# AXL F RS UNI 1H

**Axioline F communication module for serial data transmission, 1 interface can be parameterized as RS–485/RS-422 or RS–232**

Data sheet
8533_en_02_C01

## 1    Description

The module is designed for use within an Axioline F station.

It is used to operate standard I/O devices with serial interfaces on a bus system.

**Features**

– A serial input and output channel in RS-232, RS-422 and RS-485 format
– Various protocols supported
– Transmission speed can be set up to 250000 baud
– Parameterization via the PDI channel
– Device type label stored
– Diagnostic and status indicators



This data sheet is only valid in association with the UM EN AXL SYS INST user manual.



Make sure you always use the latest documentation.
It can be downloaded from the product at phoenixcontact.net/products.

## 2 Table of contents

# 3    Ordering data

| Description | Type | Order No. | Pcs. / Pkt. |
| --- | --- | --- | --- |
| Axioline F communication module for serial data transmission, 1 interface can be parameterized as RS–485/RS-422 or RS–232 (including bus base module and plugs) | AXL F RS UNI 1H | 2688666 | 1 |

| Accessories | Type | Order No. | Pcs. / Pkt. |
| --- | --- | --- | --- |
| Axioline F bus base module for housing type H (Replacement item) | AXL F BS H | 2700992 | 5 |
| Axioline plug set (e.g., for AXL DI 16/1) (Replacement item) | AXL CNS 2L-O/D/UI/E1/E2 | 2700985 | 1 |
| Zack marker strip for Axioline (device labeling), in 2 x 20.3 mm pitch, un-printed, 25-section, for individual labeling with B-STIFT 0.8, X-PEN, or CMS-P1-PLOTTER (Marking) | ZB 20,3 AXL UNPRINTED | 0829579 | 25 |
| Zack marker strip, flat, in 10 mm pitch, unprinted, 10-section, for individual labeling with M-PEN 0,8, X-PEN, or CMS-P1-PLOTTER (Marking) | ZBF 10/5,8 AXL UNPRINTED | 0829580 | 50 |
| Axioline shield connection set (contains 2 busbar holders and 2 SK 5 shield connection clamps) | AXL SHIELD SET | 2700518 | 1 |
| Insert label, Roll, white, Unlabeled, Can be labeled with: Thermomark R, Thermomark X,  Thermomark S, Mounting type: Snapped into marker car-rier, Lettering field: 35 x 28 mm (Marking) | EMT (35X28)R | 0801602 | 1 |

| Documentation | Type | Order No. | Pcs. / Pkt. |
| --- | --- | --- | --- |
| User manual, English, Axioline: System and installation | UM EN AXL SYS INST | - | - |

# 4    Technical data

**Dimensions (nominal sizes in mm)**
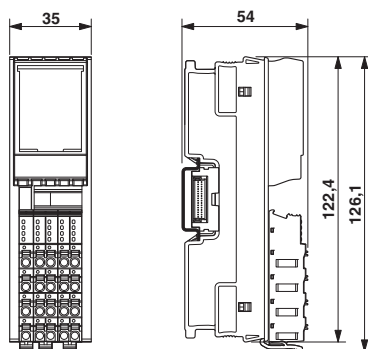


| Width | 35 mm |
| --- | --- |
| Height | 126.1 mm |
| Depth | 54 mm |
| Note on dimensions | The depth is valid when a TH 35-7.5 DIN rail is used (according to EN 60715). |

**General data**

| Color | Military gray RAL 7042 |
| --- | --- |
| Weight | 135 g |
| Mounting type | DIN rail |
| Ambient temperature (operation) | -25 °C ... 60 °C |
| Ambient temperature (storage/transport) | -40 °C ... 85 °C |
| Permissible humidity (operation) | 5 % ... 95 % (no condensation) |

## General data

| | |
|---|---|
| Permissible humidity (storage/transport) | 5 % ... 95 % (no condensation) |
| Air pressure (operation) | 70 kPa ... 106 kPa (up to 3000 m above sea level) |
| Air pressure (storage/transport) | 70 kPa ... 106 kPa (up to 3000 m above sea level) |
| Degree of protection | IP20 |
| Protection class | III, IEC 61140, EN 61140, VDE 0140-1 |

## Connection data

| | |
|---|---|
| Name | Axioline plug |
| Connection method | Push-in technology |
| Conductor cross section solid / stranded | 0.2 mm$^2$ ... 1.5 mm$^2$ |
| Conductor cross section [AWG] | 24 ... 16 |

## Interface Axioline F local bus

| | |
|---|---|
| Connection method | Bus base module |
| Transmission speed | 100 MBit/s |

## Interface RS-232, RS-485, RS-422

| | |
|---|---|
| Connection method | Push-in technology |
| Transmission speed | 110 Bit/s ... 250 kBit/s (can be parameterized) |
| Transmission physics | Copper |
| Protocols supported | Transparent, end-to-end, XON/XOFF, Modbus RTU (master support) |
| Input buffer | 4 kByte |
| Output buffer | 1 kByte |
| Data bits | 5 … 8 |
| Stop bits | 1 or 2 |

## Communications power

ℹ️ The I/O is also supplied from the communications power $U_{Bus}$.
The I/O supply is electrically isolated from the local bus.

| | |
|---|---|
| Communications power $U_{bus}$ | 5 V DC (via bus base module) |
| Current consumption from $U_{bus}$ | typ. 200 mA, max. 240 mA |

## PROFIBUS telegram data

| | |
|---|---|
| Required parameter data | 14 Byte |
| Need for configuration data | 7 Byte |

## Electrical isolation/isolation of the voltage areas

| | |
|---|---|
| 5 V supply (logic)/functional earth ground | 500 V AC, 50 Hz, 1 min |
| 5 V supply (logic)/5 V supply (I/O) | 500 V AC, 50 Hz, 1 min |
| 5 V supply (I/O)/functional earth ground | 500 V AC, 50 Hz, 1 min |

## Mechanical tests

| | |
|---|---|
| Vibration resistance in acc. with IEC 60068-2-6 | 5 g |
| Shock test in acc. with IEC 60068-2-27 | 30g, 11 ms period, half-sine shock pulse |
| Bump endurance test according to EN 60068-2-27 | 10 g |

## Conformance with EMC Directive 2004/108/EC

**Noise immunity test in accordance with EN 61000-6-2**

| | |
|---|---|
| Electrostatic discharge (ESD) EN 61000−4−2/IEC 61000−4−2 | Criterion B; 6 kV contact discharge, 8 kV air discharge |
| Electromagnetic fields EN 61000−4−3/IEC 61000−4−3 | Criterion A; Field intensity: 10 V/m |
| Fast transients (burst) EN 61000−4−4/IEC 61000−4−4 | Criterion B, 2 kV |
| Transient surge voltage (surge) EN 61000−4−5/IEC 61000−4−5 | Criterion B; DC supply lines: ±0.5 kV/±0.5 kV (symmetrical/asymmetrical) |
| Conducted interference EN 61000−4−6/IEC 61000−4−6 | Criterion A; Test voltage 10 V |

**Noise emission test according to EN 61000-6-3**

| | |
|---|---|
| Radio interference properties EN 55022 | Class B |

## Error messages to the higher level control or computer system

| | |
|---|---|
| I/O supply failure | Yes |

## Approvals

For the latest approvals, please visit phoenixcontact.net/products.

# 5 Internal circuit diagram



Figure 1     Internal wiring of the terminal points
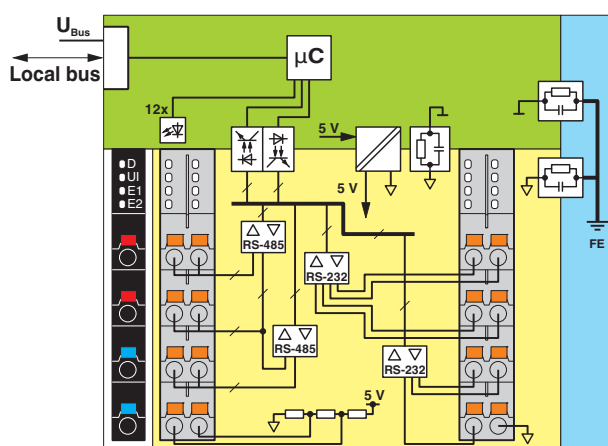
Key:

| | |
|---|---|
| Local bus | Axioline F local bus (hereinafter referred to as local bus) |
| µC | Microcontroller |
| | Optocoupler |
| | Diagnostic and status indicators |
| | RS-485/422/232 interface |
| | DC/DC converter with electrical isolation |
| | Coupling network |
| | Electrically isolated area |
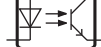
# 6 Terminal point assignment
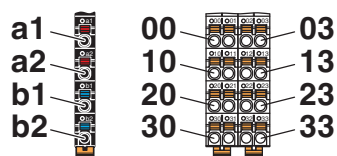


Figure 2    Terminal point assignment

| Plug 1 | | |
|---|---|---|
| **Terminal point** | **Color** | **Assignment** |
| a1, a2 | Red | Not used (bridged internally) |
| b1, b2 | Blue | Not used (bridged internally) |

You can use plug 1 for potential routing of the 24 V supply voltage, however the voltage is not used by the module.

| Plug 2 | | | RS-485 | RS-422 | Notes |
|---|---|---|---|---|---|
| **Terminal point** | **Color** | **Signal** | **Description** | **Description** | |
| 00 | Orange | TxD+ | Not used | Transmit data | positive |
| 10 | Orange | RxD+ | Transmit/receive data | Receive data | positive |
| 20 | Orange | RxD+ | Transmit/receive data | Receive data | Positive; for external bridge to 30 |
| 30 | Orange | R+ | Termination resistor | Termination resistor | Positive pole; for external bridge to 20 |
| 01 | Orange | TxD- | Not used | Transmit data | negative |
| 11 | Orange | RxD- | Transmit/receive data | Receive data | negative |
| 21 | Orange | RxD- | Transmit/receive data | Receive data | Negative; for external bridge to 31 |
| 31 | Orange | R- | Termination resistor | Termination resistor | Negative pole; for external bridge to 21 |

| Plug 3: RS-232 | | | |
|---|---|---|---|
| **Terminal point** | **Color** | **Signal** | **Description** | |
| 02 | Orange | RxD | Serial data input | |
| 12 | Orange | RTS | Request to send | Request to send; handshake signal; output |
| 22 | Orange | DTR | Data terminal ready | Startup request to the connected device; handshake signal; output |
| 32 | Orange | DCD | Data carrier detect | Connected device ready to operate; handshake signal; input |
| 03 | Orange | TxD | Serial data output | |
| 13 | Orange | CTS | Clear to send | Permission to send; connected device ready to receive; handshake signal; input |
| 23 | Orange | DSR | Data set ready | Connected device ready to operate; handshake signal; input |
| 33 | Orange | GND | Ground for the serial interface | |

# 7 Connection notes

– Connect the shield to a busbar before the module
– Fit the receive signals of the RS-485 or RS-422 network with a termination resistor at the relevant end point.
– If you use the integrated termination resistor, the polarization of the data cable will also be active.

# 8 Connection examples

| Operating mode | Special feature | Notes | Image |
|---|---|---|---|
| RS-485 | Module as the network end point | Termination resistor required | Figure 3 |
| RS-485 | Module in the network center | | Figure 4 |
| RS-422 | Module is last receiver | Termination resistor required | Figure 5 |
| RS-232 | Four-wire handshake | | Figure 6 |
| RS-232 | Without handshake | | Figure 7 |

## 8.1 RS-485: module as the network end point
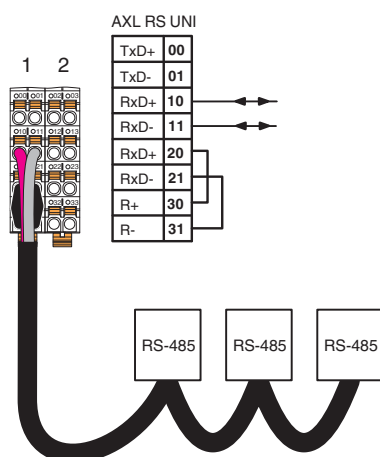


Figure 3    RS-485 interface wiring:
Module as the network end point

## 8.2 RS-485: module in the network center
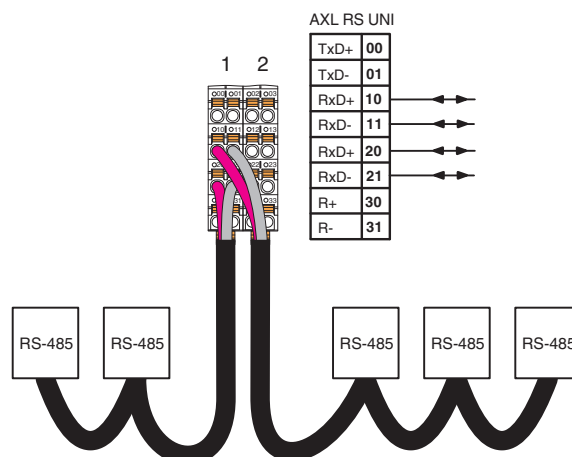


Figure 4    RS-485 interface wiring:
Module in the network center

### 8.3 RS-422
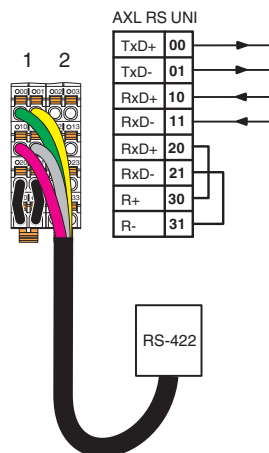


Figure 5          RS-422 or RS-485 interface wiring, full duplex

### 8.4 RS-232: With 4-wire handshake



Figure 6          RS-232 interface wiring: four-wire handshake

### 8.5 RS-232: Without handshake



Figure 7          RS-232 interface wiring: without handshake

For wiring without handshake, insert a bridge between terminal points 22 (DTR) and 13 (CTS).

# 9 Local status and diagnostic indicators
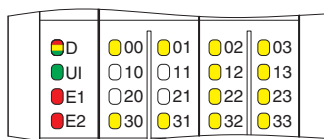


Figure 8     Local status and diagnostic indicators

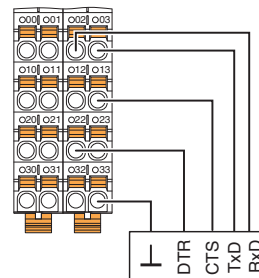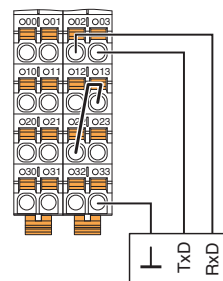| Plug 1 | | | | |
|---|---|---|---|---|
| **Designa-tion** | **Color** | **Meaning** | **State** | **Description** |
| D | Red/yel-low/green | Diagnostics of local bus communication | | |
| | | Power down | OFF | Device in (power) reset. |
| | | Not connected | Red flash-ing | Device operating, but there is no connection to previous device. |
| | | Reset | Red ON | Application reset<br>Device operating, but there is still a connection to the previous de-vice, the application is reset. |
| | | Ready | Yellow ON | Device operating, there is still a connection to the previous device, but the device has not yet detected a valid cycle after power on. |
| | | Connected | Yellow flashing | Valid data cycles have been detected, but the device is (not) yet part of the current configuration. |
| | | Device applica-tion not active | Green/yel-low alter-nating | Valid data cycles are being detected.<br>The master application set the output data to valid, however, the slave application has not set the input data to valid as yet. |
| | | Active | Green flashing | Device operating, communications within the station is OK.<br>The master application does not read the input data.<br>(The connection to the controller has not yet been established, for example.) |
| | | Run | Green ON | Valid data cycles are being detected. All data is valid |
| UI | Green | I/O voltage | ON | I/O supply voltage is present (generated from $U_L$) |
| | | | OFF | I/O supply voltage is not present (generated from $U_L$) |
| E1 | Red | Peripheral fault | ON | I/O error present. |
| | | | OFF | No I/O error. |
| E2 | Red | Channel error | ON | Channel error present. |
| | | | OFF | Channel error not present. |

| Plug 2: RS-485/422 | | | | |
|---|---|---|---|---|
| **Designation** | | **Color** | **State** | **Description** |
| 00 | TxD | Yellow | ON | Module is transmitting data to the connected device |
| | | | OFF | Module is not transmitting data |
| 10 | - | - | - | Not used |
| 20 | - | - | - | Not used |
| 30 | RS-485 | Yellow | ON | Module is parameterized for RS-485 |
| | | | OFF | Module is not parameterized for RS-485 |
| 01 | RxD | Yellow | ON | Module is receiving data from the connected device |
| | | | OFF | Module is not receiving data |
| 11 | - | - | - | Not used |
| 21 | - | - | - | Not used |
| 31 | RS-422 | Yellow | ON | Module is parameterized for RS-422 |
| | | | OFF | Module is not parameterized for RS-422 |

| Plug 3: RS-232 | | | | |
|---|---|---|---|---|
| **Designation** | | **Color** | **State** | **Description** |
| 02 | RxD | Yellow | ON | Module is receiving data from the connected device |
| | | | OFF | Module is not receiving data |
| 12 | RTS | Yellow | | Request to send |
| | | | ON | Handshake signal is set by the module |
| | | | OFF | Handshake signal is not set |
| 22 | DTR | Yellow | | Data terminal ready |
| | | | ON | Handshake signal is set by the module |
| | | | OFF | Handshake signal is not set |
| 32 | DCD | Yellow | | Data carrier detect |
| | | | ON | Handshake signal is set by partner |
| | | | OFF | Handshake signal is not set |
| 03 | TxD | Yellow | ON | Module is transmitting data to the connected device |
| | | | OFF | Module is not transmitting data |
| 13 | CTS | Yellow | | Clear to send |
| | | | ON | Handshake signal is set by partner |
| | | | OFF | Handshake signal is not set |
| 23 | DSR | Yellow | ON | Data set ready |
| | | | ON | Handshake signal is set by partner |
| | | | OFF | Handshake signal is not set |
| 33 | RS-232 | Yellow | ON | Module is parameterized for RS-232 |
| | | | OFF | Module is not parameterized for RS-232 |

| **i** | Channel errors are errors that can be associated with a channel.<br>Periphery errors are errors that affect the entire module. |
|---|---|

| **i** | For more information on the meaning of local diagnostic and status indicators, please refer to the UM EN AXL SYS INST user manual. |
|---|---|

# 10 Serial interfaces

## 10.1 RS-232

The RS-232 interface on the terminal represents some form of DTE (data termination equipment).

This means that plug 2 terminal point 2.1 (TxD) is always used to transmit and plug 2 terminal point 1.1 (RxD) is always used to receive.

According to the standard, some form of DCE (data communication equipment) should be connected to the RS-232 interface as a peer. DTE can also be connected.

Measuring the voltage between the connection points for the TxD and GND signals in idle state will determine whether the device to be connected to the RS-232 interface is a form of DTE or DCE. If the voltage measures approximately -5 V, the device is a form of DTE. If the voltage is approximately 0 V, the device is a form of DCE.

Example: when using a 25-pos. standard plug-in connector the voltage between pin 2 (TxD) and pin 7 (GND) must be measured.

| i | In order to obtain the correct results, perform the measurement on the open cable end, i.e., if the modules are not connected together. |
|---|---|

**RS-232 module handshake signals**

Any device with an RS-232 interface can be connected to the RS-232 interface on the module.

Both the module and the device connected to the RS-232 interface can act as a transmitter and a receiver for data exchange.

To avoid errors during data exchange, e.g., a buffer overrun, the handshake is used as a procedure for the mutual signaling of clear to receive and clear to send.

The module supports the RTS, CTS, DTR, DSR, and DCD handshake signals. Each uses one wire of the connecting cable.

The connecting signals are described from the point of view of the module, i.e., from the point of view of the DTE.

## 10.2 RS-485 (2-wire)

In RS-485 mode, you can create a network with several devices using an existing network consisting of two signal cables.

Use a twisted pair, common shielded data cable to connect the devices. Fit a termination resistor to the data cable at both end points of the RS-485 network. For this, you can use the integrated termination resistor in the module via connections R+ and R-.

If you use the integrated termination resistor, the data cable will also be polarized in order to generate a defined cable idle level.

This operating mode only supports half duplex transmission. Make sure that data is not sent simultaneously by several devices.

## 10.3 RS-422

In RS-422 mode, you can create a network with several devices using an existing network consisting of four signal cables.

The cable pairs create a data forward path and a data return path.

Specify a master for a structure with more than two devices. This master can send telegrams to all the other devices (slaves) and receive from them.

The slaves can send telegrams between each other.

Use a twisted pair, common shielded data cable to connect the devices. Fit the receive signals of the network with termination resistors at the relevant end point. Use the integrated termination resistor when connecting to the module.

This operating mode supports full duplex transmission.

# 11 Data storage and transmission

The module stores the received serial data in an intermediate buffer until it is requested from the serial interface by the bus controller board or the device. Serial data traffic can be managed using various protocols. The protocol used depends on the type of protocol supported by the peer.

## 11.1 Supported protocols

| Protocol | Receive memory | Transmit memory | Special features when receiving |
|---|---|---|---|
| Transparent | 4096 bytes | 1023 bytes | |
| End-to-end | 3 buffers each with 340 bytes (PD) or 245 bytes (PDI) | 1023 bytes (including end characters) | Two end characters are filtered out |
| XON/XOFF | 4096 bytes | 1023 bytes | Software handshake |
| Modbus RTU | 3 buffers each with 340 bytes (PD) or 245 bytes (PDI) | 3 buffers each with 340 bytes (PD) or 245 bytes (PDI) | |

PD          Data exchange via process data
PDI         Data exchange via PDI

## 11.2 Transparent protocol

If the transparent protocol is used, serial data is transmitted in the same format it is received from the serial interface or from the bus side.

The transmit FIFO (first-in, first-out memory) can store 1023 bytes (1 kbyte) and the receive FIFO can store 4096 bytes (4 kbytes). If the module receives another character after the 4095th character, the error pattern is stored in the receive FIFO. All other subsequent characters are ignored.

This protocol supports a CTS hardware handshake.

If the available space in the receive memory is less than 15 bytes, DTR is set to logic 0. As soon as more memory space becomes available again, DTR is set to logic 1.

## 11.3 End-to-end protocol

The serial data is conditioned for the end-to-end protocol.

If serial data is sent from the bus side, two additional characters, the first and second delimiters, are attached for transmission to the serial interface.

A block of serial data sent from the serial interface is only valid if the module has received the first and second delimiters. It is only then that the data can be read via the bus side. The delimiters are not forwarded to the higher-level bus with the user data, they are filtered out.

Unlike in the transparent protocol, the receive memory is not organized as a FIFO but as a buffer.

Three buffers are available.

The size of the buffer depends on the parameterized data way:

| | |
|---|---|
| Data exchange via process data | 340 bytes per buffer |
| Data exchange via PDI | 245 bytes per buffer |

If the maximum buffer size is exceeded without the two delimiters being detected, the previous characters will be ignored and the buffer will be written again.

The transmit FIFO can store 1023 bytes (1 kbyte). The delimiters are attached to, and stored with, the data to be sent.

## 11.4    XON/XOFF protocol

This protocol operates in the same way as the transparent protocol, but uses a software handshake.

Data transmission with this protocol is controlled by the XON and XOFF characters. XON is preset to $11_{hex}$ and XOFF to $13_{hex}$.

If the module receives an XOFF, no more serial data will be sent until an XON is received.

The module itself will transmit an XOFF if the available space in the receive memory is less than 15 bytes. As soon as more memory becomes available, the module will transmit a single XON.

Transmission does not depend on the CTS input.

Serial data is not filtered when it is transmitted. Any characters with the code defined for XON or XOFF are therefore transmitted and may trigger undesirable events at the receiver. When serial data is received, the XON and XOFF characters are filtered and are not available to the higher-lever system as data. Any characters with the XON or XOFF code are lost. Ensure that characters with these codes do not appear in the regular data stream.

## 11.5    Modbus RTU (master support)

**Telegram structure**

| Address | Function | Data | CRC |
|---------|----------|------|-----|
| 1 byte | 1 byte | 0 ... 244 bytes | 2 bytes |
| Data | | | Automatic generation and check |

In the Modbus environment, two time values play an important role: 1.5 characters, also referred to as t1.5 and 3.5 characters, also referred to as t3.5. The times for baud rates above 19,200 baud are set to t1.5 = 1.0 ms and t3.5 = 2.0 ms.

If a break between two characters is greater than t1.5 and less than t3.5, this is identified as a transmission error. The message is considered incomplete and is indicated with a receive error.
If the break after a character is greater than t3.5, this is identified as the telegram end.

Three transmit and three receive buffers are available.

The size of the buffer depends on the parameterized data way:

| | |
|---|---|
| Data exchange via process data | 340 bytes per buffer |
| Data exchange via PDI | 245 bytes per buffer |

Modbus RTU mode offers Modbus RTU master support. However, the module does not have programmable logic.

Modbus RTU slave support is not implemented.

**Data transfer sequence**

The two checksum bytes are automatically generated and sent for the user data supplied via the bus (via process data or PDI). Once sent this has been sent, a response is expected.

If the response is not received within 2.5 seconds, a transmit error is generated by setting error bit Tx in the status word.

If the response is received in time, the checksum is checked. If the checksum is invalid, time t1.5 has elapsed on receipt or a parity error has occurred, a receive error is then generated by setting error bit Rx in the status word.

On error-free receipt, the "Rx buffer not empty" bit is set in the status word.

The checksum bytes are not included when reading the receive data.

# 12 Process data

The module is parameterized via the PDI channel, data exchange with the higher-level controller depends on the parameter data way and is either via process data or via PDI.

The module has ten words of process data. These are made up of 3 bytes of control data and a maximum of 17 bytes of user data.

**Assignment of the ten process data words**

| Word | 0 | | 1 | | 2 | | ... | 9 | |
|---|---|---|---|---|---|---|---|---|---|
| Byte in the Motorola format | 0 | 1 | 2 | 3 | 4 | 5 | ... | 18 | 19 |
| Byte in the Intel format | 1 | 0 | 3 | 2 | 5 | 4 | ... | 19 | 18 |
| OUT | K/P | S | L | D | D | D | ... | D | D |
| IN | K/P | S | L | D | D | D | ... | D | D |

K/P Command/parameter

S Control bits (OUT) or status bits (IN)

L Length: Number of characters to be written (OUT) or to be read (IN)

D Data

The byte representation in the Motorola format, also called Big Endian (high order byte at starting address) corresponds to the INTERBUS standard representation. All byte representations in the data sheet have this format.

The byte representation in the Intel format is also called Little Endian (low order byte at starting address).

The command is used to determine the function. The actually transmitted data depends on the command.

# 13 Process data word 0

## 13.1 Control word

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 0 | Command | | | OUT parameter | | | | Control bits | | | | | | | |

**Control bits**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DTR | 0 | 0 | 0 | 0 | Reset transmit error | Reset receive error | 0 |

| DTR (if DTR control enabled) | | |
|---|---|---|
| Code (bin) | Meaning | Protocol |
| 0 | DTR = logic 0 | All |
| 1 | DTR = logic 1 | |

| Reset transmit error | | |
|---|---|---|
| Code (bin) | Meaning | Protocol |
| 0 | No action | Modbus RTU |
| 1 | Reset transmit error | |

| Reset receive error | | |
|---|---|---|
| Code (bin) | Meaning | Protocol |
| 0 | No action | Modbus RTU |
| 1 | Reset receive error | |

**i** The DTR signal can only be controlled via the bit if you have parameterized interface type RS-232 and DTR control via process data in the parameter table.

**Commands**

| Code (bin) | Code (hex) | Command |
|---|---|---|
| 000 | 0 | Read number of characters received (in the transparent and XON/XOFF protocol) and fill level of the receive buffer |
| 001 | 1 | Transmit characters |
| 010 | 2 | Store characters temporarily |
| 011 | 3 | OUT parameter = $0_{hex}$: Read characters |
| | | OUT parameter = $C_{hex}$: Read firmware version |
| | | OUT parameter = $E_{hex}$: Read counters |
| 100 | 4 | Reserved |
| 101 | 5 | Toggling command 1: Transmit characters |
| 110 | 6 | Toggling command 2: Store characters temporarily |
| 111 | 7 | Toggling command 3: Read characters |

**Command toggling**

Command toggling is used to execute a command on a terminal again. In this way, a second command code is available for the same function.

This applies for the following commands:
– Transmit characters
– Store characters temporarily
– Read characters
– Read counters

Here, bit 14 is used for toggling. If, for example, you wish to transmit character strings in sequence, use command code $001_{bin}$ for the first transmission and then use $101_{bin}$ and $001_{bin}$ alternately.

## 13.2 Status word

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| St | Command | | | IN parameter | | | | Status bits | | | | | | | |

St        Error bit

Reasons for an error bit set:
– Invalid parameter for the specified command
– Failure of the I/O voltage

**Status bits**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|----|----|
| DSR | Transmit buffer not empty | Transmit buffer full | Receive buffer full | DCD | Transmit error | Receive error | Receive buffer not empty |

| DSR | | |
|-----|---|---|
| **Code (bin)** | **Meaning** | **Protocol** |
| 0 | DSR = logic 0 | All |
| 1 | DSR = logic 1 | |

| DCD | | |
|-----|---|---|
| **Code (bin)** | **Meaning** | **Protocol** |
| 0 | DCD = logic 0 | All |
| 1 | DCD = logic 1 | |

The DSR signal state is displayed, if you have parameterized interface type RS-232.

The DCD signal state is displayed, if you have parameterized interface type RS-232.

| Transmit buffer not empty | | |
|-----|---|---|
| **Code (bin)** | **Meaning** | **Protocol** |
| 0 | Empty | All |
| 1 | Not empty, transmission in progress | |

| Transmit error | | |
|-----|---|---|
| **Code (bin)** | **Meaning** | **Protocol** |
| 0 | No error | Modbus RTU |
| 1 | Transmit error; 2.5 s timeout has elapsed with no response received | |

| Transmit buffer full | | |
|-----|---|---|
| **Code (bin)** | **Meaning** | **Protocol** |
| 0 | Not full | All |
| 1 | Full | |

| **Protocol** | **Meaning: transmit buffer full** |
|-----|---|
| Modbus RTU | No more buffer space |
| Other | Space remaining in the transmit buffer ≤ 30 characters |

| Receive error | | |
|-----|---|---|
| **Code (bin)** | **Meaning** | **Protocol** |
| 0 | No error | Modbus RTU |
| 1 | Receive error; telegram received with invalid checksum | |

| Receive buffer full | | |
|-----|---|---|
| **Code (bin)** | **Meaning** | **Protocol** |
| 0 | Not full | All |
| 1 | Full | |

| Receive buffer not empty | | |
|-----|---|---|
| **Code (bin)** | **Meaning** | **Protocol** |
| 0 | Empty | All |
| 1 | Not empty; characters to be read are available | |

| **Protocol** | **Meaning: receive buffer full** |
|-----|---|
| Transparent | Space remaining ≤ 15 characters |
| XON/XOFF | Space remaining ≤ 15 characters |
| Modbus RTU | No more buffer space |
| End-to-end | No more buffer space |

# 14 Commands

### 14.1 "Read number of characters received and fill level of the receive buffer" command

For the transparent and XON/XOFF protocol, the command result is the number of characters that have been received but not yet read.

The number is a 16-bit value and is mapped to word 1.

This command can be used to first reach a minimum number of characters before transmitting the "Read characters" command.

For all protocols, the fill level of the receive buffer is specified in byte 4 as the command result.

**Process data assignment for the "Read number of characters received and fill level of the receive buffer" command**

| Word | 0 | | 1 | | 2 | | ... | 9 | |
|---|---|---|---|---|---|---|---|---|---|
| Byte | 0 | 1 | 2 | 3 | 4 | 5 | ... | 18 | 19 |
| OUT | 00$_{hex}$ | xx | xx | xx | xx | xx | ... | xx | xx |
| IN | 00$_{hex}$ | Status bits | Number of characters received | | Fill level | xx | ... | xx | xx |

| Fill level | | |
|---|---|---|
| **Byte 4** | **Protocol** | |
| | **Transparent, XON/XOFF** | **End-to-end, Modbus RTU** |
| | **Number of kbytes free** | **Number of buffers free** |
| 00$_{hex}$ | 4 | 3 |
| 01$_{hex}$ | < 3 | 2 |
| 02$_{hex}$ | < 2 | 1 |
| 03$_{hex}$ | < 1 | 0 |

### 14.2    "Transmit characters" command

The transmit data located in the process data is stored in the transmit memory. From there the data is transmitted directly via the interface.

A maximum of 17 characters can be transmitted.

Specify the number of characters to be transmitted in the third byte.

If there are characters in the intermediate buffer, these are transmitted first via the selected RS interface.

After the command has been executed successfully, the intermediate buffer is cleared.

**Process data assignment for the "Transmit characters" command with 17 characters (C1 - C17)**

| Word | 0 | | 1 | | 2 | | ... | 9 | |
|---|---|---|---|---|---|---|---|---|---|
| Byte | 0 | 1 | 2 | 3 | 4 | 5 | ... | 18 | 19 |
| OUT | $10_{hex}$ | xx | $17_{dez}$ | Z1 | C2 | C3 | ... | C16 | C17 |
| IN | $10_{hex}$ | Status bits | xx | xx | xx | xx | ... | xx | xx |

Reasons for an error bit set:
–    Byte 2 (number of characters to be transmitted)
     = 0 or > maximum user data length (17 characters)
–    Interface type "Deactivated"
–    Data way: data exchange via PDI

### 14.3    "Store characters temporarily" command

If more than 17 characters are to be transmitted, the transmit data located in the process data is stored in an intermediate buffer which can store up to 340 characters. No characters are transmitted. The "Transmit characters" command is used to transmit temporarily stored data. In this way, blocks of up to 340 characters can be transmitted without a break. They are divided over 20 telegrams with 17 characters each, for example.

Reasons for an error bit set:
–    Byte 2 (number of characters to be transmitted)
     = 0 or > maximum user data length (17 characters)
–    Not enough space in the intermediate buffer

### 14.4 "Read characters" command

A maximum of 17 characters can be read.

For block-oriented protocols (end-to-end, Modbus RTU), the received telegram may be longer than the user data length. In order to indicate that there is more data to be retrieved from the telegram after reading, bit 7 is set in the third receive byte (byte 2).

**Process data assignment for the "Read characters" command with 17 characters (C1 - C17)**

| Word | 0 | | 1 | | 2 | | ... | 9 | |
|------|------|------|------|------|------|------|------|------|------|
| Byte | 0 | 1 | 2 | 3 | 4 | 5 | ... | 18 | 19 |
| OUT | $30_{hex}$ | xx | xx | xx | xx | xx | ... | xx | xx |
| IN | $30_{hex}$ | Status bits | $11_{hex}$ | Z1 | C2 | C3 | ... | C16 | C17 |

Bit 7 in the third input byte (byte 2)

| Byte 2, bit 7 | Effect | Protocol |
|------|------|------|
| 0 | The characters read are the last ones in the block received. | E.g., end-to-end |
| 1 | There are still more characters to be read from the block received. | E.g., end-to-end |

Reasons for an error bit set:
– Interface type "Deactivated"
– Data way: data exchange via PDI

### 14.5 "Read counters" command

This command can be used to read several counters. The counters are used for interface diagnostics.

**Process data assignment for the "Read characters" command**

| Word | 0 | | 1 | | 2 | | 3 | | 4 | 5 | | 6 | |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| Byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| OUT | $3E_{hex}$ | $00_{hex}$ | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx |
| IN | $3E_{hex}$ | Status bits | Number of valid characters received | | Number of invalid characters received (parity, overrun or framing errors) | | Number of characters transmitted | | Reserved | | | | |

# 15 Parameter, diagnostics and information (PDI)

Parameter and diagnostic data as well as other information is transmitted via the PDI channel.

| **i** | For information on PDI, please refer to the UM EN AXL SYS INST user manual. |
| --- | --- |

The standard and application objects stored in the module are described in the following section.

| **i** | Please refer to the basic profile for comprehensive information. |
| --- | --- |

The following applies to all tables below:

Please refer to the UM DE AXL SYS PRO or the basic profile for an explanation of the object codes and data types.

| Abbreviation | Meaning |
| --- | --- |
| A | Number of elements |
| L | Length of the elements |
| R | Read |
| W | Write |

| **i** | Every visible string is terminated with a zero terminator ($00_{hex}$). The length of a visible string element is therefore one byte larger than the amount of user data. |
| --- | --- |

# 16 Standard objects

## 16.1 Objects for identification (device rating plate)

| Index (hex) | Object name | Object type | Data type | A | L | Rights | Meaning | Contents |
|---|---|---|---|---|---|---|---|---|
| **Manufacturer** | | | | | | | | |
| 0001 | VendorName | Var | Visible String | 1 | 16 | R | Manufacturer name | Phoenix Contact |
| 0002 | VendorID | Var | Visible String | 1 | 7 | R | Manufacturer identification | 00A045 |
| 0003 | VendorText | Var | Visible String | 1 | 49 | R | Comment on the manufacturer | Components and systems for industrial automation |
| 0012 | VendorURL | Var | Visible String | 1 | 30 | R | URL of the manufacturer | http://www.phoenix-contact.com |
| **Module - general** | | | | | | | | |
| 0004 | DeviceFamily | Var | Visible String | 1 | 15 | R | Device family | I/O function module |
| 0006 | ProductFamily | Var | Visible String | 1 | 33 | R | Product family | Axioline - High speed I/O system |
| 000E | CommProfile | Var | Visible String | 1 | 4 | R | Communication profile | 633 |
| 000F | DeviceProfile | Var | Visible String | 1 | 5 | R | Device profile | 0010 |
| 0011 | ProfileVersion | Record | Visible String | 2 | 11; 20 | R | Device profile version | 2011-12-07; Basic Profile V2.0 |
| 003A | VersionCount | Array | Unsigned 16 | 4 | 4 * 2 | R | Version counter | 0007 0000 0001 0001$_{hex}$ |
| **Module - special** | | | | | | | | |
| 0005 | Capabilities | Var | Visible String | 1 | 8 | R | Properties | Energ_0 |
| 0007 | ProductName | Var | Visible String | 1 | 19 | R | Product designation | AXL F RS UNI 1H ME |
| 0008 | SerialNo | Var | Visible String | 1 | 11 | R | Serial number | xxxxxxxxxx (e. g., 1234512345) |
| 0009 | ProductText | Var | Visible String | 1 | 24 | R | Product text | 1 communication channel |
| 000A | OrderNumber | Var | Visible String | 1 | 8 | R | Order No. | 2688679 |
| 000B | HardwareVersion | Record | Visible String | 2 | 11; 3 | R | Hardware version | e. g., 2011-02-04; 00 |
| 000C | FirmwareVersion | Record | Visible String | 2 | 11; 6 | R | Firmware version | 0000-00-00; V1.00 |
| 000D | PChVersion | Record | Visible String | 2 | 11; 6 | R | Parameter channel version | 0000-00-00; PDI-API V0.30 |
| 0037 | DeviceType | Var | Octet string | 1 | 8 | R | Module identification | 00 00 08 14 00 00 00 C0$_{hex}$ |
| **Use of the device** | | | | | | | | |
| 0014 | Location | Var | Visible String | 1 | 59 | R/W | Installation location | Can be filled out by the user. |
| 0015 | EquipmentIdent | Var | Visible String | 1 | 59 | R/W | Equipment identifier | Can be filled out by the user. |
| 0016 | ApplDeviceAddr | Var | Unsigned 16 | 1 | 2 | R/W | User-defined device number | Can be filled out by the user. |

## 16.2 Object for multilingual capacity

| Index (hex) | Object name | Object type | Data type | A | L | Rights | Meaning | Contents |
|---|---|---|---|---|---|---|---|---|
| 0017 | Language | Record | Visible String | 2 | 6; 8 | R | Language | en-us; English |

## 16.3 Diagnostics objects

| Index (hex) | Object name | Object type | Data type | A | L | Rights | Assignment/content |
|---|---|---|---|---|---|---|---|
| 0018 | DiagState | Record | | 6 | 2; 1; 1; 2; 1; 14 | R | Diagnostics state; see below |
| 0019 | ResetDiag | Var | Unsigned 8 | 1 | 1 | W | Reset diagnostics |

**Diagnostics state (0018$_{hex}$: DiagState)**

This object is used for a structured message of an error.

| 0018$_{hex}$: DiagState (Read) | | | | | |
|---|---|---|---|---|---|
| **Subindex** | **Data type** | **Length in bytes** | **Meaning** | **Contents** | |
| 0 | Record | 21 | Diagnostic state | Complete diagnostics information | |
| 1 | Unsigned 16 | 2 | Error number | 0 ... 65535$_{dec}$ | |
| 2 | Unsigned 8 | 1 | Priority | 00$_{hex}$ | No error |
| | | | | 01$_{hex}$ | Error |
| | | | | 02$_{hex}$ | Warning |
| | | | | 81$_{hex}$ | Error removed |
| | | | | 82$_{hex}$ | Warning eliminated |
| 3 | Unsigned 8 | 1 | Group | 00$_{hex}$ | No error |
| | | | | 01$_{hex}$ | Channel 1 |
| | | | | FF$_{hex}$ | Entire device |
| 4 | Unsigned 16 | 2 | Error code | See table below | |
| 5 | Unsigned 8 | 1 | More information follows | 00$_{hex}$ (not supported) | |
| 6 | Visible String | 14 | Text (14 characters) | See table below | |

| i | The message with the priority 81$_{hex}$ or 82$_{hex}$ is a one-time internal message to the bus coupler that is implemented onto the error mechanisms of the higher-level system by the bus coupler. |
|---|---|

| i | After all errors have been eliminated, it is automatically reset. |
|---|---|

Error code and status of the local status and diagnostics indicators

| Error | Error code | Text | Priority | Group | D LED | UI LED | E1 LED | E2 LED |
|---|---|---|---|---|---|---|---|---|
| No error | 0000$_{hex}$ | Status OK | 00$_{hex}$ | 00$_{hex}$ | Green ON | ON | OFF | OFF |
| Receive buffer full | 7610$_{hex}$ | Rx-Buf full | 02$_{hex}$ | 01$_{hex}$ | Green ON | ON | OFF | ON |
| Transmit buffer full | 7611$_{hex}$ | Tx-Buf full | 02$_{hex}$ | 01$_{hex}$ | Green ON | ON | OFF | ON |
| Faulty supply voltage | 5160$_{hex}$ | Supply fail | 01$_{hex}$ | FF$_{hex}$ | Flashing green/yellow | OFF | ON | ON |
| Device error | 6301$_{hex}$ | CS FLASH | 01$_{hex}$ | FF$_{hex}$ | Flashing green/yellow | ON | OFF | ON |

**Reset diagnostics (0019$_{hex}$: ResetDiag)**

You can delete the diagnostics memory and acknowledge the diagnostic messages with this object.

| 0019$_{hex}$: ResetDiag (Write) | | | | | |
|---|---|---|---|---|---|
| Subindex | Data type | Length in bytes | Meaning | Contents | |
| 0 | Unsigned 8 | 1 | Reset diagnostics | 00$_{hex}$ | All diagnostic messages approved |
| | | | | 02$_{hex}$ | Deletes and acknowledges all pending diagnostic messages that have not been read out |
| | | | | 06$_{hex}$ | Deletes and acknowledges all the diagnostic messages and allows no further diagnostic messages |
| | | | | Other | Reserved |

## 16.4 Objects for process data management

| Index (hex) | Object name | Object type | Data type | A | L | Rights | Assignment |
|---|---|---|---|---|---|---|---|
| 0025 | PDIN | Var | Octet string | 1 | 20 | R | Input process data |
| 0026 | PDOUT | Var | Octet string | 1 | 20 | R | Output process data |
| 003B | PDIN_Descr | Array of Records | | 6 | 24 | R | Description of the IN process data |
| 003C | PDOUT_Descr | Array of Records | | 6 | 24 | R | Description of the output process data |

The objects 003B$_{hex}$ and 003C$_{hex}$ are only applicable to tools.

**IN process data (0025hex: PDIN)**

You can read the IN process data of the module with this object.

The structure corresponds to the representation in the "Process data" section.

| 0025hex: PDIN (Read) | | | |
|---|---|---|---|
| **Subindex** | **Data type** | **Length in bytes** | **Meaning** |
| 0 | Octet string | 20 | Input process data |

**OUT process data (0026hex: PDOUT)**

You can read the OUT process data of the module with this object.

The structure corresponds to the representation in the "Process data" section.

| 0026hex: PDOUT (Read) | | | |
|---|---|---|---|
| **Subindex** | **Data type** | **Length in bytes** | **Meaning** |
| 0 | Octet string | 20 | Output process data |

# 17 Application objects

| Index (hex) | Object name | Object type | Data type | A | L | Rights | Assignment |
|---|---|---|---|---|---|---|---|
| 0080 | ParaTable | Array | Octet string | 1 | 16 | R/W | Parameter table |
| 0081 | SerialData | Var | Octet string | 1 | 0 ... 245 | R/W | Serial data |
| 0095 | EventCounter | Var | Octet string | 1 | 12 | R | Event counter |

## 17.1    Parameter table (0080hex: ParaTable)

Parameterize the module using this object.

In the case of valid parameters, the parameterization is stored in the module permanently.

After resetting, the module works with the last permanently stored data. Upon delivery, the module works with the default data (default settings).

| Byte | Meaning | Default value |
|---|---|---|
| 0 | Type, protocol | $30_{hex}$ |
| 1 | Baud rate, data width | $72_{hex}$ |
| 2 | 1st delimiter | $0D_{hex}$ |
| 3 | 2nd delimiter | $0A_{hex}$ |
| 4 | Error pattern | $24_{hex}$ |
| 5 ... 7 | Uni1 ... Uni3 | $00\ 00\ 00_{hex}$ |
| 8 | Reserved | $00_{hex}$ |
| 9 | Data way | $00_{hex}$ |
| 10 | Tv | $00_{hex}$ |
| 11 | Tn | $00_{hex}$ |
| 12 ... 15 | Reserved | $00\ 00\ 00\ 00_{hex}$ |

| $0080_{hex}$: ParaTable (read, write) | | |
|---|---|---|
| Subindex | Data type | Length in bytes |
| 0 | Octet string | 16 |

| Word | 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| | T/P | B/DB | 1st del | 2nd del | ErrP | Uni1 | Uni2 | Uni3 | res. | Data way | Tv | Tn | res. | res. | res. | res. |

**Element value range**



| The options in bold are default settings. |
|---|

**Byte 0 (T/P: interface type/protocol)**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| res. | DTR | Interface type | | Protocol | | | |

| DTR: DTR control | |
|---|---|
| **Code (bin)** | **Meaning** |
| **0** | **Automatic** |
| 1 | Can be controlled via process data |

DTR control is only significant for interface type RS-232.

For automatic control, the following applies in the transparent protocol and in the XON/XOFF protocol:

| DTR = logic 0 | There are a maximum of 15 characters free in the receive buffer. |
|---|---|
| DTR = logic 1 | There are more than 15 characters free in the receive buffer. |

For control via process data, the DTR signal can be controlled directly with bit 7 (DTR).

| Interface type | |
|---|---|
| **Code (bin)** | **Meaning** |
| 00 | RS-232 |
| 01 | RS-485 |
| 10 | RS-422 |
| **11** | **Deactivated** |

Select an RS-xxx interface type. With the "Deactivated" default setting, no data is transmitted or received. When using the default setting, an error message is generated and the error bit is set.

| Protocol | |
|---|---|
| **Code (hex)** | **Meaning** |
| **0** | **Transparent** |
| 1 | End-to-end |
| 2 | XON/XOFF |
| 3 | Modbus RTU |
| Other | Reserved |

**Byte 1 (D/DB: baud rate/data width)**

| Baud rate | |
|---|---|
| **Code (hex)** | **Wert (baud)** |
| 0 | 110 |
| 1 | 300 |
| 2 | 600 |
| 3 | 1200 |
| 4 | 1800 |
| 5 | 2400 |
| 6 | 4800 |
| **7** | **9600** |
| 8 | 15625 |
| 9 | 19200 |
| A | 38400 |
| B | 57600 |
| C | 115200 |
| D | 230400 |
| E | 250000 |
| F | Direct (Uni1 ... 3) |

The specified baud rates of 110 baud to 250000 baud are adequate for most applications. However, you can freely choose the baud rate by direct programming. For this, use the $0F_{hex}$ baud rate code.

| Data width | | | |
|---|---|---|---|
| **Code (hex)** | **Meaning** | | |
| | **Data bits** | **Parity** | **Stop bits** |
| 0 | 7 | Even | 1 |
| 1 | 7 | Odd | 1 |
| **2** | **8** | **Even** | **1** |
| 3 | 8 | Odd | 1 |
| 4 | 8 | Without | 1 |
| 5 | 7 | Without | 1 |
| 6 | 7 | Even | 2 |
| 7 | 7 | Odd | 2 |
| 8 | 8 | Even | 2 |
| 9 | 8 | Odd | 2 |
| A | 8 | Without | 2 |
| B | 7 | Without | 2 |
| C | 8 | Constantly at 0 | 1 |
| D | 8 | Constantly at 1 | 1 |
| E | 6 | Without | 1 |
| F | Directly (Uni1) | | |

The specified combinations of data width, parity and stop bits are adequate for most applications. However, you can freely choose the combination by direct programming. For this, use the $0F_{hex}$ data width code.

### Byte 2 and 3 (1st del, 2nd del: 1st and 2nd delimiters)

The 1st delimiter and 2nd delimiter contain the end characters for the end-to-end and dual buffer protocols.

When selecting the end-to-end protocol, the two delimiters can be adjusted.

Default settings:

| Delimiter | Default value | ASCII character |
|-----------|---------------|------------------|
| 1st del | $0D_{hex}$ | CR = carriage return |
| 2nd del | $0A_{hex}$ | LF = line feed |

### Byte 4 (ErrP: error pattern)

The error pattern contains the character that is written to the FIFO if a character was received with errors (e.g., in the event of a parity error).

The error pattern is used for the following protocols:

– Transparent
– End-to-end
– XON/XOFF

| Error pattern | |
|---------------|---------|
| **Code (hex)** | **Meaning** |
| **24** | **$** |
| xx | Any character |
| 00 | If a character is received with an error, no error pattern is stored. |
| FF | The invalid character is stored instead of the error pattern. |

### Byte 5 ... 7 (Uni1 ... Uni3: universal byte 1 ... 3)

These bytes can be used as universal bytes for direct specification of the baud rate or the data width.

| Byte 5 | | | | | | | | Byte 6 | | | Byte 7 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | ... | 0 | 7 | ... | 0 |
| Data width | | | | | | | | Baud rate | | | | | |

### Direct specification of the data width

If the provided combinations of data width, parity and stop bits are inadequate, they can be directly specified. Use the Uni1 byte. Bit 1 and bit 0 are reserved.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Parity | | Enable parity | Stop bits | Data bits | | res. | res. |

| Parity | |
|--------|---------|
| **Code (bin)** | **Meaning** |
| **00** | **Odd** |
| 01 | Even |
| 10 | Constantly at 1 |
| 11 | Constantly at 0 |

| Enable parity | |
|---------------|---------|
| **Code (bin)** | **Meaning** |
| **0** | **Disable** |
| 1 | Enable |

| Stop bits | |
|-----------|---------|
| **Code (bin)** | **Meaning** |
| **0** | **1 stop bit** |
| 1 | 2 stop bits |

| Data bits | |
|-----------|---------|
| **Code (bin)** | **Meaning** |
| **00** | **5 bits** |
| 01 | 6 bits |
| 10 | 7 bits |
| 11 | 8 bits |

**Direct specification of baud rate**

If the provided baud rates do not correspond to the desired baud rate, the baud rate can be directly specified. For this, write the desired baud rate on the Uni1 to Uni3 fields in the form of a numeric value. Transfer is right-aligned. However, only the 2 lower bits of Uni1 are permitted.

The maximum value is 18 bits. The maximum value = $3FFFF_{hex}$ which corresponds to 262143 baud.

Example for direct specification of the baud rate:
–   Baud rate: 100000 baud
–   $100000_{dez}$ = 01 86 $A0_{hex}$

| Word | 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | |
|------|------|--------|-----|-------------|------|------|------|------|------|------|------|------|------|------|
| Byte | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| OUT | $40_{hex}$ | $xx_{hex}$ | T/P | B/DB | res. | res. | ErrP | Uni1 | Uni2 | Uni3 | res. | res. | Tv | Tn |
| OUT | $40_{hex}$ | $xx_{hex}$ | xx | $Fx_{hex}$ | 00 | 00 | xx | 01 | 86 | A0 | 00 | 00 | xx | xx |

**Byte 9 (data way)**

Here you can control the way used to exchange data with the selected interface.

| Data way | |
|----------|---------|
| **Code (hex)** | **Meaning** |
| **$00_{hex}$** | **Data exchange via process data** |
| $01_{hex}$ | Data exchange via PDI |

| Data exchange via process data | 340 bytes per buffer |
|--------------------------------|----------------------|
| | A maximum of 17 characters can be transmitted together using the "Transmit characters" command. If you want to transmit more characters, use the "Store characters temporarily" command. |
| Data exchange via PDI | 245 bytes per buffer |

**Byte 10 and 11 (Tv, Tn: lead time, lag time)**

These two times are used for the RS-232 protocol

Tv          Lead time

Tn          Lag time

The times are specified with a resolution of 1 ms. This allows for waiting times of 1 ms ... 255 ms.

When transmitting data via RS-232 the RTS signal is set. If Tv does not equal 0, transmission only starts after this time has elapsed. After all the data is sent and Tn does not equal 0, the RTS signal is only set to LOW after this time has elapsed.

### 17.2 Serial data (0081$_{hex}$: SerialData)

If you have parameterized data exchange via PDI in the "Data way" parameter in the parameter table, this object can be used to read and write the serial data.

| 0081$_{hex}$: SerialData (read, write) | | | |
|---|---|---|---|
| Subindex | Data type | Length in bytes | Contents |
| 0 | Var | 0 ... 245 | Serial data |

On write access, the user data of this service is transmitted directly via the interface.

On read access, the user data corresponds to the received characters.

The maximum user data length is 245 bytes.

In the case of the Modbus RTU protocol, on write access after transmitting the user data, two bytes are appended for a checksum determined by the module.
On read access, the checksum is no longer present when the user data is received.

| Errors | Error class (hex) | Error code (hex) | Additional code (hex) |
|---|---|---|---|
| "Data way" parameter = 00$_{hex}$ (data exchange via process data) | 8 | 0 | 0022 |
| "Interface" parameter = 3$_{hex}$ (deactivated) | 8 | 0 | 0122 |
| On write access, there is not enough space in the transmit memory | 8 | 0 | 0222 |
| On read/write access, the previous read/write access has not yet been completed | 8 | 0 | 0322 |
| On write access, the length is equal to 0 or greater than 245 | 6 | 5 | 0000 |

### 17.3 Event counter (0095hex: EventCounter)

You can read multiple counters which are used for interface diagnostics with this object.

| 0095hex: EventCounter (read) | | |
|---|---|---|
| **Subindex** | **Data type** | **Length in bytes** |
| 0 | Octet string | 12 |

| Word | Meaning |
|---|---|
| 1 | Number of valid characters received |
| 2 | Number of invalid characters received (parity, overrun or framing errors) |
| 3 | Number of characters transmitted |
| 4 ... 6 | Reserved |

## 18 Device descriptions

The device is described in the device description files.
The device descriptions for controllers from Phoenix Contact are included in PC Worx and the corresponding service packs.
The device description files for other systems are available for download at phoenixcontact.net/download in the download area of the bus coupler used.

# OCEAN CHIPS
## Океан Электроники
### Поставка электронных компонентов

Компания «Океан Электроники» предлагает заключение долгосрочных отношений при поставках импортных электронных компонентов на взаимовыгодных условиях!

Наши преимущества:

- Поставка оригинальных импортных электронных компонентов напрямую с производств Америки, Европы и Азии, а так же с крупнейших складов мира;
- Широкая линейка поставок активных и пассивных импортных электронных компонентов (более 30 млн. наименований);
- Поставка сложных, дефицитных, либо снятых с производства позиций;
- Оперативные сроки поставки под заказ (от 5 рабочих дней);
- Экспресс доставка в любую точку России;
- Помощь Конструкторского Отдела и консультации квалифицированных инженеров;
- Техническая поддержка проекта, помощь в подборе аналогов, поставка прототипов;
- Поставка электронных компонентов под контролем ВП;
- Система менеджмента качества сертифицирована по Международному стандарту ISO 9001;
- При необходимости вся продукция военного и аэрокосмического назначения проходит испытания и сертификацию в лаборатории (по согласованию с заказчиком);
- Поставка специализированных компонентов военного и аэрокосмического уровня качества (Xilinx, Altera, Analog Devices, Intersil, Interpoint, Microsemi, Actel, Aeroflex, Peregrine, VPT, Syfer, Eurofarad, Texas Instruments, MS Kennedy, Miteq, Cobham, E2V, MA-COM, Hittite, Mini-Circuits, General Dynamics  и др.);

Компания «Океан Электроники» является официальным дистрибьютором и эксклюзивным представителем в России одного из крупнейших производителей разъемов военного и аэрокосмического назначения «JONHON», а так же официальным дистрибьютором и эксклюзивным представителем в России производителя высокотехнологичных и надежных решений для передачи СВЧ сигналов «FORSTAR».



**«JONHON»** (основан в 1970 г.)

Разъемы специального, военного и аэрокосмического назначения:

(Применяются в военной, авиационной, аэрокосмической, морской, железнодорожной, горно- и нефтедобывающей отраслях промышленности)

**«FORSTAR»** (основан в 1998 г.)

ВЧ соединители, коаксиальные кабели, кабельные сборки и микроволновые компоненты:

(Применяются в телекоммуникациях гражданского и специального назначения, в средствах связи, РЛС, а так же военной, авиационной и аэрокосмической отраслях промышленности).



Телефон:    8 (812) 309-75-97 (многоканальный)
Факс:        8 (812) 320-03-32
Электронная почта:   ocean@oceanchips.ru
Web:       http://oceanchips.ru/
Адрес:  198099, г. Санкт-Петербург, ул. Калинина, д. 2, корп. 4, лит. А