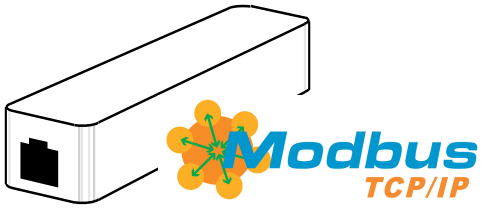
dNode P1

Swedish Standard HAN/P1-port reader for RJ12 interface and DLMS/COSEM protocol translation



Modbus TCP Programmers Guide Version 1.01

This document applies to Novion dNode P1 with firmware build SE-Kxxx

novion technologies

Table of Content

List of Figures

List of Tables

1 Introduction

1.1 General information

This programmers guide provides information required to use the Modbus TCP interface for reading parameters of certain registers of the Novion P1 product.

The Modbus TCP interface can be used for example by energy managers to monitor consumption or generation of the electrical meter it is connected to.

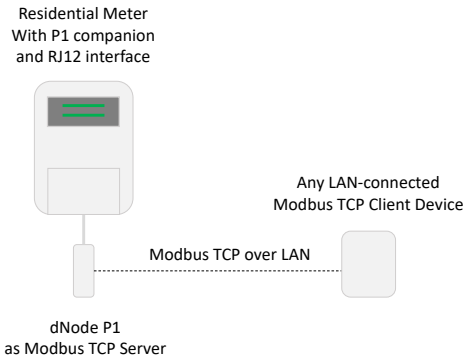


Figure 1 – Schematic view

The Novion P1 SE is compatible with the KEBA P30 EV-charger and connects automatically to the EV-charger if the Novion P1 SE's TCP/IP-address, and port, are configured in the KEBA P30 configuration as an External TCP Meter, KeContact E10.

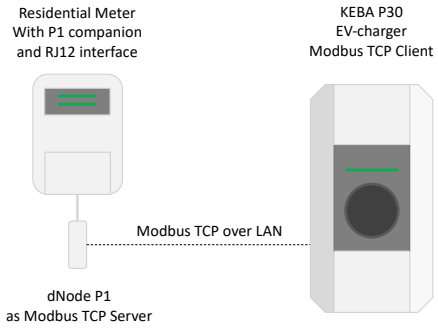


Figure 2 – Schematic view connected to P30

Note! Too many continuously reads, per time unit, from additional client(s) can result in loss of data for the main client, for example P30.

1.2 Safety instructions

This document is an extension to the supplied manuals of Novion dNode P1.
Please see manual for safety instructions.

1.3 Verification of validity

The user must ensure this document as valid for the present product.

1.4 Target group

This document contains information for individuals with technical knowledge and programming skills in the field(s) concerned, likewise being qualified to carry out the necessary operations without technical support.

1.5 Requirements

The following requirements must be met to use the Modbus TCP functionality:

- dNode P1 with firmware build SE-Kxxx
- dNode P1 must be connected to the same Local Area Network as the device used to access it. This includes the KEBA P30 EV-charger
- A device for reading registers via the Modbus TCP interface.
- A suitable client software.

1.6 Legal disclaimer

Specifications are subject to change due to further technical developments. Details presented may be subject to correction.

This programmers guide applies exclusively to dNode P1.

It is possible that the present program guide still has printing defects or printing errors. However, the information in this program guide will be checked regularly and corrections will be made in the next edition.

Liability claims against Novion relating to material or immaterial damage caused by the use or non-use of the information contained in the program guide or using incorrect or incomplete information are excluded.

Novion expressly reserves the right to change, supplement, adapt or delete parts of the program guide without prior notice or to discontinue the entire program guide temporarily or permanently.

Each integrator is responsible for updating and maintaining the applicable system. There are no legal claims and/or liability claims for failure of systems due to non-updated Modbus TCP commands.

1.7 Documents for further reading

Detailed protocol description of the Modbus TCP standards is not given here. Further informations can be found on Internet (e.g. http://www.modbus.org).

2 Modbus Overview

2.1 General information

Modbus is a standardized communication protocol that enables data exchange between a Modbus RTU Master, or a Modbus TCP-client, and several Modbus RTU Slaves, or Modbus TCP Servers. Modbus is part of the IEC 61158 standard. The Modbus protocol enables control of the connected slaves/servers and transmission of measurement data from the slave to the master. The data are sent via Serial RS-485 or TCP/IP. dNode P1 can only use Modbus TCP and can only act as a Server (Slave).

When communicating via Modbus TCP with dNodeP1, the following applies:

- Each participant must have a unique TCP/IP address.
- dNode P1 always use TCP port 502 for Modbus TCP communication.
- The Unit ID of a dNode P1 is always 1.
- Supported function codes are FC3 (Read)
- Starting register address count is 0.
  Depending on the used implementation, +1 might have to be added to address the read the correct register.
- It is not possible to read several registers at once.
  The maximum reading length is 2 words, as the return values for a single register are UINT32.
- The recommended timing intervals for reading registers is maximum 5 sec. For data, which does not change on a frequent basis, higher intervals are recommended.

2.2 Modbus Register Addresses

This chapter describes all readable registers that are supported by dNode P1.

The table below (Table 1) shows all available Modbus Register Addresses and accessible data,

The columns should be read and understood according to the following:

- *Address* is the offset address in the address space 40001 – 49999 (Holding registers). Each address is 1 x Word (uint16).
- *Length* is number of addresses to be read.
- *Source* is from where data is derived. *From meter* = read from the meter P1 companion/interface. *Calculated value* = a value that has been calculated from one or several meter values. *Fix value* is a value that always is the same.
- *P1 ref. (OBIS)* is the corresponding OBIS code matching the Swedish standard P1 interface.
- *Type* is data type for the value of one or several subsequent addresses (derived from *Length*).

- *Scale/Unit* is the factor the presented value is multiplied with to get the unit value. For example, Active Power + (import) has the *Scale/Unit* = 0.1W. This means that a value presented as 20000 equals 20000 * 0.1 = 2000 W.

Table 1 - Modbus addresses

Address	Length	Description	Source	P1 ref. (OBIS)	Type	Scale /Unit
0	2	Active Power + (import)	From meter	1-0.1.7.0	unit32	0.1W
2	2	Active Power - (export)	From meter	1-0.2.7.0	unit32	0.1W
4	2	Reactive Power + (import)	From meter	1-0.3.7.0	unit32	0.1VAr
6	2	Reactive Power - (export9	From meter	1-0.4.7.0	unit32	0.1VAr
16	2	Apparent Power + (import)	Calculated value	na	unit32	0.1VA
18	2	Apparent Power - (export)	Calculated value	na	unit32	0.1VA
24	2	Power Factor	Calculated value	na	unit32	0.001 cos(phi)
26	2	Supply Frequency	Fix value = 50000	na	unit32	0.001Hz
40	2	Active Power L1 + (import)	From meter	1-0.21.7.0	unit32	0.1W
42	2	Active Power L1 - (export)	From meter	1-0.22.7.0	unit32	0.1W
44	2	Reactive Power L1 + (import)	From meter	1-0.23.7.0	unit32	0.1VAr
46	2	Reactive Power L1 - (export)	From meter	1-0.24.7.0	unit32	0.1VAr
56	2	Apparent Power L1 + (import)	Calculated value	na	unit32	0.1VA
58	2	Apparent Power L1 - (export)	Calculated value	na	unit32	0.1VA
60	2	Current L1	From meter	1-0.31.7.0	unit32	0,001A
62	2	Voltage L1	From meter	1-0.32.7.0	unit32	0.001V
64	2	Power Factor L1	Calculated value	na	unit32	0.001 cos(phi)
80	2	Active Power L2 + (import)	From meter	1-0.41.7.0	unit32	0.1W
82	2	Active Power L2 - (export)	From meter	1-0.42.7.0	unit32	0.1W
84	2	Reactive Power L2 + (import)	From meter	1-0.43.7.0	unit32	0.1VAr
86	2	Reactive Power L2 - (export)	From meter	1-0.44.7.0	unit32	0.1VAr
96	2	Apparent Power L2 + (import)	Calculated value	na	unit32	0.1VA
98	2	Apparant Power L2 - (export)	Calculated value	na	unit32	0.1VA
100	2	Current L2	From meter	1-0.51.7.0	unit32	0,001A
102	2	Voltage L2	From meter	1-0.52.7.0	unit32	0.001V
104	2	Power Factor L2	Calculated value	na	unit32	0.001 cos(phi)
120	2	Active Power L3 + (import)	From meter	1-0.61.7.0	unit32	0.1W
122	2	Active Power L3 - (export)	From meter	1-0.62.7.0	unit32	0.1W
124	2	Reactive Power L3 + (import)	From meter	1-0.63.7.0	unit32	0.1VAr
126	2	Reactive Power L3 - (export)	From meter	1-0.64.7.0	unit32	0.1VAr
136	2	Apparent Power L3 + (import)	Calculated value	na	unit32	0.1VA
138	2	Apparant Power L3 - (export)	Calculated value	na	unit32	0.1VA
140	2	Current L3	From meter	1-0.71.7.0	unit32	0,001A
142	2	Voltage L3	From meter	1-0.72.7.0	unit32	0.001V
144	2	Power Factor L3	Calculated value	na	unit32	0.001 cos(phi)
146	2	Minimun Active Power + *3	Fix value = 0	na	unit32	0.1W
256	1	Measurement intervall	Fix value = 0	na	uint64	(unitless)
257	1	CT clamp	Fix value = 0	na	uint64	(unitless)
258	1	Modbus RTU baudrate	Fix value = 0	na	uint64	(unitless)
512	4	Active Energy + (import)	From meter meter	1-0.1.8.0	uint64	0.1Wh
516	4	Active Energy - (export)	From meter meter	1-0.2.8.0	uint64	0.1Wh
520	4	Reactive Energy + (import)	From meter meter	1-0.3.8.0	uint64	0.1VArh
524	4	Reactive Energy - (export)	From meter meter	1-0.4.8.0	uint64	0.1VArh
545	4	Apparent Energy +	Fix value = 0	na	uint64	0.1VAh
549	4	Apparent Energy -	Fix value = 0	na	uint64	0.1VAh
592	4	Active Energy L1 + (import)	Fix value = 0	na	uint64	0.1Wh
596	4	Active Energy L1 - (export)	Fix value = 0	na	uint64	0.1Wh
600	4	Reactive Energy L1 + (import)	Fix value = 0	na	uint64	0.1VArh
604	4	Reactive Energy L1 - (export)	Fix value = 0	na	uint64	0.1VArh
624	4	Apparent Energy L1 +	Fix value = 0	na	uint64	0.1VAh
628	4	Apparent Energy L1 -	Fix value = 0	na	uint64	0.1VAh
672	4	Active Energy L2 + (import)	Fix value = 0	na	uint64	0.1Wh
676	4	Active Energy L2 - (export)	Fix value = 0	na	uint64	0.1Wh
680	4	Reactive Energy L2 + (import)	Fix value = 0	na	uint64	0.1VArh

684	4	Reactive Energy L2 - (export)	Fix value = 0	na	uint64	0.1VArh
704	4	Apparent Energy L2 +	Fix value = 0	na	uint64	0.1VAh
708	4	Apparent Energy L2 -	Fix value = 0	na	uint64	0.1VAh
752	4	Active Energy L3 + (import)	Fix value = 0	na	uint64	0.1Wh
756	4	Active Energy L3 - (export)	Fix value = 0	na	uint64	0.1Wh
760	4	Reactive Energy L3 + (import)	Fix value = 0	na	uint64	0.1VArh
764	4	Reactive Energy L3 - (export)	Fix value = 0	na	uint64	0.1VArh
784	4	Apparent Energy L3 +	Fix value = 0	na	uint64	0.1VAh
788	4	Apparent Energy L3 -	Fix value = 0	na	uint64	0.1VAh
8192	1	Manufacturer ID *	Fix value = [Manufact.ID]	na	unit16	(unitless)
8193	1	Product ID *	Fix value = *[Prod.ID]*	na	unit16	(unitless)
8194	1	Product Version *	Fix value = *[Prod.ver]*	na	unit16	(unitless)
8195	1	Firmware Version *	Fix value = *[Firmw.ver]*	na	unit16	(unitless)
8196	16	Vendor Name *	Fix value = [Vendor Name]	na	String[32]	(unitless)
8212	16	Product Name *	Fix value = *[Prod.Name]*	na	String[32]	(unitless)
8228	16	Serial Number *	Fix value = *[Serial no]*	na	String[32]	(unitless)
8244	1	Measuring Interval	Fix value = 5000	na	unit16	(unitless)

*)

- ManufacturerID is a static value containing the manufacturer's ID. This allows a higher-level system to differentiate between different devices
- ProductID depends on the installed hardware of the product.
- ProductVersion denotes the version of the product's hardware.
- FirmwareVersion indicates the version of the product's software.
- VendorName and ProductName contain the OEM manufacturer's brand name and the product's brand name as strings. All strings are padded to their full length using NULL bytes and spaces (0x32).

2.3 Python Example

Note! This example is using the Python library PyModbus (see https://pypi.org/project/pymodbus/). Newer versions of this library might require updates of the code. This example is tested in Python version 3.8.2

Below is an example of how to read total active power that is currently presented by the residential meter via its P1 interface.

```
# Read total active power that is being consumed
#
# Modbus register: Active Power + (import)
# Holding Address: 0
# Length = 2 (2 x Words)
# Data type = unit32

from pymodbus.constants import Endian
from pymodbus.payload import BinaryPayloadDecoder
from pymodbus.payload import BinaryPayloadBuilder
from pymodbus.client.sync import ModbusTcpClient as ModbusClient
from pymodbus.compat import iteritems
from pymodbus import payload

p1Address = "ip-address or hostname as a string value"
p1Port = 502
p1ModbusAddress = 0
p1ModbusAddressLength = 2
p1ModbusScale = 0.1
client = ModbusClient(p1Address, port = p1Port, timeout = 3)
client.connect()

read = client.read_holding_registers(address=p1ModbusAddress,
count=p1ModbusAddressLength, unit =1)

decoder = payload.BinaryPayloadDecoder.fromRegisters(
              read.registers,
              byteorder = Endian.Big,
              wordorder = Endian.Big
          )

value = decoder.decode_32bit_int() * p1ModbusScale

print("Value = %s kW" % (value / 1000))
```