MODBUS TCP with Kolver K-Ducer introduction

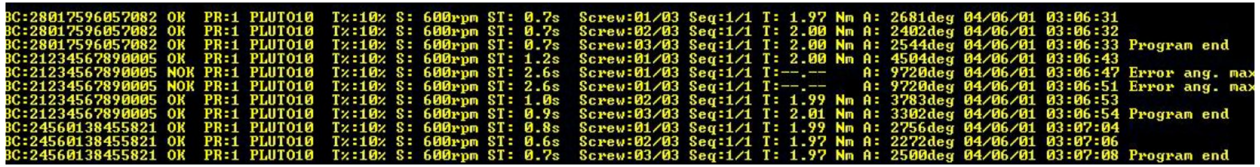
Industrial communication protocols with serial cables

Based on RS-232 and/or RS-485 standards

Note:

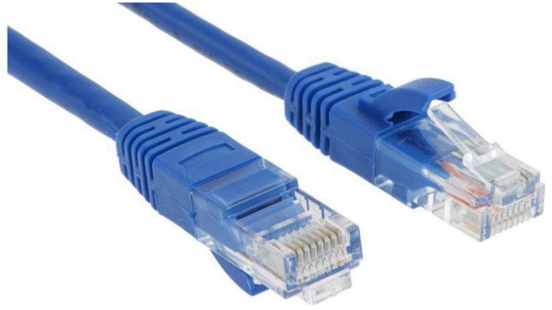
Kolver EDU/TOP and K-Ducer don’t follow any of these protocols on their serial ports.

They simply transmit the results string (printer string) at the end of each screwdriving or on any errors, following the RS-232 standard.



```
BC:28017596057082 OK  PR:1 PLUTO10  T%:10% S: 600rpm ST: 0.7s  Screw:01/03 Seq:1/1 T: 1.97 Nm A: 2681deg 04/06/01 03:06:31
BC:28017596057082 OK  PR:1 PLUTO10  T%:10% S: 600rpm ST: 0.7s  Screw:02/03 Seq:1/1 T: 2.00 Nm A: 2402deg 04/06/01 03:06:32
BC:28017596057082 OK  PR:1 PLUTO10  T%:10% S: 600rpm ST: 0.7s  Screw:03/03 Seq:1/1 T: 2.00 Nm A: 2544deg 04/06/01 03:06:33 Program end
BC:21234567890005 OK  PR:1 PLUTO10  T%:10% S: 600rpm ST: 1.2s  Screw:01/03 Seq:1/1 T: 2.00 Nm A: 4504deg 04/06/01 03:06:43
BC:21234567890005 NOK PR:1 PLUTO10  T%:10% S: 600rpm ST: 2.6s  Screw:01/03 Seq:1/1 T:--.--     A: 9720deg 04/06/01 03:06:47 Error ang. max
BC:21234567890005 NOK PR:1 PLUTO10  T%:10% S: 600rpm ST: 2.6s  Screw:01/03 Seq:1/1 T:--.--     A: 9720deg 04/06/01 03:06:51 Error ang. max
BC:21234567890005 OK  PR:1 PLUTO10  T%:10% S: 600rpm ST: 1.0s  Screw:02/03 Seq:1/1 T: 1.99 Nm A: 3783deg 04/06/01 03:06:53
BC:21234567890005 OK  PR:1 PLUTO10  T%:10% S: 600rpm ST: 0.9s  Screw:03/03 Seq:1/1 T: 2.01 Nm A: 3302deg 04/06/01 03:06:54 Program end
BC:24560138455821 OK  PR:1 PLUTO10  T%:10% S: 600rpm ST: 0.8s  Screw:01/03 Seq:1/1 T: 1.99 Nm A: 2756deg 04/06/01 03:07:04
BC:24560138455821 OK  PR:1 PLUTO10  T%:10% S: 600rpm ST: 0.6s  Screw:02/03 Seq:1/1 T: 1.97 Nm A: 2272deg 04/06/01 03:07:06
BC:24560138455821 OK  PR:1 PLUTO10  T%:10% S: 600rpm ST: 0.7s  Screw:03/03 Seq:1/1 T: 1.97 Nm A: 2500deg 04/06/01 03:07:08 Program end
```

Industrial communication protocols via ethernet cables



Based on the TCP/IP standard

Open Protocol

Modbus TCP

Supported directly by the K-Ducer on CN5

EtherNet/IP™

PROFI® INDUSTRIAL ETHERNET NET

(and others) indirectly supported by the K-Ducer via procol converters such as AnyBus. Must purchase a device with "MODBUS TCP Client" interface, and configure it to communicate with the K-Ducer using MODBUS TCP

TCP/IP is the same standard used by non-industrial communication protocols such as:
http (web browsing), FTP, SSH, telnet, etc

OPC® FOUNDATION

Based on other/custom standards

PROFI® INDUSTRIAL ETHERNET NET RT real time

EtherCAT® P

DeviceNet™

CAN

MODBUS TCP vs OPEN PROTOCOL

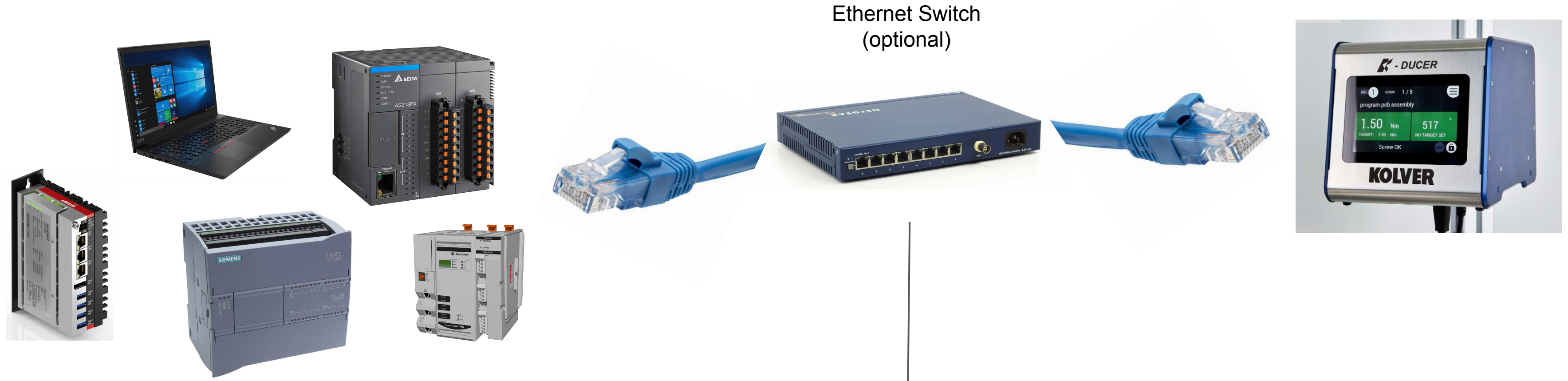
Both supported by K-Ducer

	MODBUS TCP	OPEN PROTOCOL
Specifies how messages are constructed and how data is to be exchanged	✔	✔
Specifies a list of MID commands specific to screwdriving	✘	✔
Ready-to-use libraries for PC and virtually every PLC with an ethernet port	✔	✘
Plug-and-play use on many 'MES' software packages (VGP+, FactoryLogix, ToolsNet, Tulip, ...)	✘	✔

MODBUS TCP: Client vs Server

MODBUS TCP **<u>Client</u>** = PC, PLC

MODBUS TCP **<u>Server</u>** = K-Ducer



Client role:

Open/close TCP connection (port 502)*

<u>Sending</u> MODBUS requests
Messages must be constructed appropriately, following:

1. MODBUS specifications (unit ID, function codes, etc)*
2. K-Ducer MODBUS map

*usually taken care of by the library provided by PLC manufacturer

Server role:

Accept/refuse connection

<u>Replying</u> to messages (requests)

Server does NOT send unsolicited messages (server only sends replies)

MODBUS Map

The MODBUS protocol has two types of data and two types of data access:

- 2 data types: single ***bit*** , and 16-bit ***word***
- 2 access types: read only, and read+write

This results in four subdivisions (four arrays): discrete inputs, coils, input registers, and holding registers

Array name =>	**Discrete inputs**	**Coils**	**Input registers**	**Holding registers**
Data type =>	1-bit (boolean)	1-bit (boolean)	16-bit (word)	16-bit (word)
Access type =>	Read only	Read + Write	Read only	Read + Write

Each of these four is an array of data, either *bit* or *word*, either read only or rear + write.

The manufacturer decides how to organize the data (settings, remote control commands, parameters etc) onto these four arrays; the MODBUS protocol does not specify how these arrays are to be used, only the types of data and acces that each one has.

The way the data is organized is called the **MODBUS Map**

MODBUS function codes

Every MODBUS request sent by the client contains the following:

- whether the request is to *read* or *write* some data
- which one of the four arrays to access
- which address (or range of addresses) of the array to read or write

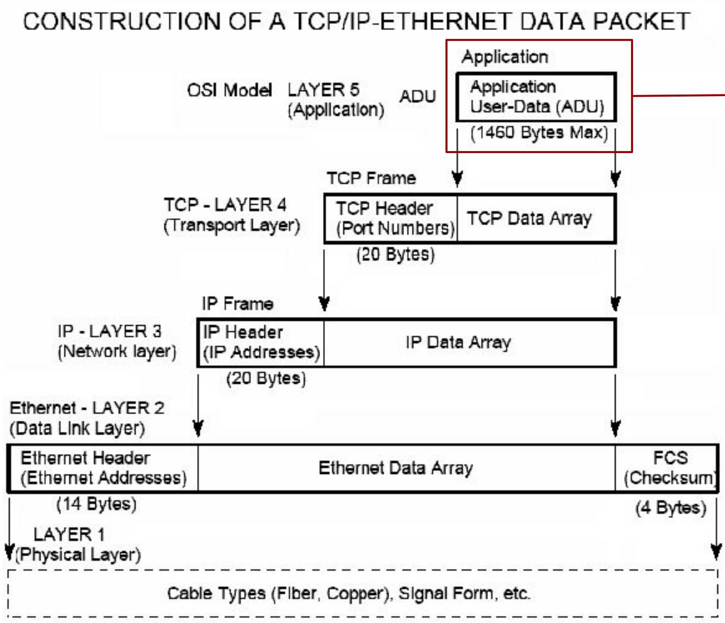
The MODBUS function code is simply a number identifying the type of data request, as follows:

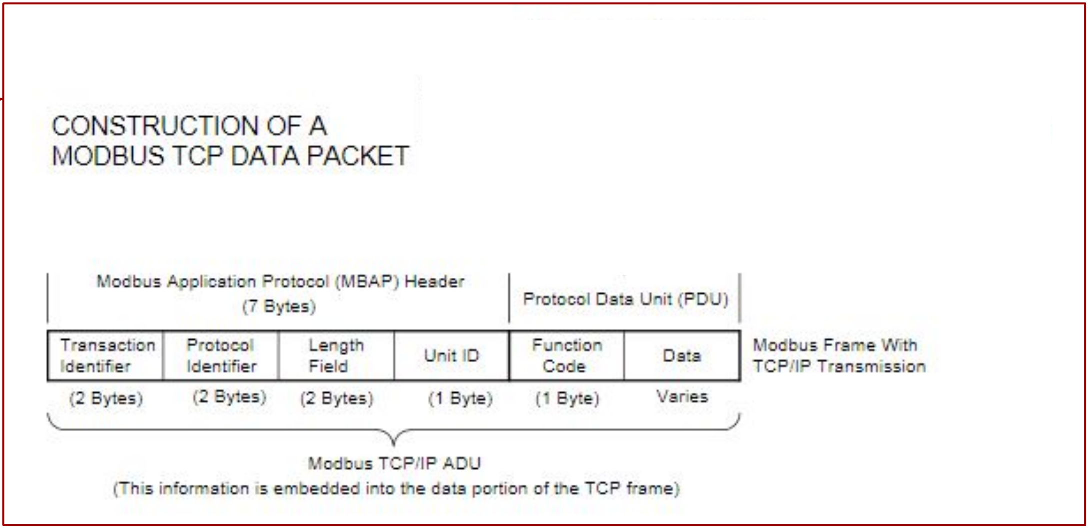
Array name =>	**Discrete inputs**	**Coils**	**Input registers**	**Holding registers**
Data type =>	1-bit (boolean)	1-bit (boolean)	16-bit (word)	16-bit (word)
Access type =>	Read only	Read + Write	Read only	Read + Write
Function code to read or write to the array =>	0x02 (read discrete input)	0x01 (read coil) 0x05 (write coil)	0x04 (read input register)	0x03 (read holding register) 0x06 (write holding register)

MODBUS: message structure

Fortunately, the MODBUS protocol is widely supported by PLCs and PC libraries, so the end user rarely has to worry about how to construct each message and TCP packet correctly.

Rather, the end user simply specifies the function code, the address of the data to read, or the address of the data to write along with the value to write. The PLC/PC library will take care of building the TCP packet, sending it, and receiving the response from the server (ie from the K-Ducer).


CONSTRUCTION OF A TCP/IP-ETHERNET DATA PACKET
OSI Model
LAYER 5 (Application)
ADU
Application
Application User-Data (ADU)
(1460 Bytes Max)
TCP Frame
TCP - LAYER 4 (Transport Layer)
TCP Header (Port Numbers)
(20 Bytes)
TCP Data Array
IP Frame
IP - LAYER 3 (Network layer)
IP Header (IP Addresses)
(20 Bytes)
IP Data Array
Ethernet - LAYER 2 (Data Link Layer)
Ethernet Header (Ethernet Addresses)
(14 Bytes)
Ethernet Data Array
FCS (Checksum)
(4 Bytes)
LAYER 1 (Physical Layer)
Cable Types (Fiber, Copper), Signal Form, etc.


CONSTRUCTION OF A MODBUS TCP DATA PACKET
Modbus Application Protocol (MBAP) Header (7 Bytes)
Protocol Data Unit (PDU)
Transaction Identifier
(2 Bytes)
Protocol Identifier
(2 Bytes)
Length Field
(2 Bytes)
Unit ID
(1 Byte)
Function Code
(1 Byte)
Data
Varies
Modbus Frame With TCP/IP Transmission
Modbus TCP/IP ADU
(This information is embedded into the data portion of the TCP frame)

Bringing it all together: K-Ducer MODBUS Map

Array name =>	**Discrete inputs**	**Coils**	**Input registers**	**Holding registers**
Data type =>	1-bit (boolean)	1-bit (boolean)	16-bit (word)	16-bit (word)
Access type =>	Read only	Read + Write	Read only	Read + Write
Function code to read or write to the array =>	0x02 (read discrete input)	0x01 (read coil) 0x05 (write coil)	0x04 (read input register)	0x03 (read holding register) 0x06 (write holding register)
K-Ducer data or commands contained =>	Bits mirroring the <u>physical input pins</u> 1-20 of CN3 (read only! Not for remote control via modbus)	Bits mirroring the <u>physical output pins</u> 23-43 of CN3 (read only), plus: Bits for remote control of the screwdriver via MODBUS: start, stop, reverse, OK, ESC, Reset (read + write access)	*Words* with data related to the state of the screwdriver: Current screwdriving state (ie tightening, reversing) Last torque and angle result, Last error, if any Screwdriver serial number, And more (read only)	*Words* with all K-Ducer configuration data: All program and sequence parameters (read+write, write not recommended) Program or sequence currently selected (read+write)

The complete map with all data addresses can be found in excel format in the “MODBUS TCP Resources” packet provided by Kolver

Communication with a K-Ducer via MODBUS TCP:

Steps to take:

1. On the K-Ducer, ensure the communication protocol selected (General Settings > page 4) is MODBUS TCP and assign an appropriate IP address and subnet mask

2. From the PLC or PC, through the modbus functionality provided by the PLC vendor or PC library, open a MODBUS TCP connection towards the K-Ducer IP address on port 502 (default MODBUS TCP port)

3. Once the connection is opened, user can start sending and receiving MODBUS messages. Typically, the PLC/PC will have pre-built commands for sending messages, where the user simply enters:
   a. Desired MODBUS function code
   b. Desired MODBUS data address to read/write
   c. Desired value to write (for write commands) or desired PLC/PC variable to store results

Code examples for PLC/PC and more can be found in the "MODBUS TCP Resources" packet provided by Kolver