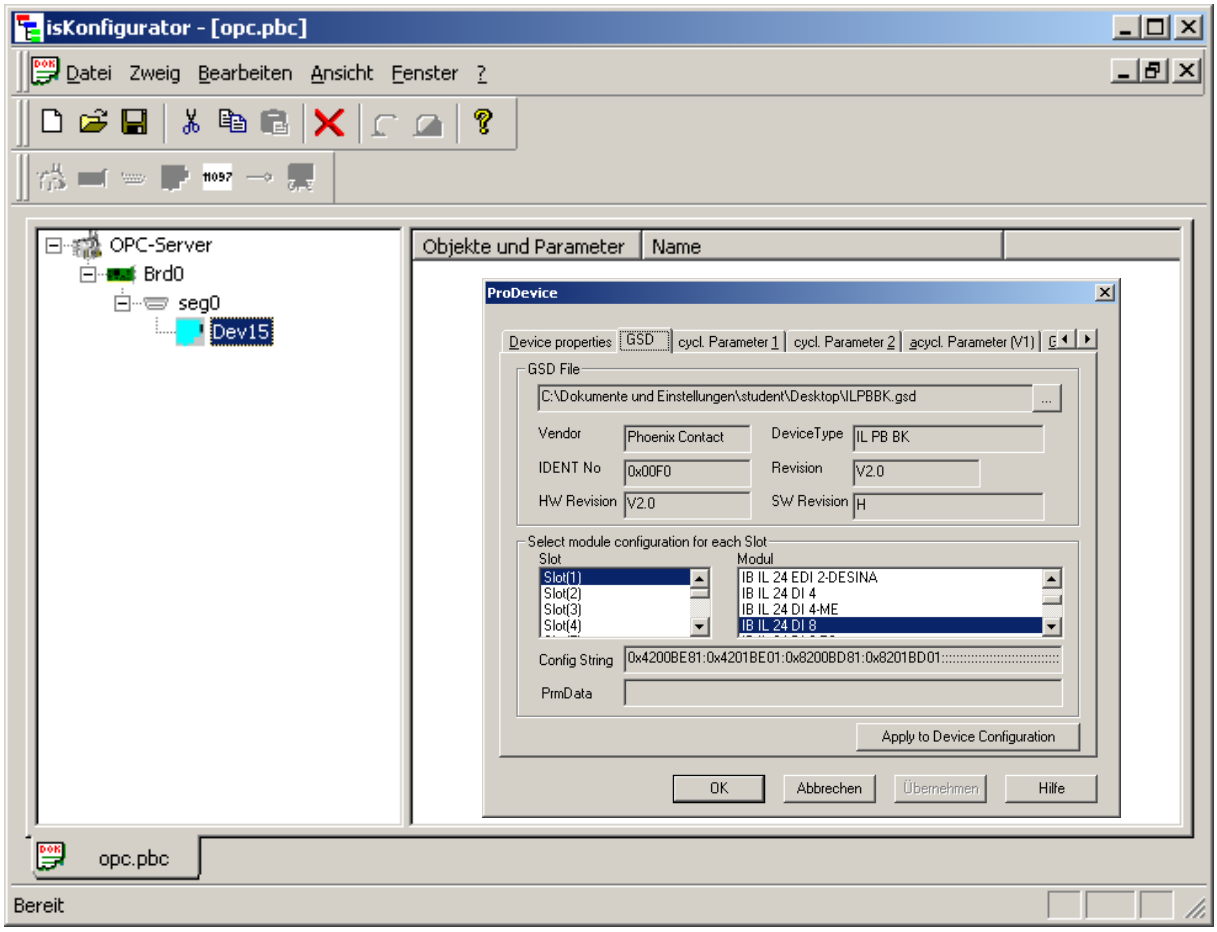

Figure 5 – is OPC Konfigurator

If no GSD is available, it may be sufficient just to configure the size and position of the I/O data (see Figure 6).

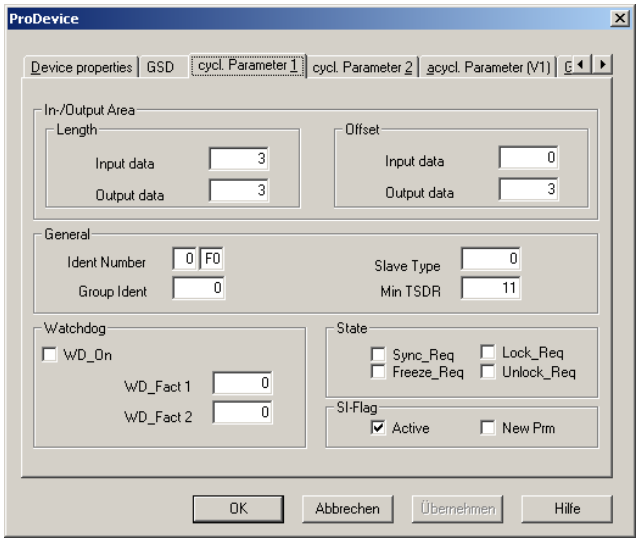

Figure 6 – Setting size and position of I/Os

Within 4DIAC the Function Block application is designed to access the OPC server and to retrieve specific I/O data from the OPC Server (see Figure 7). Please note, that the fbInputItems need to be mapped to the OutArea of the device and the fbOutputItems need to be mapped to the InArea of the device.

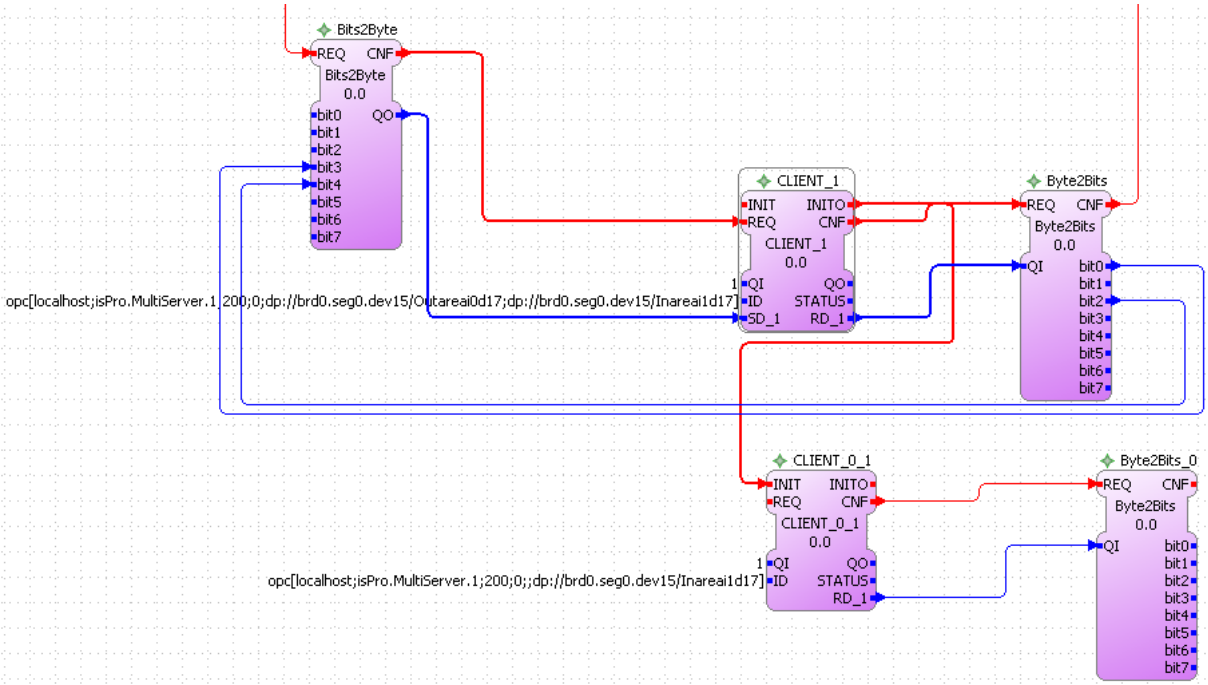

Figure 7 – Designing the access to data

When the application is started and the Client-Function Blocks are initialized, they create a connection to the OPC Server and access the configured items. If these items do not yet exist, the OPC Server executes a validation and creates the items after successful validation. These items then can also be accessed with other OPC Clients (see).

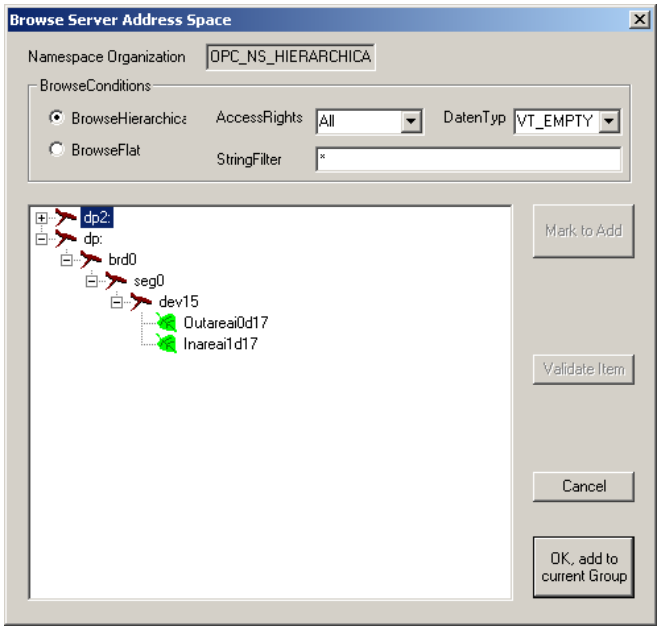

Figure 8 – The dynamically created items in other OPC Client

5 Summary

With this report an approach is introduced to integrate fieldbus communication into an IEC 61499 environment via OPC. Since the approach is based on OPC connections, the field of application for this approach may be limited, but it allows for fast and easy creation of the required fieldbus access. It may be used for non-critical applications like build automation or prototyping of control design.

6 Literature

[Die12] Diedrich, C.: Lecture Communication Systems 2012, Magdeburg, 2012.

[DRI01] DriveServer Specification v1.1, Blomberg, 2001.

[ifa08] ifak system GmbH Hrsg.: isPro MultiServer - Benutzerhandbuch. ifak system GmbH, Magdeburg, 2008.

[RTH01] Riedl, M.; Thron, M.; Hadlich, T.: DriveServer-significantly reduce in engineering expense: IECON 2001. The 27th Annual Conference of the IEEE Industrial Electronics Society. IEEE Press, Piscataway, NJ, 2001; S. 285–288.

[Zoi14a] Distributed Industrial Automation Web Site: FORTE - Communication Architecture. http://fordiac.sourceforge.net/ehelp/html/development/forte_communicationArchitecture.html, Geprüft am: 21.08.2014.

[Zoi14b] Distributed Industrial Automation Web Site: 4DIAC Help. http://fordiac.sourceforge.net/ehelp/, Geprüft am: 21.08.2014.

[Zoi14c] Distributed Industrial Automation Web Site: FORTE - OPC DA. http://fordiac.sourceforge.net/ehelp/html/communication/opc.html, Geprüft am: 21.08.2014.