con::lyte V5
Modbus Protocol BASIC

Version V1.01
08.04.2020

Erstellt / geändert:	08.04.2020	CH	Freigabe:	08.04.2020	ABE	Informationssicherheit:	**öffentlich**
Dateiname:	Modbus_Protocol_conlyteV5_BASIC_20200408_GF.docx					Revision Nr.:	1.01

List of contents

1 About this Document

This guide explains how to transfer results using MODBUS RTU from the con::lyte V5 slave RS-485 interface. This allows to implement the terminal to your SCADA system.

2 Modbus Mapping

Supporting documents

- **Modbus** MODBUS application protocol specification v1.1a [1]
- **Modbus RTU** MODBUS Serial Line Protocol and Implementation Guide V1.02 [2]

2.1 Structure of the Document

Register	Tag name	Address	Type	Length	R/W	Description	Address	R/W-Range	Default value
There are two different register types: input registers with read only access and holding register with read write access. For a couple of registers we have a group name, for example "device description".	a tagname must start with a prefix (see sheet "datatypes") followed by a short abbreviation for the variable.	is the address of the register in HEX (register number - 1)	datatype, see "datatypes" table	length in multiples of register size	allowed access: R = read, R/W = read/ write	describes the value of the register	is the address of the register in DEC (register number - 1)	gives the range of the value	default value for this register

2.2 Main Datatypes

Type	Description	Size (Bytes)	Prefix	Note
char[x]	string	x	ab	ASCII
bitmask	bitmask of 16 bits	2	bm	
uint16	unsigned 16 bit integer	2	ui	NaN = UINT16_MAX
float	IEEE754 floating point	4	f	NaN = 0xffff ffff
enum	enum type	2	e	
timestamp (TAI64N) *1	timestamp	12	ts	

*1 = http://cr.yp.to/proto/tai64.txt

For all data types the Big-Endian encoding is used.

2.3 Register Mapping

Input registers, 16Bit, RO	Tag name	Address	Type	Length	R/W	Description	Address		Range	Default value
device description	uiVersion	0x0000	uint16	1	R	modbus-mapping protocol version. For all changes in public registers: 0xAABB AA..major version, BB..minor version (compatible).	0		full	0x0107 = v1.07
	eVendor	0x0001	enum	1	R	vendor code	1		full	0x96C3
	eModel	0x0002	enum	1	R	device model	2		full	0x2900
	abModel	0x0003	char[20]	10	R	device model description. padded with spaces.	3		ASCII	"con::lyte V5 "
	abSerialNumber	0x000D	char[8]	4	R	serial number. padded with spaces.	13		ASCII	"YYKWXXXX"
	uiParameterCount	0x0016	uint16	1	R	number of parameters	22		0-8	0
parameter 1 result	bmP1Status	0x0080	bitmask	1	R	parameter 1 status.	128		full	0000 0000 0000 0000b
	bmP1PrivStatus	0x0084	bitmask	1	R	parameter 1 private status.	129		full	0000 0000 0000 0000b
	xP1Value	0x0082	float	2	R	parameter 1 result	130		full	NaN
parameter 2 result	bmP2Status	0x0088	bitmask	1	R	parameter 2 status.	136		full	0000 0000 0000 0000b
	bmP2PrivStatus	0x0089	bitmask	1	R	parameter 2 private status.	137		full	0000 0000 0000 0000b
	xP2Value	0x008A	float	2	R	parameter 2 result	138		full	NaN
parameter 3 result	bmP3Status	0x0090	bitmask	1	R	parameter 3 status.	144		full	0000 0000 0000 0000b
	bmP3PrivStatus	0x0091	bitmask	1	R	parameter 3 private status.	145		full	0000 0000 0000 0000b
	xP3Value	0x0092	float	2	R	parameter 3 result	146		full	NaN

parameter 4 result	bmP4Status	0x0098	bitmask	1	R	parameter 4 status.	152		full	0000 0000 0000 0000b
	bmP4PrivStatus	0x0099	bitmask	1	R	parameter 4 private status.	153		full	0000 0000 0000 0000b
	xP4Value	0x009A	float	2	R	parameter 4 result	154		full	NaN
parameter 5 result	bmP5Status	0x00A0	bitmask	1	R	parameter 5 status.	160		full	0000 0000 0000 0000b
	bmP5PrivStatus	0x00A1	bitmask	1	R	parameter 5 private status.	161		full	0000 0000 0000 0000b
	xP5Value	0x00A2	float	2	R	parameter 5 result	162		full	NaN
parameter 6 result	bmP6Status	0x00A8	bitmask	1	R	parameter 6 status.	168		full	0000 0000 0000 0000b
	bmP6PrivStatus	0x00A9	bitmask	1	R	parameter 6 private status.	169		full	0000 0000 0000 0000b
	xP6Value	0x00AA	float	2	R	parameter 6 result	170		full	NaN
parameter 7 result	bmP7Status	0x00B0	bitmask	1	R	parameter 7 status.	176		full	0000 0000 0000 0000b
	bmP7PrivStatus	0x00B1	bitmask	1	R	parameter 7 private status.	177		full	0000 0000 0000 0000b
	xP7Value	0x00B2	float	2	R	parameter 7 result	178		full	NaN
parameter 8 result	bmP8Status	0x00B8	bitmask	1	R	parameter 8 status.	184		full	0000 0000 0000 0000b
	bmP8PrivStatus	0x00B9	bitmask	1	R	parameter 8 private status.	185		full	0000 0000 0000 0000b
	xP8Value	0x00BA	float	2	R	parameter 8 result	186		full	NaN

controller 1 result	bmC1Status	0x1000	uint16	1	R	controller 1 status. See "datatypes"	4096		full	65535
	bmC1RunStatus	0x1001	uint16	1	R	controller 1 running status. See "datatypes"	4097		full	0
	xC1Value	0x1002	float	2	R	controller 1 value; 1% results in 0.01	4098		full	nan
	xC1Process	0x1004	float	2	R	controller 1 process value, actual value	4100		full	nan
	xC1Disturbance	0x1006	float	2	R	controller 1 disturbance input value	4102		full	nan
	xC1Setpoint	0x1008	float	2	R	controller 1 setpoint value	4104		full	nan
controller 2 result	bmC2Status	0x1100	uint16	1	R	controller 1 status. See "datatypes"	4352		full	65535
	bmC2RunStatus	0x1101	uint16	1	R	controller 1 running status. See "datatypes"	4353		full	0
	xC2Value	0x1102	float	2	R	controller 1 value; 1% results in 0.01	4354		full	nan
	xC2Process	0x1104	float	2	R	controller 1 process value, actual value	4356		full	nan
	xC2Disturbance	0x1106	float	2	R	controller 1 disturbance input value	4358		full	nan
	xC2Setpoint	0x1108	float	2	R	controller 1 setpoint value	4360		full	nan

controller 3 result	bmC3Status	0x1200	uint16	1	R	controller 1 status. See "datatypes"	4608		full	65535
	bmC3RunStatus	0x1201	uint16	1	R	controller 1 running status. See "datatypes"	4609		full	0
	xC3Value	0x1202	float	2	R	controller 1 value; 1% results in 0.01	4610		full	nan
	xC3Process	0x1204	float	2	R	controller 1 process value, actual value	4612		full	nan
	xC3Disturbance	0x1206	float	2	R	controller 1 disturbance input value	4614		full	nan
	xC3Setpoint	0x1208	float	2	R	controller 1 setpoint value	4616		full	nan

Holding registers, 16Bit, RW	Tag name	Address	Type	Length	R/W	Description	Address		Range	Default value
parameter 1 configuration public	abP1Name	0x0078	char[8]	4	R	parameter 1 name, filled up with spaces	120		full	
	abP1Unit	0x007C	char[8]	4	R	parameter 1 unit, filled up with spaces	124		full	
	xP1UpperLimit	0x0080	float	2	R	upper limit for parameter 1 result (measuring range)	128		full	
	xP1LowerLimit	0x0082	float	2	R	lower limit for parameter 1 result (measuring range)	130		full	
parameter 2 configuration public	abP2Name	0x00F0	char[8]	4	R	parameter 2 name, filled up with spaces	240		full	
	abP2Unit	0x00F4	char[8]	4	R	parameter 2 unit, filled up with spaces	244		full	
	xP2UpperLimit	0x00F8	float	2	R	upper limit for parameter 2 result (measuring range)	248		full	
	xP2LowerLimit	0x00FA	float	2	R	lower limit for parameter 2 result (measuring range)	250		full	
Parameter 3 configuration public	abP3Name	0x0168	char[8]	4	R	parameter 3 name, filled up with spaces	360		full	
	abP3Unit	0x016C	char[8]	4	R	parameter 3 unit, filled up with spaces	364		full	
	xP3UpperLimit	0x0170	float	2	R	upper limit for parameter 3 result (measuring range)	368		full	
	xP3LowerLimit	0x0172	float	2	R	lower limit for parameter 3 result (measuring range)	370		full	
parameter 4 configuration public	abP4Name	0x01E0	char[8]	4	R	parameter 4 name, filled up with spaces	480		full	
	abP4Unit	0x01E4	char[8]	4	R	parameter 4 unit, filled up with spaces	484		full	
	xP4UpperLimit	0x01E8	float	2	R	upper limit for parameter 4 result (measuring range)	488		full	

	xP4LowerLimit	0x01EA	float	2	R	lower limit for parameter 4 result (measuring range)	490		full	
parameter 5 configuration public	abP5Name	0x0258	char[8]	4	R	parameter 5 name, filled up with spaces	600		full	
	abP5Unit	0x025C	char[8]	4	R	parameter 5 unit, filled up with spaces	604		full	
	xP5UpperLimit	0x0260	float	2	R	upper limit for parameter 5 result (measuring range)	608		full	
	xP5LowerLimit	0x0262	float	2	R	lower limit for parameter 5 result (measuring range)	610		full	
parameter 6 configuration public	abP6Name	0x02D0	char[8]	4	R	parameter 6 name, filled up with spaces	720		full	
	abP6Unit	0x02D4	char[8]	4	R	parameter 6 unit, filled up with spaces	724		full	
	xP6UpperLimit	0x02D8	float	2	R	upper limit for parameter 6 result (measuring range)	728		full	
	xP6LowerLimit	0x02DA	float	2	R	lower limit for parameter 6 result (measuring range)	730		full	
parameter 7 configuration public	abP7Name	0x0348	char[8]	4	R	parameter 7 name, filled up with spaces	840		full	
	abP7Unit	0x034C	char[8]	4	R	parameter 7 unit, filled up with spaces	844		full	
	xP7UpperLimit	0x0350	float	2	R	upper limit for parameter 7 result (measuring range)	848		full	
	xP7LowerLimit	0x0352	float	2	R	lower limit for parameter 7 result (measuring range)	850		full	
parameter 8 configuration public	ab8Name	0x03C0	char[8]	4	R	parameter 8 name, filled up with spaces	960		full	
	abP8Unit	0x03C4	char[8]	4	R	parameter 8 unit, filled up with spaces	964		full	

	xP8UpperLimit	0x03C8	float	2	R	upper limit for parameter 8 result (measuring range)	968		full	
	xP8LowerLimit	0x03CA	float	2	R	lower limit for parameter 8 result (measuring range)	970		full	
controller configuration private	xC1ControllerSetpoint	0x0400	float	2	R W	controller 1 setpoint value	1024		full	0
	xC2ControllerSetpoint	0x0402	float	2	R W	controller 2 setpoint value	1026		full	0
	xC3ControllerSetpoint	0x0404	float	2	R W	controller 3 setpoint value	1028		full	0
parameter configuration private	fP1LowerAlarmLimit	0x0500	float	2	R	parameter 1 lower alarm limit	1280		full	nan
	fP1UpperAlarmLimit	0x0502	float	2	R	parameter 1 upper alarm limit	1282		full	nan
	fP2LowerAlarmLimit	0x0520	float	2	R	parameter 2 lower alarm limit	1312		full	nan
	fP2UpperAlarmLimit	0x0522	float	2	R	parameter 2 upper alarm limit	1314		full	nan
	fP3LowerAlarmLimit	0x0540	float	2	R	parameter 3 lower alarm limit	1344		full	nan
	fP3UpperAlarmLimit	0x0542	float	2	R	parameter 3 upper alarm limit	1346		full	nan
	fP4LowerAlarmLimit	0x0560	float	2	R	parameter 4 lower alarm limit	1376		full	nan
	fP4UpperAlarmLimit	0x0562	float	2	R	parameter 4 upper alarm limit	1378		full	nan
	fP5LowerAlarmLimit	0x0580	float	2	R	parameter 5 lower alarm limit	1408		full	nan

	fP5UpperAlarmLimit	0x0582	float	2	R	parameter 5 upper alarm limit	1410		full	nan
	fP6LowerAlarmLimit	0x05A0	float	2	R	parameter 6 lower alarm limit	1440		full	nan
	fP6UpperAlarmLimit	0x05A2	float	2	R	parameter 6 upper alarm limit	1442		full	nan
	fP7LowerAlarmLimit	0x05C0	float	2	R	parameter 7 lower alarm limit	1472		full	nan
	fP7UpperAlarmLimit	0x05C2	float	2	R	parameter 7 upper alarm limit	1474		full	nan
	fP8LowerAlarmLimit	0x05E0	float	2	R	parameter 8 lower alarm limit	1504		full	nan
	fP8UpperAlarmLimit	0x05E2	float	2	R	parameter 8 upper alarm limit	1506		full	nan

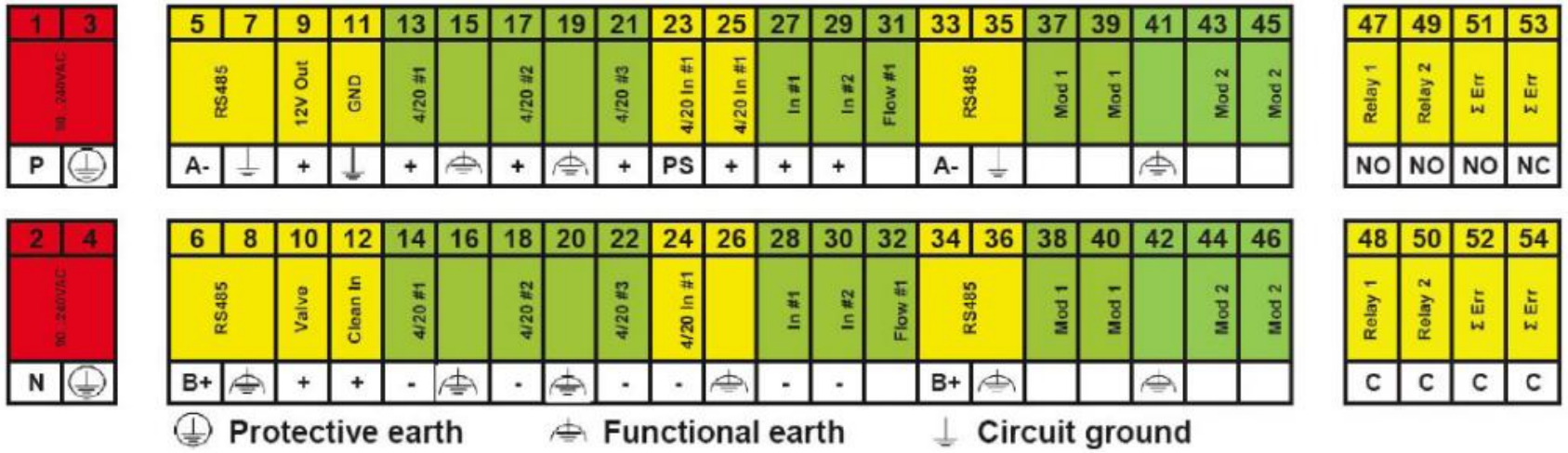
2.4 Additional Datatypes

Status information within registers bmPxStatus and bmPxPrivStatus are related to parameter shown. For details see sensors manual.

	Value	Description
bmC1Status (multiple errors may sum up)	0x0000	no error
	0x0001	general error
	0x0002	input error
	0x0004	output error
	0x0008	disturbance error
	0xFFFF	controller disabled

	Value	Description
bmC1RunStatus	0	stopped
	1	running
	2	hold, manual

3 Plan of Terminal Connectors



Connect transmission lines to

- A-: terminal no. 33
- B+: terminal no. 34
- GND: terminal no. 35

Factroy default communication settings:

modbus address	1
modbus	RTU
baudrate	38400
parity	odd

4 Implementation

Important hints:

- Please be careful that only the required registers are written. Changes in other registers can be result in a wrong behavior of the con::lyte.
- We recommend to check the written values by an additional reading.

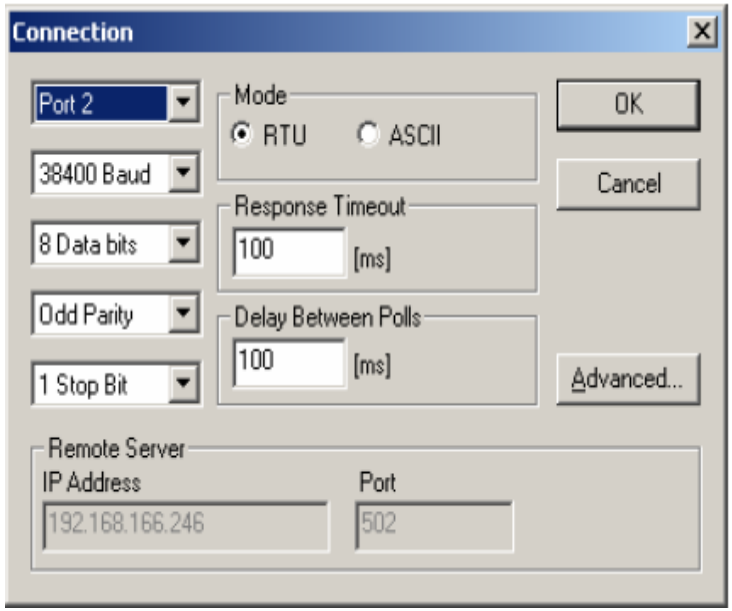
5 Timing rules for communication

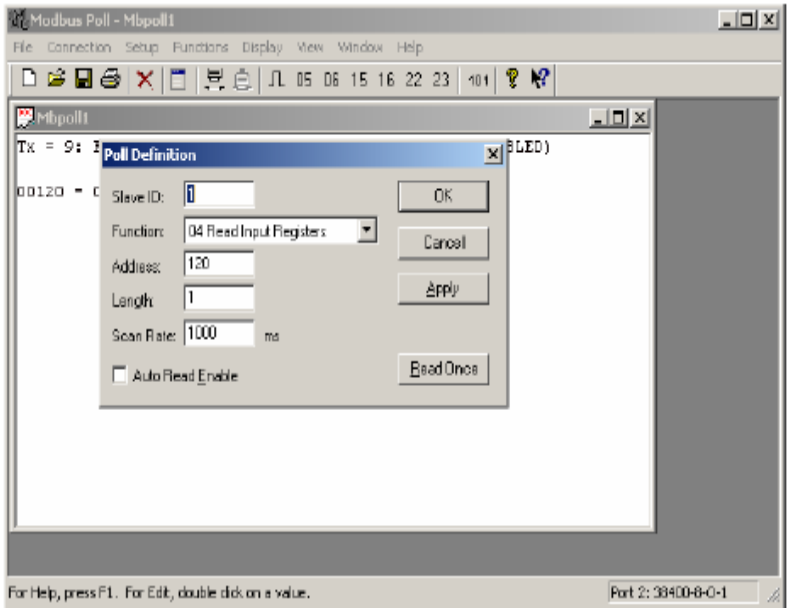
T1: max. allowed time sign to sign within one datagram = see protocol definition.
T2: max. allowed time between command completely received by sensor and sensor begin to sends his answer= 100msec.
(The controller has to add the sending time of the command and 2*the link latency for the first sign timeout).
T3: min. needed time between two datagram = see protocol definition.
T4: min. needed time after failure < T2.
T5: min. needed time after CRC without any following signs = see protocol definition.

6 FAQ

- **What is the best way to read measurement values?**
  We recommend to read all the parameter values together with their status, the device status and sample time. This has the advantage that all results are atomic in the sense that they belong to exactly the same timestamp and less MODBUS transactions are needed. The disadvantage is that the MODBUS master must post process the response by converting the individual registers.

- **How to test the MODBUS connection?**
  We recommend to use the Modbus Poll Utility from Modbus Tools. It is an excellent testing tool and a demo version is freely available. Simply enter the Connections menu and if your s::can probe is still in the default configuration, you can use the same settings as shown in the left figure below with the exception of the communication port.
  Then you should create a new Poll Definition as shown in the right figure below which polls the status register of the device. After you have created the Poll Definition Click on Connect and the register should be read from the device and should read 0x0000.





Erstellt / geändert:	08.04.2020	CH	Freigabe:	08.04.2020	ABE	Informationssicherheit:	öffentlich	
Dateiname:	Modbus_Protocol_conlyteV5_BASIC_20200408_GF.docx						Revision Nr.:	1.01

- **What is the recommended workaround for timeouts on MODBUS requests during measurements?**
  We recommend to retry the communication if a timeout occurs (3 re-tries using 1000msec. timeout).

- **Can other MODBUS devices be connected to the same network?**
  Yes - If you operate the s::can probe in a MODBUS network it is safe to connect other probes.

- **Why do your MODBUS register start at address zero and my registers are named differently?**
  The MODBUS Legacy Protocol (See “Modicon Modbus Protocol Reference Guide PI–MBUS–300 Rev. J”)
  Input Registers are addressed as 3XXXX where the register XXXX corresponds to the register XXXX – 1 in our mapping. For example to read the input registers starting at address 0x80 to 0x82 one should read the registers 30080 to 30082.

- **Is there a continuous set of registers for parameter values?**
  There is no continuous register set available but all registers can be read in a single MODBUS transaction.

- **What MODBUS timeout should be configured on the master?**
  The MODBUS timeout should be set to 1000ms.

- **What MODBUS register types are supported?**
  We support input registers. Input registers are used for parameter readings and status information.

7 Referred Documents

[1] Modbus application protocol specification v1.1a, http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1a.pdf
[2] MODBUS Serial Line Protocol and Implementation Guide V1.02, http://www.modbus.org/docs/Modbus_over_serial_line_V1_02.pdf

Änderungsverzeichnis: (NICHT ins PDF)

Rev. Nr.	Datum	Autor	Beschreibung und Grund der Änderungen
1.0	18.02.2020	CH	First complete version
1.01	08.04.2020	CH	Minor changes in text, removed referrer of status information, added reference document links

scan Messtechnik GmbH

Brigittagasse 22-24, 1200 Vienna, Austria
Tel.: +43 (0) 1 219 73 93 – 0
Fax: +43 (0) 1 219 73 93 – 12
office@s-can.at
www.s-can.at