hPDU Modbus Specification

Document revision: v2.20.r1

CONTEG PDU Modbus Specification

hPDU Modbus Specification

Document revision: v2.20.r1

1. Introduction

This document describes the Modbus/TCP service implemented in CONTEG Hybrid PDUs. The information provided is applicable to firmware versions 2.14 and upwards.

This document **does not** describe the actual Modbus registers available on the device, rather how the CONTEG Data Model (SPDM) document is mapped to Modbus. Please refer to up-to-date SPDM document to see all available registers on the device, which are also accessible over Modbus by use of information provided by this document.

2. Configuration

Modbus/TCP related options can be configured from the **Interfaces** section of hPDU's web interface.

Group	Name	Description
Modbus	modbus TCP	Enables or disables Modbus/TCP service. hPDU mode setting **must** be Hybrid or Bridge.
Modbus	modbus TCP behavior	Read-only or Read-write* access. * Note that Modbus protocol has no authentication or security built-in. It is not recommended to enable Read-Write access except for trusted networks.
Modbus	modbus TCP port	TCP port. Defaults to 502.
Network Configuration	hPDU mode	If set to Classic, Modbus/TCP **cannot** be enabled. If set to Hybrid, Modbus/TCP will only work for the local unit. If set to Bridge, Modbus/TCP will work for local unit as well as remote units on data bus.
Access Control	allowed IP range	IP access restrictions common to all hPDU services

3. Connection

Default Modbus port is 502 as per Modbus/TCP specification.

hPDU supports multiple consecutive Modbus request-responses on the same TCP connection. But it will keep only a single TCP connection active at a time. If a new connection comes in while the previous one is IDLE, the previous connection will be dropped to favor the new connection.

4. Addressing Devices

To address any device on the data bus other than the unit itself, hPDU mode must be set to Bridge.

PDU data bus unit addresses are used as Modbus addresses. Modbus/TCP specification limits addressable devices to be in the range 0 to 247. Thus, only devices configured to have addresses 1 to 247 would be accessible by Modbus.

To address the device itself, device's own address, as well as address 0 (or less preferably 255) can be used.

5. CONTEG Data Model (SPDM)

Modbus implementation exposes all registers available on CONTEG Data Model, SPDM.

SPDM divides the registers into groups. An example group, "identification" is displayed below. Please see the original document for full list of available registers.

group	register	ext	mnemonic	name	datatype	bytes	repeats	size	access
identification	100		idspdm	SPDMVersion	int	2	1	2	ro
identification	102		idfwvs	firmwareVersion	int	2	1	2	ro
identification	104		idonbr	salesOrderNumber	ascii	16	1	16	rw
identification	120		idpart	productId	ascii	16	1	16	rw
identification	136		idsnbr	serialNumber	ascii	16	1	16	rw
identification	152		idchip	hardwareAddress	int	2	3	6	ro
identification	158		idaddr	unitAddress	int	2	1	2	rw

SPDM exposes device's internal information as a memory mapping. Each register in SPDM has a starting address, mentioned in **register** column. For example, the firmware version of a device starts at address 102, and the unique hardware address starts at address 152.

A register can have multiple channels. An example is measurements of multiple input phases. **repeats** column shows how many channels a register has.

Each channel of a register is **bytes** register numbers long. For example, **hardwareAddress** register starts at address 152 (**register**), it has 3 channels (**repeats**) and each channel is 2 bytes long (**bytes**); which means its individual channels start at addresses 152, 154 and 156.

Datatype show the type of the value stored in a single channel. It should be used together with **bytes** column.

For the purposes of Modbus, details of individual SPDM data types are irrelevant. For Modbus, a mapping from SPDM to Modbus data types is performed, which is explained in the next section.

Each register also has access permissions, where **ro** denotes read-only, **wo** denotes write-only, **rw** denotes read-write.

5.1. Extension layer

Some rows in SPDM have an extension layer. This is shown in the **ext** column of SPDM table. An example is outlet rows, some of which are depicted below.

group	register	ext	mnemonic	name	datatype	bytes	repeats	size	access
output_measures	4000	•	omkwht	kWhTotal	int	3	27	81	ro
output_measures	4081	•	omkwhs	kWhSubtotal	int	3	27	81	ro
output_measures	4162	•	ompfac	powerFactor	fd	2	27	54	ro

When a row has an extension layer, this means that that row actually has twice the number of channels, which need to be addressed separately. For example, the kWhTotal row spans between 4000-4081, each channel is 3 bytes long, having 27 channels addressable in this range (first channel is at 4000, 27th channel is at 4078).

This row also has an extension layer, which has the same address range (4000-4081), but this extension layer holds the channels 28-54 (in case the device has more than 27 metered outlets). 28th channel is at address 4000 in extension layer, 54th channel is at address 4078 in extension layer.

Accessing data in the extension layer will be discussed separately in the following sections.

6. Modbus Data Types

This section explains the conversion between SPDM document data types and Modbus data types.

For Modbus, the minimum accessible data unit is a single SPDM channel. Smaller units than a single channel cannot be read or written. Multiple SPDM channels, even multiple SPDM rows can be read or

SPDM Type	Modbus Type	Details
int, 1	1 register, integer	
int, 2	1 register, integer	
int, 3	2 consecutive registers, 32 bit integer*	Both registers must be accessed together.
int, 4	2 consecutive registers, 32 bit integer*	Both registers must be accessed together.
fd, 2	2 consecutive registers, 32 bit float*	Both registers must be accessed together.
ascii, even sized	(size/2) registers, string*	All registers must be accessed together.

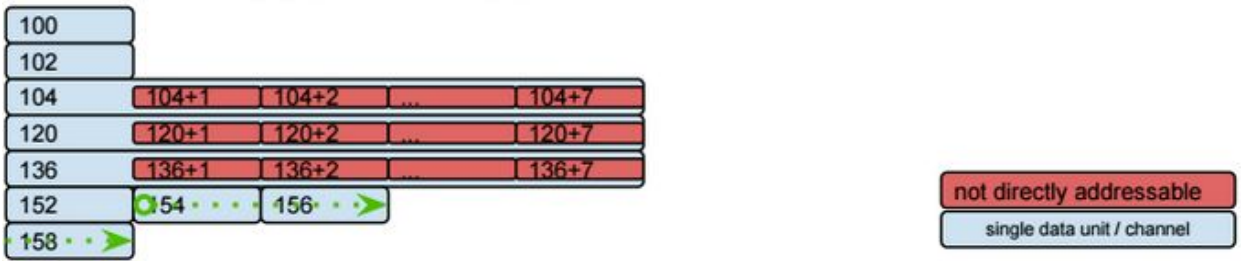
* According to Modbus specification, Modbus only supports 16-bit integer registers, and some conventions are used to represent larger data. 32-bit integers are transferred as 2 registers with least significant word first. 32-bit floats are transferred as 2 registers in IEEE floating point format. Strings are transferred as 2 bytes within each Modbus register.

7. Modbus Register Mapping

Each SPDM register is mapped to the same address on Modbus. For example, firmwareVersion register that starts at address 102 in SPDM can be read by reading the Modbus register at 102. Because the type of firmwareVersion is “int,2”, it maps to a single Modbus register. A single register must be read from address 102.

The following diagram depicts the layout of Modbus registers for the identification SPDM group, whose table is provided above. According to the Modbus Data Types table, channels starting at 100,

102, 152, 154, 156 and 158 are 1 Modbus register long (they all of type “int,2”). Channels starting at 104, 120, 136 are 8 Modbus registers long (they are all of type “ascii,16”).



Blue boxes show starting Modbus addresses of individual channels (of SPDM). Red boxes are not directly accessible. Not-mentioned numbers (such as 101 or 103) do not exist. These will be clarified in the following subsections.

7.1. Accessing single channels

In hPDU, each channel of a SPDM register is mapped to Modbus by its starting address. For example, in SPDM, 3 channels of hardwareAddress start at positions 152, 154 and 156. They are all single Modbus registers due to their SPDM type “int,2”. To read only the second channel of hardwareAddress using Modbus, a single register at address 154 must be read.

In the diagram depicted above, any of the blue boxes can be read by their starting address.

Note that each SPDM channel **must** be read as a whole. Partial reads are not possible.

For example registers 104, 120 and 136 (salesOrderNumber, productId, serialNumber; which are all “ascii, 16” SPDM types) are 8 Modbus registers long (due to their SPDM types). Thus they **must** be read with a **single** Read command of 8 Modbus registers starting at 104 (or at 120, or at 136).

7.2. Accessing multiple channels of the same row

In hPDU, Modbus registers do not have gaps between them, even if their starting addresses might say otherwise. Starting from a valid starting address, and reading multiple registers will result in all following channels in SPDM to be read.

For example; Even if individual channels of hardwareAddress row can be read as single Modbus registers at 152, 154 and 156 ; **there are no gaps at Modbus positions 153, 155, or 157 -- all 3 channels of hardwareAddress are adjacent Modbus registers (see diagram). A 3 register read starting from address 152 will return 3 Modbus registers representing channel 1, channel 2 and channel 3 of hardwareAddress.**

Note that this behavior is different than CONTEG Gateway's Modbus/TCP implementation.

7.3. Accessing multiple rows

Multiple rows can be accessed in the same way as accessing multiple channels. All rows are adjacent to each other, as well as their channels.

For example, according to the SPDM document, SPDMVersion row starts at 100, firmwareVersion row starts at 102, and they are consecutive. According to Modbus Data Types table, both of them are single Modbus registers. Thus, to read these two rows together, a read of 2 Modbus registers starting at address 100 is sufficient.

Consistent with the previous subsections, in the diagram above, it can be seen that there is no Modbus register 101; and 100 and 102 are consecutive, single Modbus registers.

Additionally, it is possible to start reading at any channel of a row, and continue reading on the next row. Depicted as a green arrow on the diagram, a single Read of 3 registers starting at address 154 (2nd channel of hardwareAddress) will return Modbus registers at 154, 156 and 158 (2nd and 3rd channels of hardwareAddress, and unitAddress respectively).

7.4. Accessing extension layers

For rows that have an extension layer in SPDM, there are 3 different ways to access data in extension layers. Any of these methods may be used interchangeably depending on client application's requirements.

7.4.1. As a continuation of base layer

Logically, channels in extension layer lie after the channels in base layer. Although they cannot be addressed independently, a multi channel read command starting from base layer can continue on to channels in extension layer, as they are consecutive.

The diagram above depicts the Modbus registers of SPDM rows kWhTotal (4000), kWhSubtotal (4081), and powerFactor (4162). Channels of these rows are of type "int,3" or "fd,2" which makes them into 2 Modbus registers each. Channels 28 to 54 lie on the extension layer, thus not accessible

by starting addresses. Still, a **single** Modbus read of 106 registers (53 channels) starting at register 4164 (2nd channel of powerFactor)) will return both base layer and extension layer channels.

** Note that this method cannot be used on CONTEG Gateway's Modbus implementation.*

7.4.2. Directly addressed with an offset of 10000

If the application requires direct accessing of individual channels in extension layer, an offset of 10000 can be added to the base layer's channels. For example, if channel 1 of kWhTotal lies at 4000, and channel 2 lies at 4002 ; reading 2 Modbus registers at address 14000 will return channel 28, and reading at address 14002 will return channel 29.

** This method is compatible with CONTEG Gateway's Modbus implementation.*

7.4.2. Directly addressed with Read Input Registers (04) function code

In normal cases, Read Holding Registers (03) Modbus function is used to read data from the device. If, instead, Read Input Registers (04) is used, PDU will return data in extension layers wherever possible.

For example, if 2 registers at address 4000 are read using Modbus function 03 ; channel 1 of kWhTotal (base layer) will be returned. If the same range (4000,2) is read using Modbus function 04 ; channel 28 of kWhTotal (extension layer) will be returned.

For non-extension rows, both read function codes (03 and 04) behave the same.

Note that this method cannot be used for writing.

** This method is compatible with CONTEG Gateway's Modbus implementation.*

8. Supported Function Codes

CONTEG Hybrid PDU supports the following Modbus function codes. For their details, please consult Modbus protocol documentation.

8.1. Read Holding Registers (03)

Will read registers. Only entire channels can be read -- partial reads are not possible. For register addressing and size of registers, refer to previous sections.

8.2. Read Input Registers (04)

For non-extension SPDM registers, this will act exactly the same as function Read Holding Registers (03). For SPDM registers with extensions, extension layer's data will be returned in base layer's address ranges.

8.3. Write Single Register (06)

This function is used to write a single-length Modbus register. Addressing is same as Read Holding Registers (03).

Note that, partial access to SPDM channels is not possible, and this function can only specify a single Modbus register -- thus cannot be used for types larger than 1 Modbus register (for example, "fd,2" types which map to 2 Modbus registers).

8.4. Write Multiple Registers (16, 0x10)

This function may be used to write multiple Modbus registers at once. Addressing is same as Read Holding Registers (03).

This function is the only way to write to larger-than-1-register types (such as "fd,2" types which maps to 2 Modbus registers, or strings) ; as well as to write simultaneously to multiple channels at once.

9. Examples

	SPDM mnemonic	**Func Code**	**Start**	**Length in regs**	**Result**
Read FW version	idfwvs	03	102	1	1 register
Read Input 1 Voltage	imvoac	03	3036	2	2 registers representing voltage as 32-bit float
Read Input 1-3 RMS Current	imcrac	03	3024	6	6 registers, representing 3x 32-bit floats in pairs
Read Outlet 2's voltage	omvoac	03	4326	2	2 registers, representing 1x 32-bit float

Read Outlet **29**'s voltage	omvoac	**04**	4326	2	2 registers, representing 1x 32-bit float
Read Outlet **29**'s voltage	omvoac	03	**14326**	2	2 registers, representing 1x 32-bit float