

How to Connect and Use a Siemens S7 Device

von Jari Krütfeld

Prerequisites

In order to follow along, it would be helpful if you know a bit about:

- [Service Basics Lesson](#)
- Siemens SIMATIC S7 PLCs
- Basic computer programming and networking
- [YAML file format](#)
- [MQTT](#)

Git URL

[Example Project Repo](#)

Introduction

This lesson goes through the required steps to connect and use your Siemens SIMATIC S7 device with the Cybus Connectware. Following this tutorial will enable you to connect and use your own SIMATIC S7 device on the Connectware with ease!

The SIMATIC S7 is a product line of PLCs by Siemens that are widely used in industrial automation. The S7 is capable of connecting several sensors and actuators through digital or analog IOs which can be modular extended.

The read and write access to data on the PLC can be realized through the S7 Communication Services based on ISO-on-TCP (RFC1006). In this case the PLC acts as a server allowing communication partners to access PLC data without the need of projecting the incoming connections during PLC programming. We will use this feature to access the S7 from the Connectware.

Setup

To follow the lesson you need to have a computer (!), a running Cybus Connectware instance and one of the following

a) A Siemens S7 PLC and access to STEP7 (TIA Portal) The S7 PLC needs to be configured using [STEP7](#) in order to work correctly. The following configuration settings on your S7 device are needed:

- To activate the S7 Communication Services you need to enable PUT/GET access in PLC Settings. You should keep in mind that this opens up the controller access by other applications as well.

- To access data from data blocks you need to disable “Optimized Block Access” in data block attributes.

b) **Conpot PLC Emulator** [Conpot](#) can be used to emulate a Siemens S7 PLC.

Writing the Commissioning File

The YAML format based *Commissioning File* tells the Cybus Connectware the type of device to be connected, its connection configuration and specifies the endpoints that should be accessed. Commissioning File details can be found in the [Reference docs](#). For now let’s focus on the three main resources in the file, which are

- `Cybus::Connection`
- `Cybus::Endpoint`
- `Cybus::Mapping`

In the following chapters we will go through the three resources and create an example Commissioning File in which we connect to a S7 device and enable read/write access to a data endpoint.

Cybus::Connection

Inside of the resource section of the commissioning file we describe a connection to a device we want to use. All the information needed for the Connectware to talk to the device is defined here. This information for example includes the protocol to be used, the IP-Address and so on. Our connection resource could look like the following:

```
# -----#
# Connection Resource - S7 Protocol
# -----#
s7Connection:
  type: Cybus::Connection
  properties:
    protocol: S7
    connection:
      host: 192.168.2.60
      port: 102
      rack: 0
      slot: 1
      pollInterval: 1000
```

We define that we want to use the `Cybus::Connection` resource type, which tells the Connectware that we want to create a new device connection. To define what kind of connection we want to use, we are specifying

the `S7` protocol. In order to be able to establish a connection to the device, we need to specify the connection settings as well. Here we want to connect to our S7 device on the given host IP, port, rack and slot number. Furthermore, we specified that the `pollIntervall` for reading the data will be set to one second.

Cybus::Endpoint

We want to access certain data elements on the PLC to either get data from or set data on the device. Similar to the Connection section of the commissioning file, we define an Endpoint section:

```
# -----#
# Endpoint Resource - S7 Protocol
# -----#
s7EndpointDB1000:
  type: Cybus::Endpoint
  properties:
    protocol: S7
    connection: !ref s7Connection
    subscribe:
      address: DB10,X0.0
```

We define that we want to introduce a specific endpoint, which represents a data element on the device, by using the `Cybus::Endpoint` resource. Equally to what we have done in the connection section, we have to define the protocol this endpoint is relying on, namely the `S7` protocol. Every endpoint needs a connection it belongs to, so we add a reference to the earlier created connection by using `!ref` followed by the name of the connection resource. Finally we need to define which access operation we would like to perform on the data element and on which absolute address in memory it is stored at. In this case `subscribe` is used, which will read the data from the device in the interval defined by the referenced connection resource.

The boolean data element which is addressed here is on a Datablock Byte 0 Bit 0, you can learn more about addressing [here](#).

Cybus::Mapping

Now that we can access our data-points on the S7 device we want to map them to a meaningful MQTT topic. Therefore we will use the mapping resource. Here is an example:

```
# -----#
# Mapping Resource - S7 Protocol
# -----#
mapping:
  type: Cybus::Mapping
  properties:
    mappings:
      - subscribe:
          endpoint: !ref s7EndpointDB1000
        publish:
          topic: !sub '${Cybus::MqttRoot}/DB1000'
```

Our example mapping transfers the data from the endpoint to a specified MQTT topic. The important part is where we define from which source we want the data to be transferred to which target. The source is defined by using `subscribe` and setting the `endpoint` to reference the endpoint from the endpoints section above. The target is defined by using `publish` and setting the `topic` to what MQTT topic we want the data to be published on. In this example we are using `!sub` which is similar to `!ref` but substitutes values from somewhere else as a string replacement.

Interim summary

Adding up the three previous sections, a full commissioning file would look like this:

```
---
description: >
  S7 Example

metadata:
  name: "S7 Device"

resources:
# -----#
# Connection Resource - S7 Protocol
# -----#
s7Connection:
  type: Cybus::Connection
  properties:
    protocol: S7
    connection:
```

```

    host: 192.168.2.60
    port: 102
    rack: 0
    slot: 1
    pollInterval: 1000

# -----#
# Endpoint Resource - S7 Protocol
# -----#
s7EndpointDB1000:
  type: Cybus::Endpoint
  properties:
    protocol: S7
    connection: !ref s7Connection
    subscribe:
      address: DB10,X0.0

# -----#
# Mapping Resource - S7 Protocol
# -----#
mapping:
  type: Cybus::Mapping
  properties:
    mappings:
      - subscribe:
          endpoint: !ref s7EndpointDB1000
        publish:
          topic: !sub '${Cybus::MqttRoot}/DB1000'

```

Writing Data

Usually we also want to write data to the device. This can easily be accomplished by defining another endpoint where we use `write` instead of `subscribe`.

```
s7EndpointDB1000Write:
  type: Cybus::Endpoint
  properties:
    protocol: S7
    connection: !ref s7Connection
  write:
    address: DB10,X0.0
```

We also append our mappings to transfer any data from a specific **topic** to the **endpoint** we just defined. mapping:

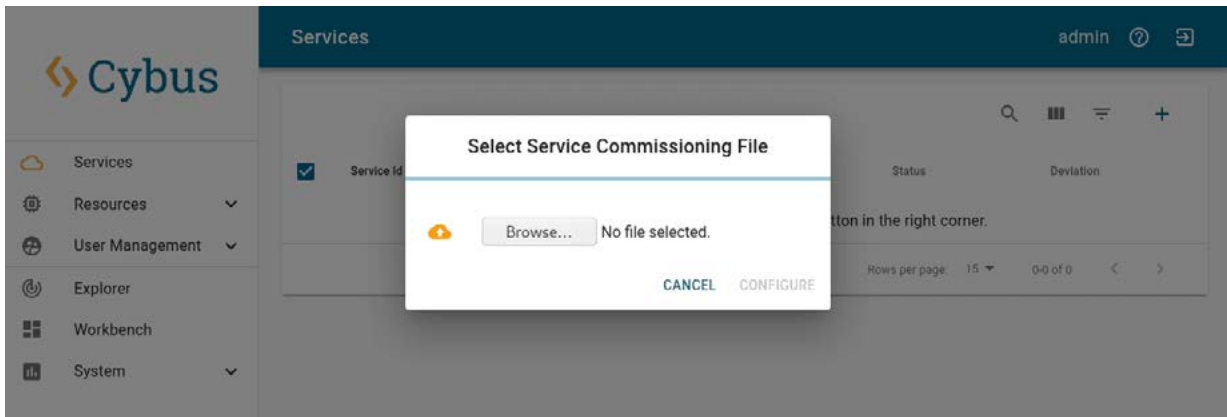
```
type: Cybus::Mapping
properties:
  mappings:
    - subscribe:
        endpoint: !ref s7EndpointDB1000
      publish:
        topic: !sub '${Cybus::MqttRoot}/DB1000'
    - subscribe:
        topic: !sub '${Cybus::MqttRoot}/DB1000/set'
      publish:
        endpoint: !ref s7EndpointDB1000Write
```

To actually write a value, we just have to publish it on the given topic. In our case the topic would be **services/s7device/DB1000/set** and the message has to look like this:

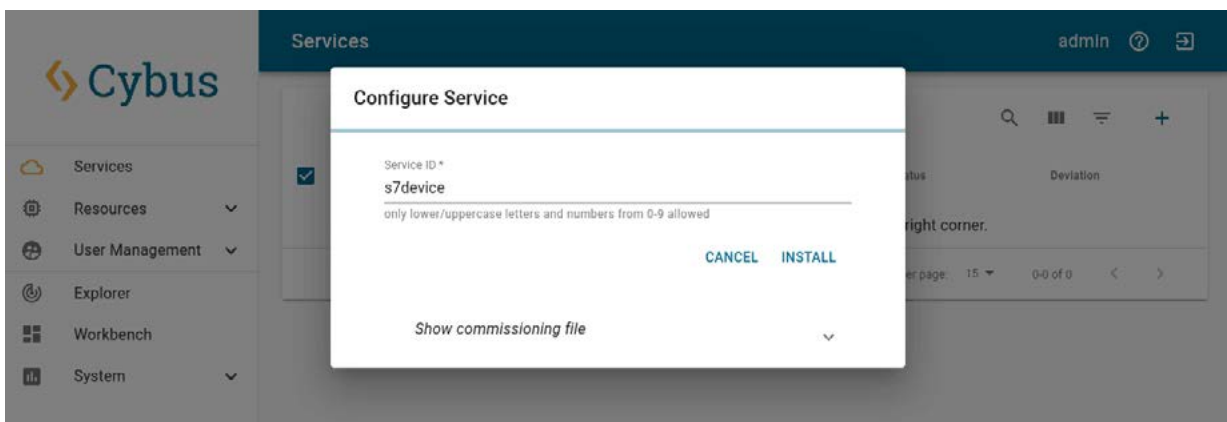
```
{
  "value": true
}
```

Commission the device on the Connectware

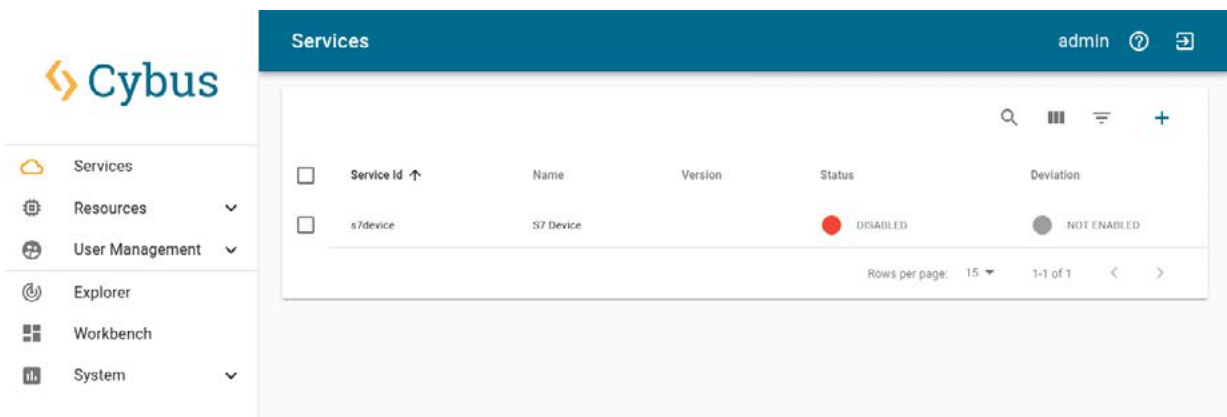
We are finally ready to connect to our Siemens S7 PLC and use it! Go to the **Services** tab in the Connectware, click on the **(+)** button in the upper right corner and choose the Commissioning File that we just created.



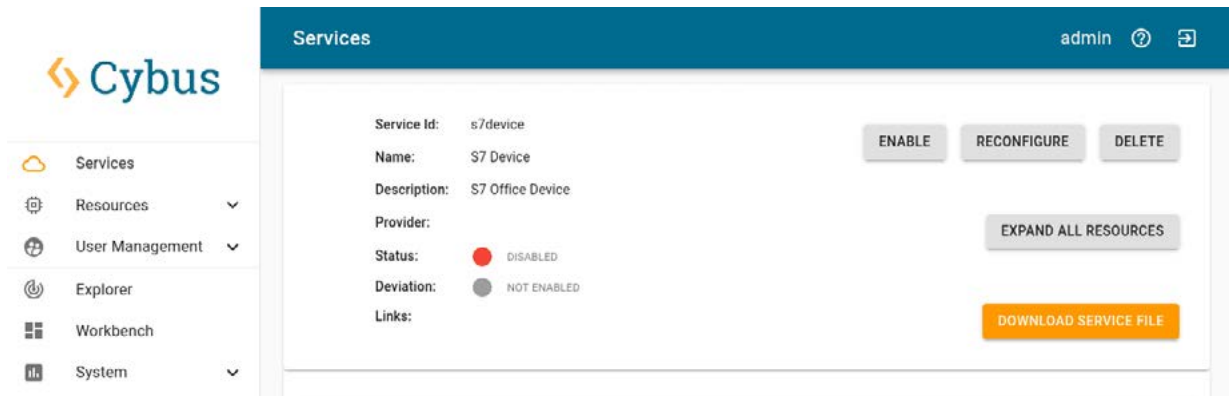
In case you used parameters in the file, you will be prompted now to fill those in.



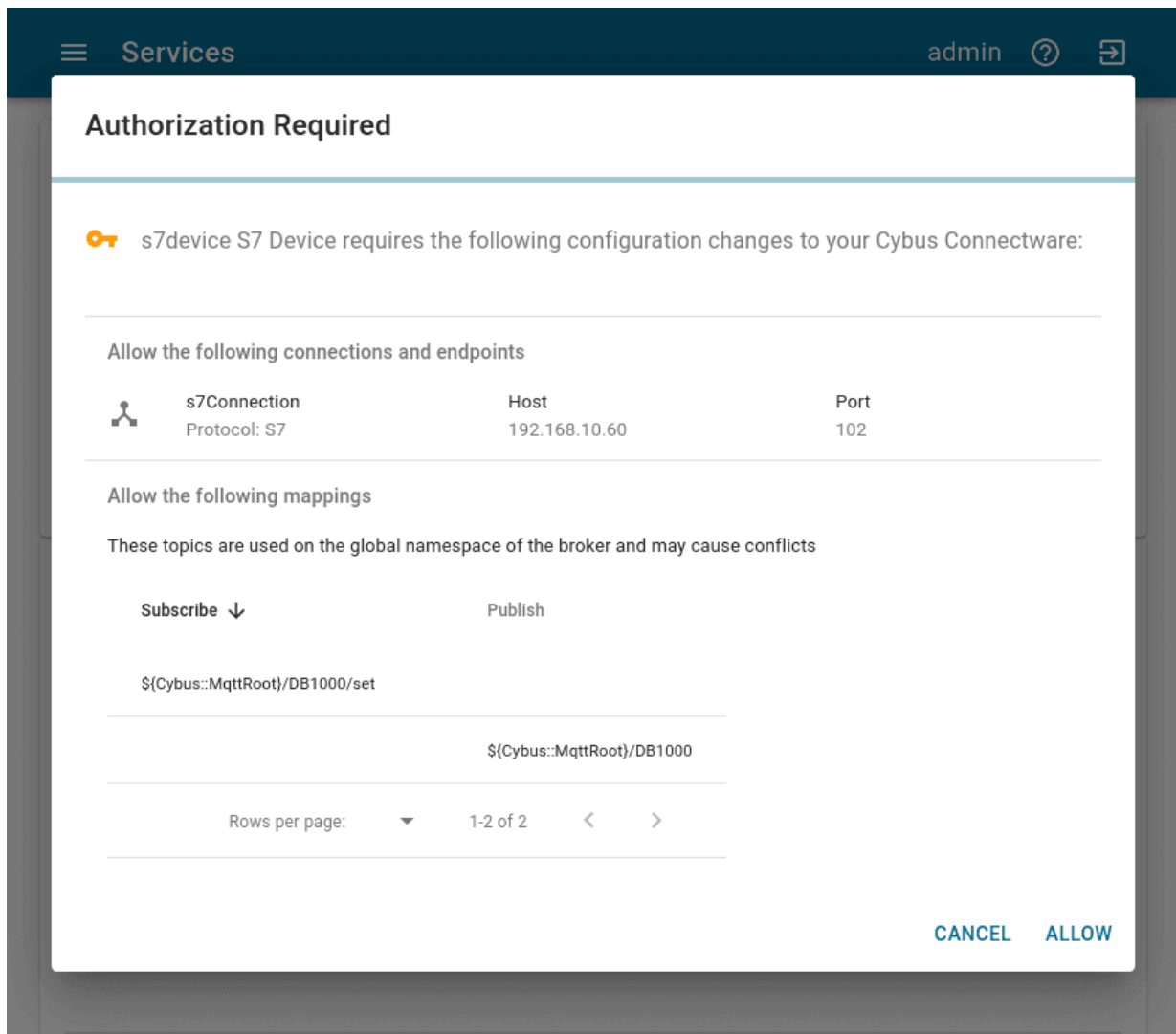
If you are ready, press *Install* and the service will be installed. The status section indicates the health of the service and the resources it defines.



Once the service is installed it needs to be enabled. Therefor click on the service and press *ENABLE* in the top, right corner.



You will be prompted to authorize all permissions the service needs to operate. After going through all of them, click on *ALLOW* to enable the service.



If everything went well, the service should change its status to **ENABLED** (green).

☰ Services
admin ? 🔗

Service Id: s7device

Name: S7 Device

Description: S7 Office Device

Provider:

Status: ● ENABLED

Deviation: ● NO DEVIATION

Links:

DISABLE
RECONFIGURE
DELETE

COLLAPSE ALL RESOURCES

DOWNLOAD SERVICE FILE

SERVICE LOGS
▾

CONTAINERS
0
▴

VOLUMES
0
▴

CONNECTIONS
1
▴

🔍 ☰ ☰

	Name	Protocol	Host	Port	Status
<input type="checkbox"/>	s/Connection	S7	192.168.10.60	102	● CONNECTED

Rows per page: 15 ▾ 1-1 of 1 < >

ENDPOINTS
2
▴

🔍 ☰ ☰

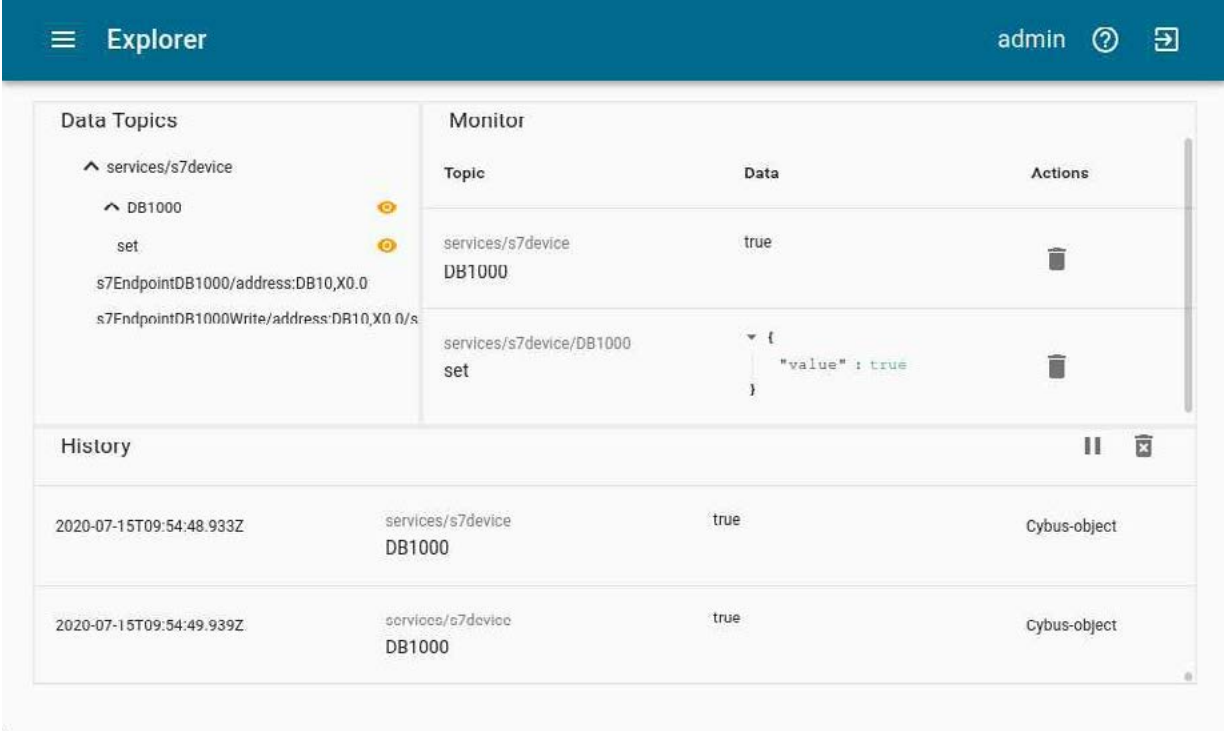
	Name	Connection	Protocol	Operation	Rules	Status
<input type="checkbox"/>	s/EndpointDB1000	s/Connection	S7	subscribe	▶ [] 0 3 items	● ENABLED
<input type="checkbox"/>	s7EndpointDB1000Write	s7Connection	S7	write	▶ [] 0 1 items	● ENABLED

Rows per page: 15 ▾ 1-2 of 2 < >

To see the incoming data go to the **Explorer** Tab in the Connectware and see the MQTT topic we specified in the Commissioning Files.

To watch data flowing simply go to the **Explorer** Tab and activate watching for the topics we are interested in.

To see the value change just publish a value like described in the **Writing Data** section.



The screenshot shows the 'Explorer' interface with a dark blue header containing a menu icon, the title 'Explorer', and user information 'admin' with help and share icons. The main content is divided into two panels: 'Data Topics' on the left and 'Monitor' on the right. The 'Data Topics' panel shows a tree view with expanded folders for 'services/s7device', 'DB1000', and 'set', each with a yellow status icon. Below these are two endpoint addresses: 's7EndpointDB1000/address:DB10,X0.0' and 's7EndpointDB1000Write/address:DB10,X0.0/s'. The 'Monitor' panel contains a table with columns 'Topic', 'Data', and 'Actions'. It lists two topics: 'services/s7device/DB1000' with data 'true' and 'services/s7device/DB1000/set' with data '{ "value" : true }'. Below the monitor is a 'History' section with a pause and close icon, showing two entries with timestamps '2020-07-15T09:54:48.933Z' and '2020-07-15T09:54:49.939Z', both for the 'services/s7device/DB1000' topic with data 'true' and source 'Cybus-object'.

Topic	Data	Actions
services/s7device/DB1000	true	[trash icon]
services/s7device/DB1000/set	{ "value" : true }	[trash icon]

Timestamp	Topic	Data	Source
2020-07-15T09:54:48.933Z	services/s7device/DB1000	true	Cybus-object
2020-07-15T09:54:49.939Z	services/s7device/DB1000	true	Cybus-object

Summary

In this Cybus Learn article we learned how to connect and use a S7 device on the Connectware. See [Example Project Repo](#) for the complete Commissioning File. If you want to keep going and get started with connecting your own S7 device with custom addressing, please visit the [Reference docs](#) to get to know all the Connectware S7 protocol features.

Going further

A good point to go further from here is the [Service Basics Lesson](#), it covers how to use the data from your S7 device.

Disclaimer:

Step7, TIA Portal, S7, S7-1200, Sinamics are trademarks of Siemens AG

Cybus is a specialist for secure IIoT Edge software, headquartered in Germany. Cybus Connectware serves smart factories as a universal Edge and DevOps hub. Machine builders and providers of IIoT services use the Cybus Connectware as a software-based gateway. As early as 2017, Cybus published the first secure industrial connector for machine data according to today's DIN SPEC 27070 standard. Industry analyst Gartner named Cybus a worldwide "Cool Vendor". Today, the company counts medium-sized and large companies from numerous industrial sectors such as mechanical engineering, automotive and aviation among its customers.

Cybus GmbH · Osterstraße 124 · 20255 Hamburg · Germany · www.cybus.io · hello@cybus.io · (+49) 40 228 58 68 51

Summary

In this Cybus Learn article we learned how to connect and use a S7 device on the Connectware. See Example Project Repo for the complete Commissioning File. If you want to keep going and get started with connecting your own S7 device with custom addressing, please visit the Reference docs to get to know all the Connectware S7 protocol features.

Going further

A good point to go further from here is the Service Basics Lesson, it covers how to use the data from your S7 device.

Disclaimer:

Step7, TIA Portal, S7, S7-1200, Sinamics are trademarks of Siemens AG