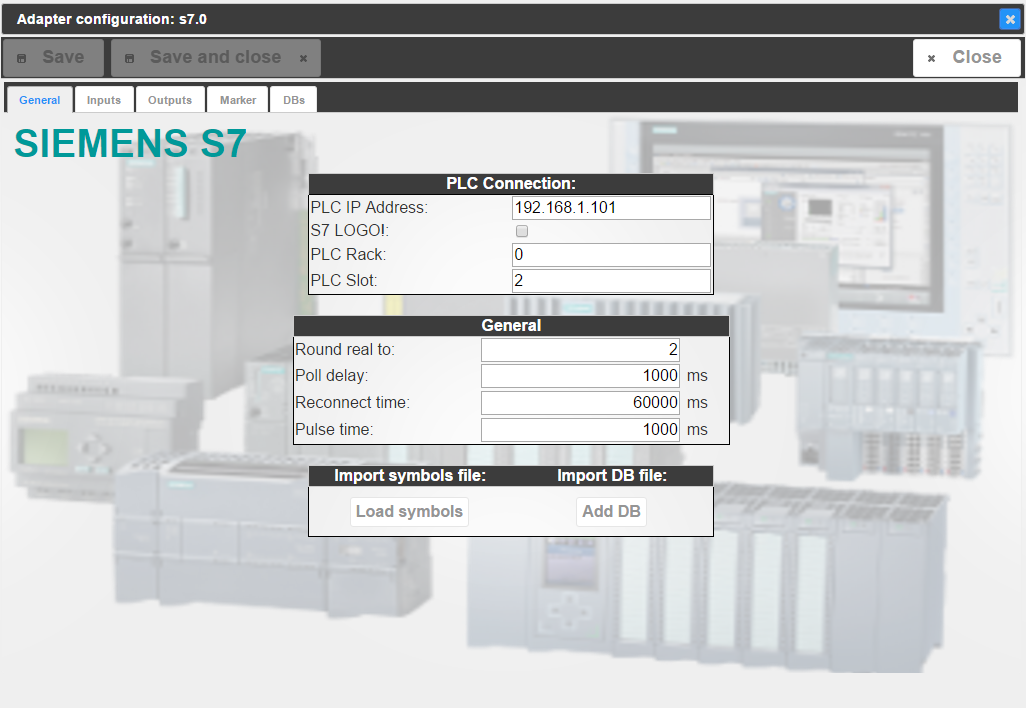
ioBroker

Siemens SIMATIC S7 Adapter



Configuration guide

Contents

[Document versioning 3](#_Toc439526889)

[Introduction 4](#_Toc439526890)

[Communication through Data Blocks (DBs) 5](#_Toc439526891)

[Generate communication DBs 5](#_Toc439526892)

[Preparation of DB20 – Binary values sent from ioBroker to the S7 5](#_Toc439526893)

[Preparation of DB21 – Binary values sent to ioBroker from the S7 6](#_Toc439526894)

[Preparation of DB22 – Real values sent from ioBroker to the S7 6](#_Toc439526895)

[Preparation of DB23 – Real values sent to ioBroker from the S7 7](#_Toc439526896)

[Create DB sources in Step7 8](#_Toc439526897)

[Populate DBs to ioBroker 13](#_Toc439526898)

[Installation of S7 Adapter instance 13](#_Toc439526899)

[Configure ioBroker for the communication 16](#_Toc439526900)

[Communication test 19](#_Toc439526901)

[Monitoring and Operation in vis 20](#_Toc439526902)

# Document versioning

* V0.1 English, dated January 02, 2016
* ioBroker S7 Adapter V0.2.4
* Step7 V5.5 SP4 HF5
* Changes to previous version:
  + Language German to English
  + DB configuration with spread sheet tool (CSV)
  + No more DB handling with text printer

# Introduction

The following document aims to guide you through the steps required to get ioBroker to talk to a Siemens SIMATIC S7 PLC.

The S7 adapter which comes with ioBroker is based on Snap7. Snap7 will be installed during the first time installation of the adapter and handles the TCP/IP communication between the S7 PLC and ioBroker.

So it is mandatory that the S7 is equipped with an Ethernet interface (integrated or external CP) in order to communicate over TCP/IP with the hardware ioBroker is running on.

As a prerequisite the user has to know the basics about TCP/IP communication and he/ she has to be able to configure the S7 PLC with the Step7 software. But that shouldn’t be a challenge for someone considering to linking an S7 to ioBroker.

This guide is based on the following configuration:

* S7-315 with integrated Ethernet interface
* Raspberry Pi 2, ioBroker running under Debian GNU/Linux 7.8 (wheezy)
* IP address range 192.168.1.xxx
* PC running:
  + Spread sheet tool like MS Excel, Apache Open Office
  + Google Chrome Browser
  + Step7 V5.5 SP4 HF5

Now let’s get started!

# Communication through Data Blocks (DBs)

This guide describes the communication between ioBroker and the S7 PLC through data blocks. Ideally dedicated DBs can be generated for the communication. The DBs have to be integrated in the code running in the S7. The advantage with that approach is that you can be sure that you won’t overwrite data accidentally for example in an instance data block which could lead to unwanted or unexpected reactions in your S7 software.

If you have to use existing data blocks due to memory restrictions or that you can not do any modification to the S7 software, make sure that you populate the relevant data to ioBroker only in order to avoid conflicts.

## Generate communication DBs

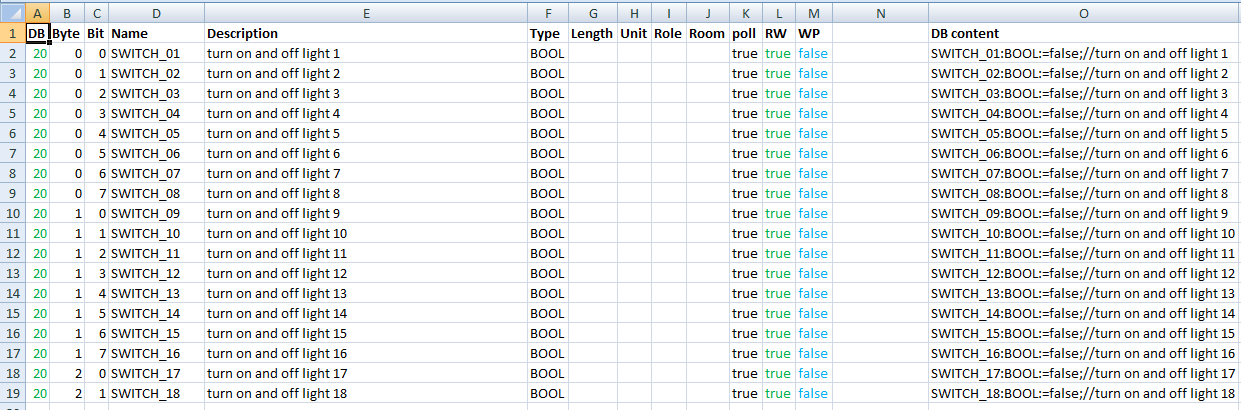
We are going to work with 4 DBs:

* DB20 – Binary values sent from ioBroker to the S7 (digital input from an S7 view)
* DB21 – Binary values sent to ioBroker from the S7 (digital output from an S7 view)
* DB22 – Real values sent from ioBroker to the S7 (analog input from an S7 view)
* DB23 – Real values sent to ioBroker from the S7 (analog output from an S7 view)

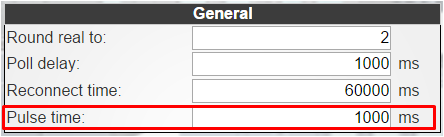
The DBs will be generated using a spread sheet with one table per data block.



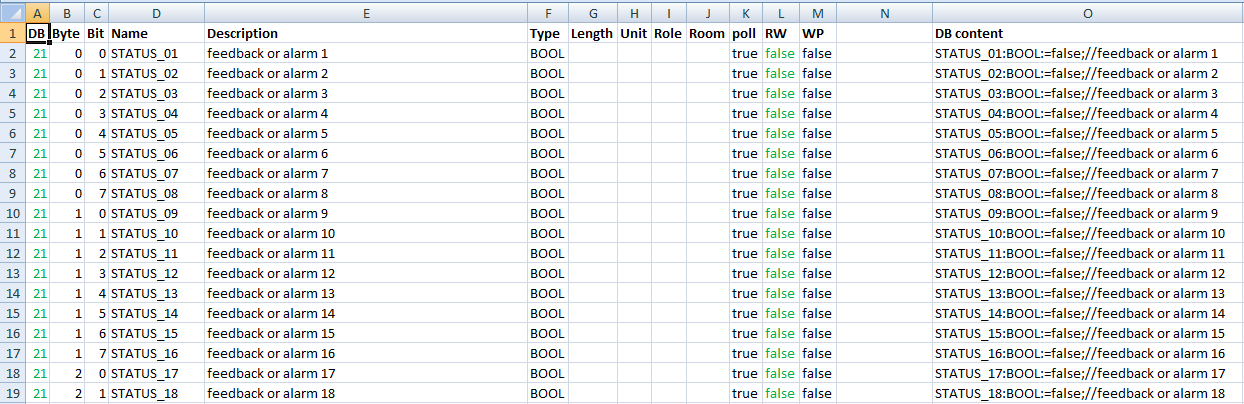
### Preparation of DB20 – Binary values sent from ioBroker to the S7



Column A through M are based on the structure in ioBroker and have to be filled in by the user based on the S7 software. You might want to make use of parts of the S7 symbol table (copy – paste). In column O the code for the S7 DB is derived from the content in column A through M.

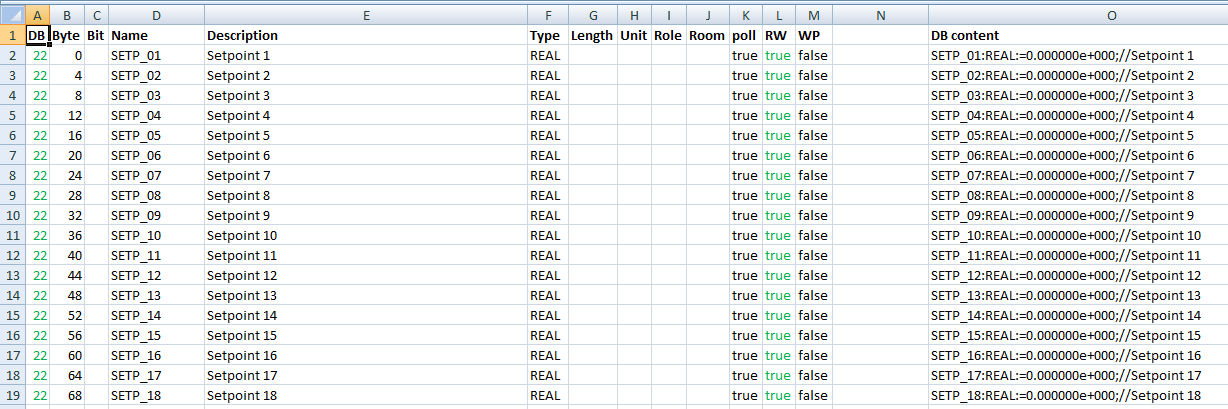
* Column A: DB = DB Number in the S7 and first part of address in ioBroker
* Column B: Byte = Byte in DB in the S7 and second part of address in ioBroker
* Column C: Bit = Bit in DB in the S7 and third part of address in ioBroker
* Column D: Name = Name in DB in the S7 and name in ioBroker
* Column E: Description = Comment in DB in the S7 and description in ioBroker
* Column F: Type = Type in DB in the S7 and type in ioBroker
* Column G: Length = length in ioBroker
* Column H: Unit = unit in ioBroker
* Column I: Role = role in ioBroker
* Column J: Room = room in ioBroker
* Column K: Poll = data point will be polled cyclically (true/false)
* Column L: RW = data point can be written (true/false) 🡺 “true” in DB20 as we want to write data to the S7
* Column M: WP = data point will be set to “1” only for the “pulse time” defined under “General – General”  
  
* Column N: intentionally left empty
* Column O: DB content = content which will be copied to Step7 for DB generation, formula: *=CONCATENATE(D2;":";F2;":=";"false;";"//";E2)*

### Preparation of DB21 – Binary values sent to ioBroker from the S7



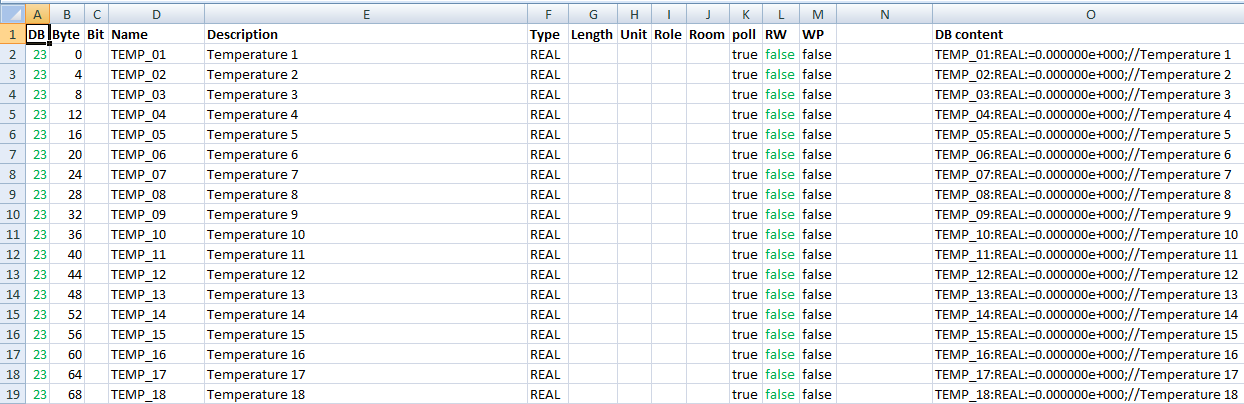
* Column L: RW 🡺 “false” in DB21 as we want to read data from the S7

### Preparation of DB22 – Real values sent from ioBroker to the S7



* Column B: Byte = start byte of real value (0, 4, 8, …)
* Column C: Bit = left empty
* Column L: RW 🡺 “true” in DB22 as we want to write data to the S7
* Column O: Formula: *=CONCATENATE* *(D2;":";F2;":=";"0.000000e+000;";"//";E2)*

### Preparation of DB23 – Real values sent to ioBroker from the S7

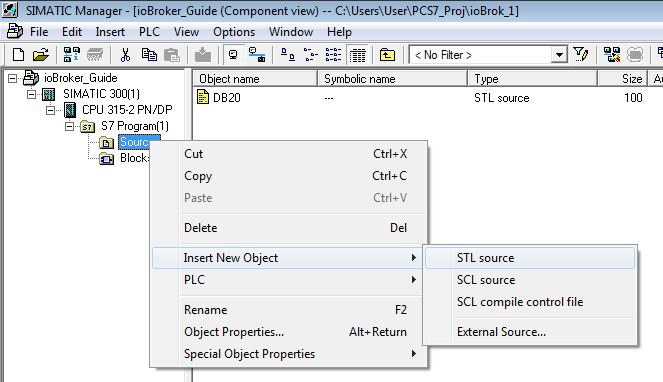


* Column B: Byte = start byte of real value (0, 4, 8, …)
* Column C: Bit = left empty
* Column L: RW 🡺 “false” in DB23 as we want to read data from the S7
* Column O: Formula: *=CONCATENATE* *(D2;":";F2;":=";"0.000000e+000;";"//";E2)*

### Create DB sources in Step7

We will now generate the DBs in Step7 using the code in column O of our spread sheet.

In your Step7 program insert an STL source by clicking the right mouse button on “Sources”.



Rename the new source to “DB20”. Insert the following code in the empty source:

*DATA\_BLOCK DB 20*

*TITLE =*

*VERSION : 0.1*

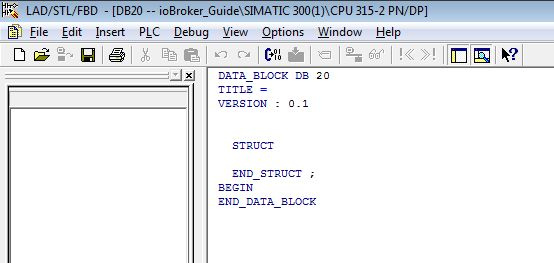
*STRUCT*

*END\_STRUCT ;*

*BEGIN*

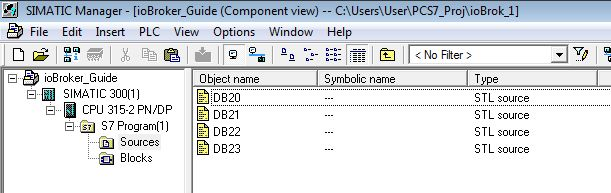
*END\_DATA\_BLOCK*

The source should look like this:



Copy the source “DB20” 3 times and name the copies DB21, DB22, DB23 while also changing line one in each source to:

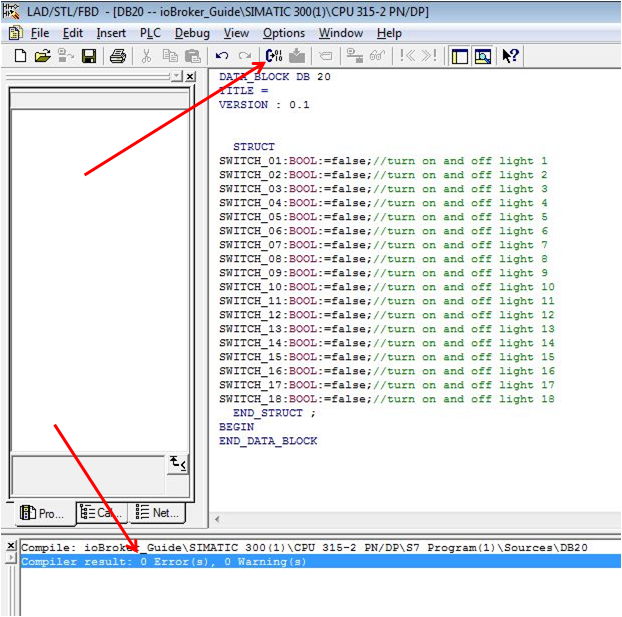
* *DATA\_BLOCK DB 21*
* *DATA\_BLOCK DB 22*
* *DATA\_BLOCK DB 23*



Now go to the spread sheet, table DB20, and copy the code in column O (without headline):

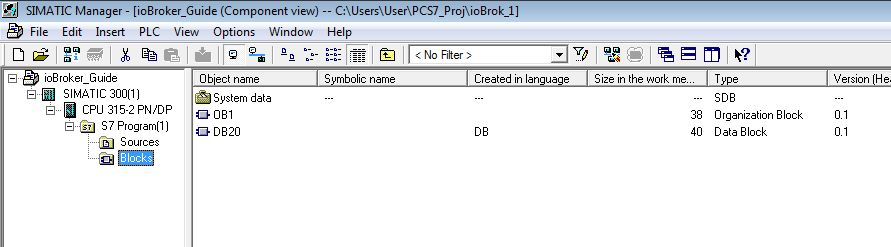


Paste the cells in the source called “DB20” in Step7 between “STRUCT” and “END\_STRUCT;”:

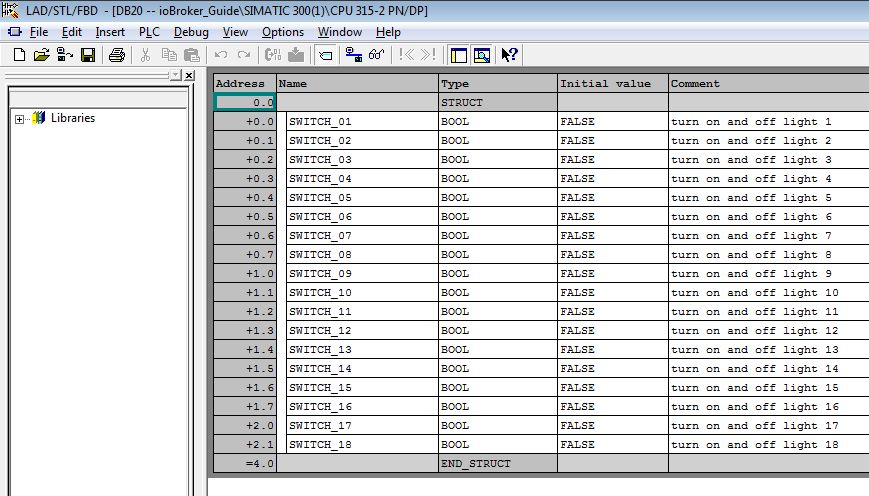


Start the compiler and the result should be 0 Errors, 0 Warnings.

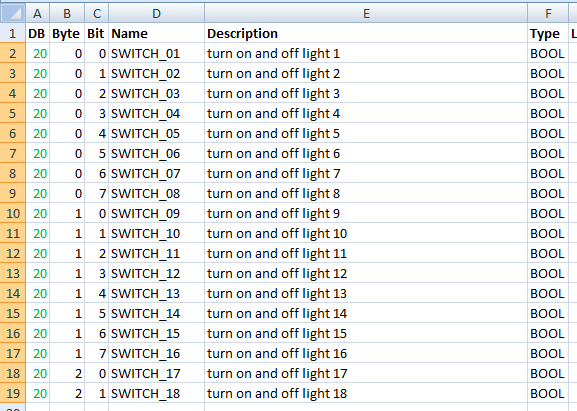
DB20 has now been generated and you find the new block under “Blocks” in your S7 program.



The block looks like this:

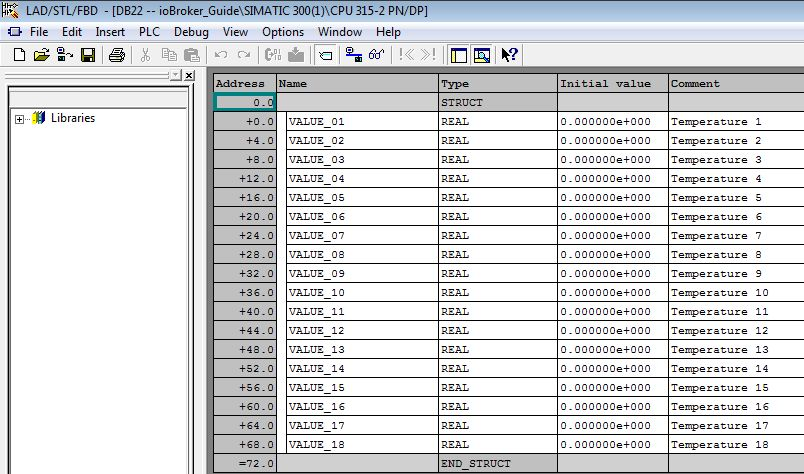


The address should be in correspondence with the address in the spread sheet, just do a sanity check by comparing the combination of Byte and Bit:

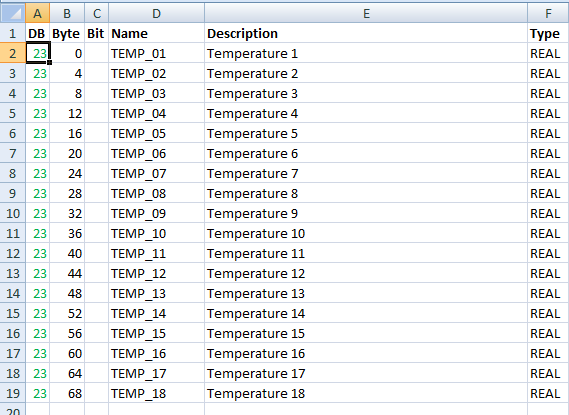


Repeat for DB21, DB22, DB23 and make sure you pick the column O from the right table and paste it to the correct source (table DB21 to source DB21 etc.)

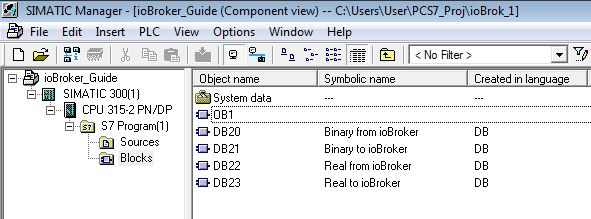
As DB22 and 23 will deal with REAL values, you can find below how the blocks will look.



Also here the address should correspond with the spread sheet (Byte):



We now have the 4 DBs required for the communication:



You should give them a symbolic name accordingly, that helps to maintain clarity.

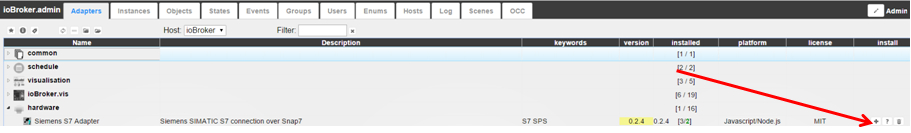
Don’t forget to connect them to the S7 logic and download the modified code.

## Populate DBs to ioBroker

Now that the 4 DBs are part of the code running in the S7, we will tell ioBroker how to communicate with the S7.

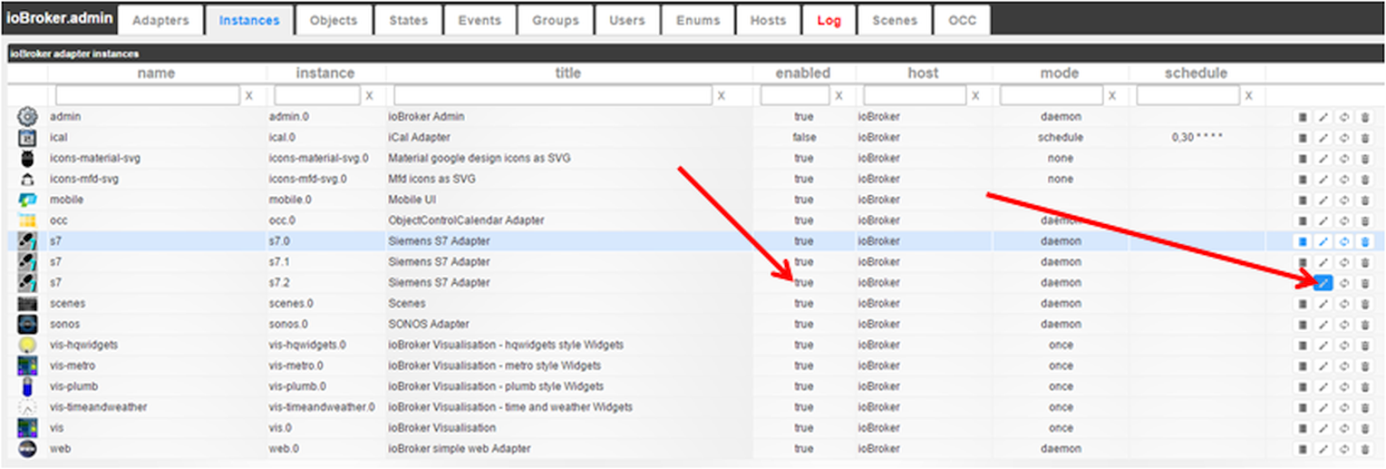
### Installation of S7 Adapter instance

Adapters – hardware – Siemens S7 Adapter – +



Multiple instances are possible in case you want your ioBroker to connect with multiple S7 CPUs.

Enable the new adapter instance:

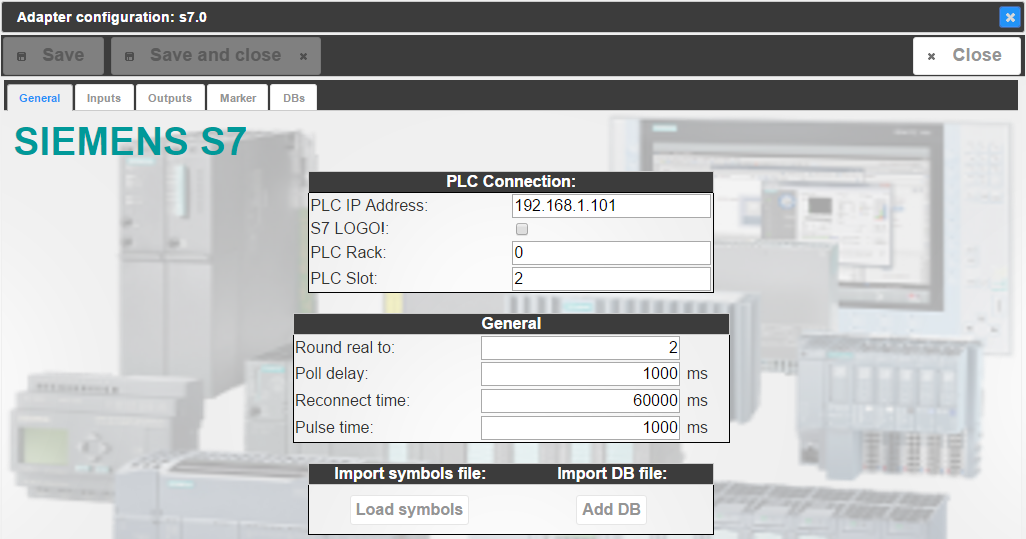


The title of the adapter (standard: Siemens S7 Adapter) can also be changed in that step. Using the IP address as part of the title would be one idea.

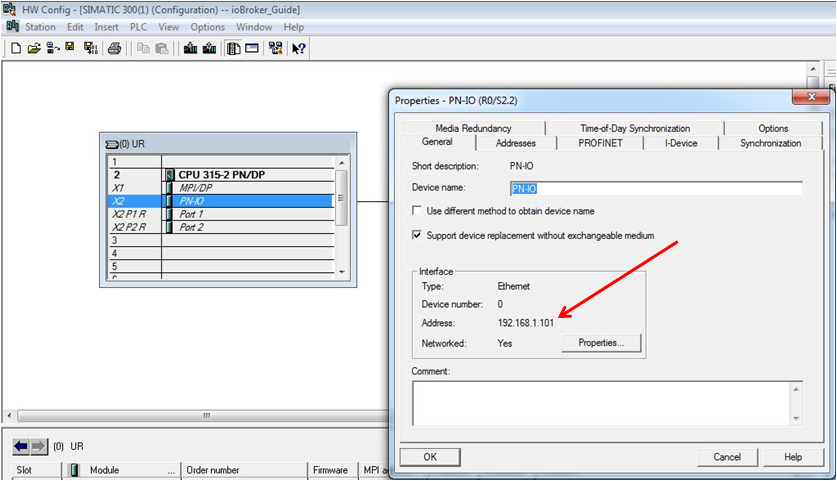
Open the adapter configuration



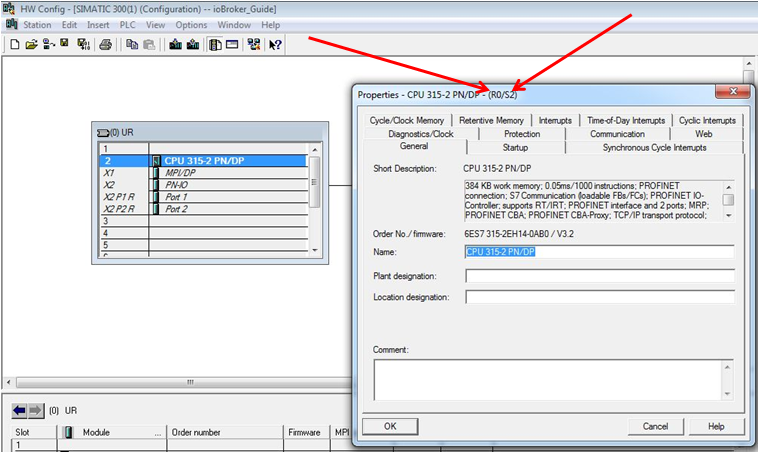
and start to configure the S7 adapter:



* Tab “General”
  + PLC Connection
    - PLC IP Address  
      IP address of the PLC as defined in the Step7 HW Config



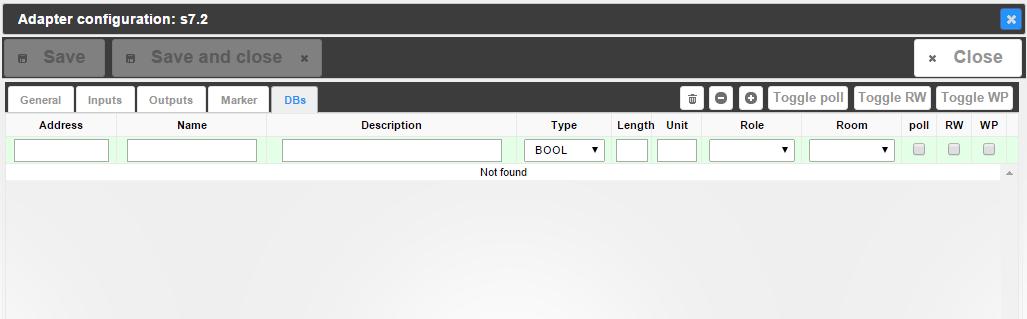
* + - S7 LOGO!  
      In case you’re using a LOGO, not an S7 PLC
    - PLC Rack  
      Rack number of the CPU as found in the Step7 HW Config (**R0**/S2)
    - PLC Slot  
      Slot number of the CPU as found in the Step7 HW Config (R0/**S2**)



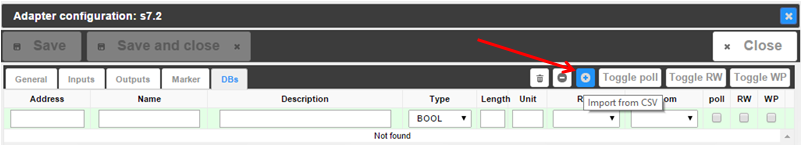
* + General
    - Round real to:  
      Number of digits real values will be rounded to after separator, e.g.:  
      2 🡺 12.12  
      3 🡺 12.123  
      …  
      9 🡺 12.123456789
    - Poll delay:  
      Communication update cycle in milliseconds
    - Reconnect time:  
      Duration in milliseconds after a reconnect will be tried once the connection to the S7 was lost
    - Pulse time:  
      Time in milliseconds for “1” for data points configured as WP = true
  + Import symbols file:
    - Load symbols  
      Feature to import Step7 symbols from an ASCII file – not used here
  + Import DB file:
    - Add DB  
      Feature to import Step7 DBs from an ASCII file – not used here

### Configure ioBroker for the communication

We skip the tabs “Inputs”, “Outputs” and “Markers” and go right to “DBs”:

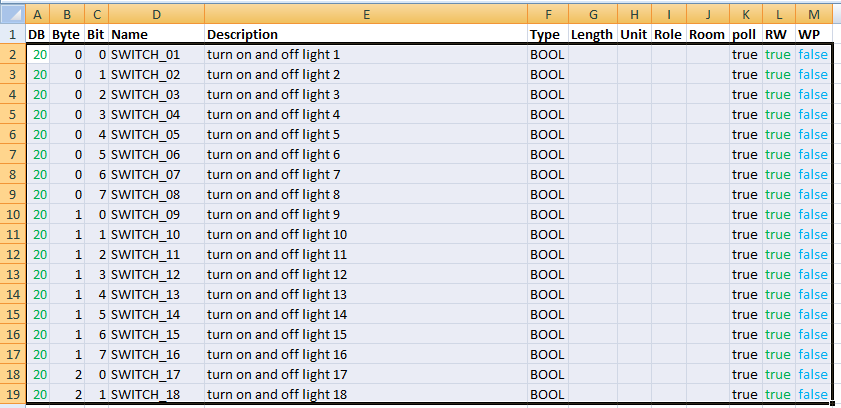
  
Here you can find the structure of the spread sheet. We’re ready for bulk engineering again.

Click the “Import from CSV” button

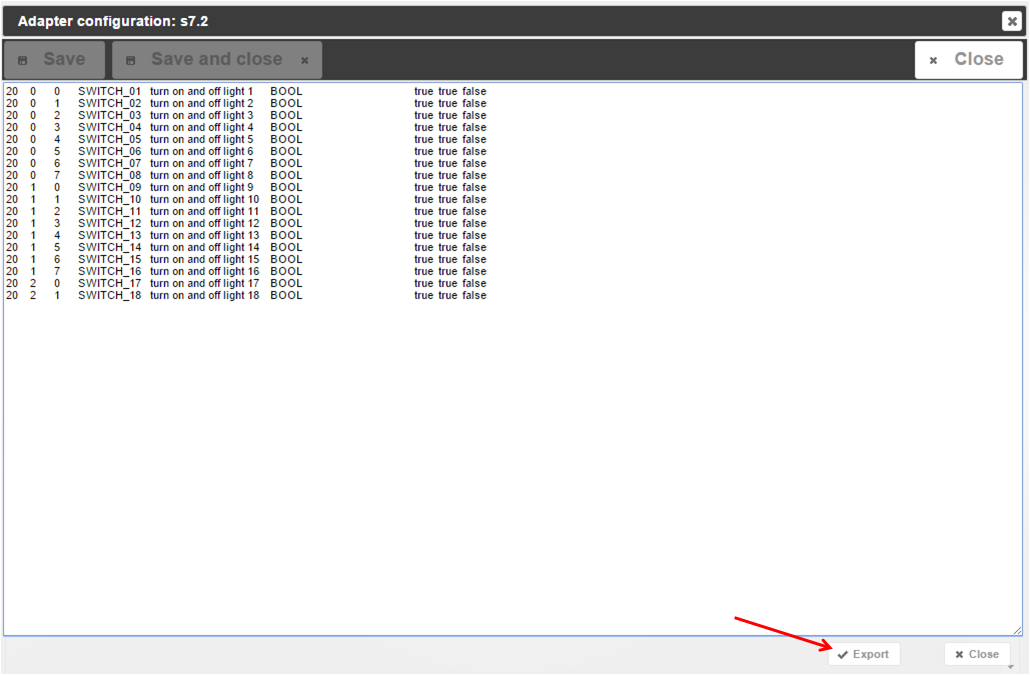


and you get an empty field.

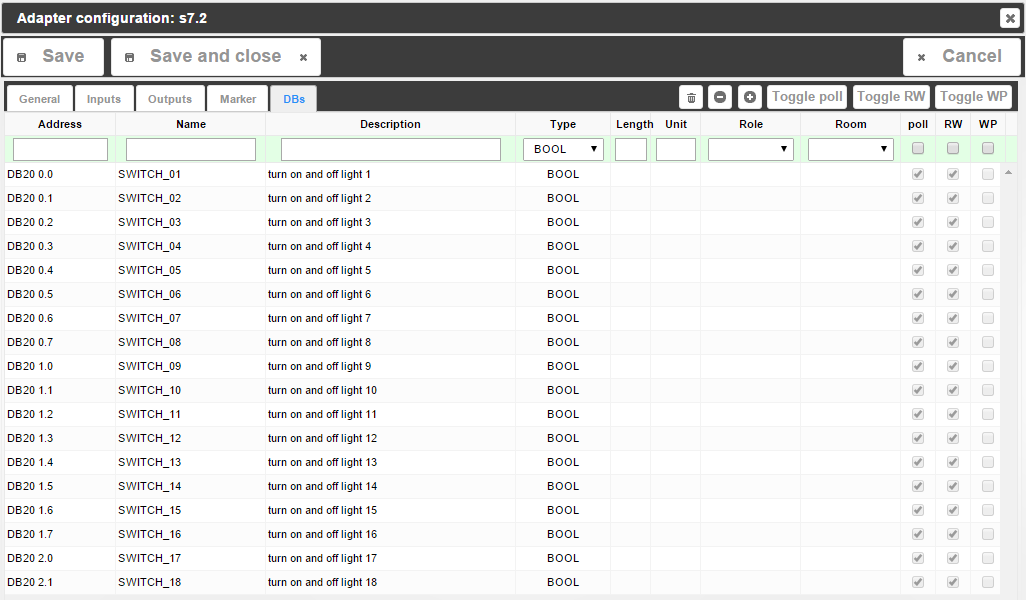
Now go to the spread sheet again, table DB20, and copy column A through M (without headlines).



Paste the cells in the empty import field in ioBroker and confirm with “Export” – which is meant to be called “Import”.

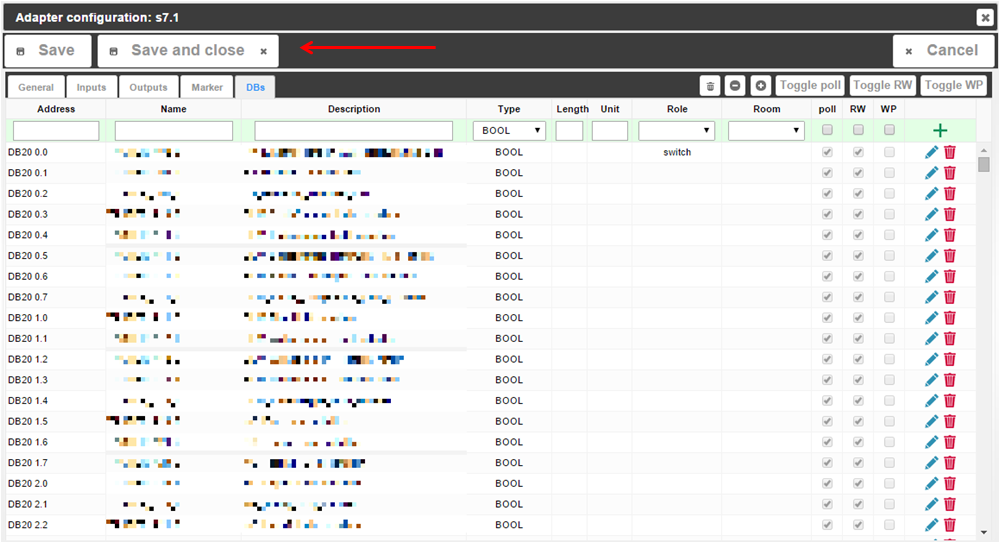


The first DB is done and ready for communication:



Repeat for DB21, DB22, DB23. Each time you click “Import from CSV” you get an empty box, but the contents will be added to the list. You should be done in no time, no matter how many data points you want to populate.

In case you want to make use of the features which come with ioBroker by filling in Length, Unit, Role, Room, you can do that in the spread sheet, too, in order to take advantage of bulk engineering. If you decide to do that later or for a couple of data points only, you can also do that directly in ioBroker under “DBs” with the integrated edit options.

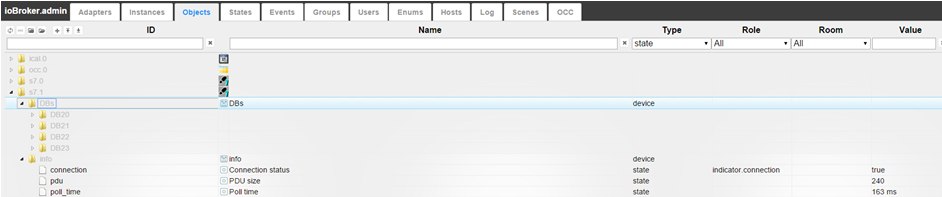
Don’t forget to save, though!

## Communication test

Go to the tab “Objects” in ioBroker and find the S7 instance (e.g. s7.0, not system.adapter.S7.0). If you’re missing anything: F5 (webpage update) is king!

Her you find two groups:

* DBs  
  with the 4 DBs we configured:
  + DB20
  + DB21
  + DB22
  + DB23
* Info  
  with information regarding the connection:
  + Connection: “true” if the S7 can be found on the network
  + pdu: PDU size Snap7 is connected with to the S7 (typically 240 for S7-300, 480 for S7-400)
  + poll\_time: time in milliseconds Snap7 takes for communication - should be lower than the poll delay configured under “General” – “General” in adapter instance configuration.



We have configured DB21 and DB23 as DBs sending information to ioBroker, i.e. if you open the DBs under “Objects”, you should see values coming in already, give that the DBs are being supplied with data from the S7 code.

# Monitoring and Operation in vis

Start ioBroker.vis from the tab “Instances”. I recommend to have the vis-hqwidgets installed.

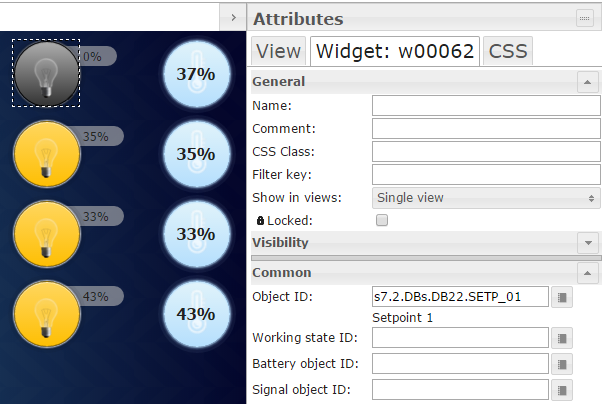
Let’s start with a switch:



Drag& drop a switch widget on your view, connect it to the Object ID of a switch in DB20 and you’re done. If you operate the switch now, you will find that the data point under  
 “Objects” – “s7.x” – “DBs” – “DB20” will toggle and the S7 will turn on and off what ever is connected to the DB. If you monitor the DB in Step7 online, you’ll see that the data point in the DB will change from “0” to “1” etc.

A binary status works exactly the same way: Drag& drop a widget in your view and connect the relevant data point from DB21 to it.

And it is the same again for real values:



Important:

The user is in charge of connecting the correct data points to the widgets. You can connect a real value to a binary status (e.g. light bulb), so the light bulb will show “on” once the real value is >1.0.

That’s all, folks, pretty easy and straight forward, huh?