Alternative Printing Services

apsolute communication specification

Specification

aps GmbH

aps GmbH, Behringstraße 7, D-71083 Herrenberg, Germany, Tel.:+49 7032 9164-01, Fax: +49 7032 9164-03

Author: Michael Kögler

File name: apsolute_Comms_v1_15b_eng.doc

Version: V1.15b First release: 18.12.2007

Change protocol

Version	Datum	Author	Changes (Chapter/content)
V1.00	24.10.06	MK	First draft / split from Release 2
V1.01	09.11.06	MK	additions
V1.02	27.11.06	MK	description UDP- and message-Telegrams
V1.03	31.01.07	MK	ISO model
V1.04	23.04.07	MK	Change to MODBUS
V1.041	01.08.07	MK	Chapter 5 Examples included
V1.042	10.08.07	MK	Error in chapter 5 corrected
V1.05	12.11.07	MK	Changes in chapter 3.2.1; 4; 4.2.1; 4.3.1.1; 4.3.2; 4.3.2.1;4.3.3; 4.3.4; 4.3.5; 4.3.8.1
V1.10	18.12.07	BS/EB/MK	complete revision
V1.11	30.01.08	MK	numbering in chapter 5 corrected
V1.12	30.09.08	MK	Command 62 corrected
V1.13	16.10.08	MK	Command 44 and 45 corrected, pages tuned with german version
V1.14	09.01.09	EB	Diverse corrections
V1.15a	09.04.09	EB	transmission of the error messages
V.1.15b	15.04.09	MK	Diverse corrections wording

Content

1 Data exchange protocol

This protocol is used to connect the apsolute to a PLC or PC by using a serial RS232 communication or alternative an Ethernet connection.

For the communication the MODBUS protocol is used. When using a RS232 communication the protocol MODBUS RTU is used, when using the Ethernet communication the MODBUS TCP protocol is used.

Remark: This document does not describe the general MODBUS specification itself, it assumes that this specification is known. You can find the specs of the MODBUS protocol on http://www.Modbus-IDA.org.

In the following document a PLC or PC that is connected via RS232 or Ethernet to the apsolute is called client.

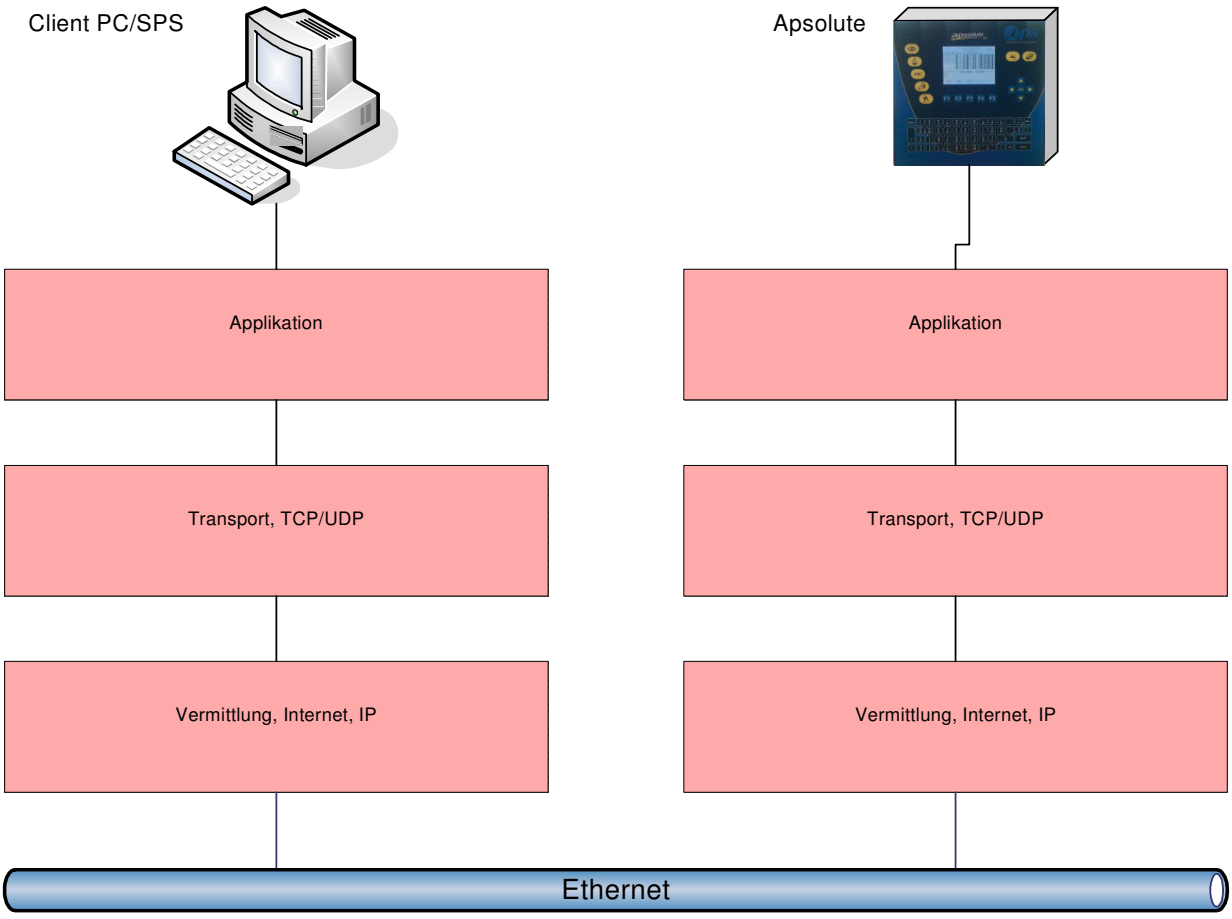
Attention: The client can only change values in the volatile memory of the apsolute (exception download of files). Therefore the values must be sent again after every reset of the apsolute!!

2 Layers for data exchange protocol

In a network a lot of duties must be performed and there are different requirements regarding reliability, security and efficiency. The problems to be solved vary from questions of electronic data transfer up to abstract tasks which have to be performed. Because of this variety of problems and tasks all these duties are separated in different layers with fixed requirements. A specific entity performs the task in each layer. The entities on the side to send and the side to receive have to work on the basis of defined rules to make sure that they use the same way to exchange the data.

The rules are defined in a protocol. Every entity offers services that can be used from an entity above. The entity itself uses services from an entity below. The real data flow is in vertical direction.

The advantage of such a layer based network protocol is its flexibility. Layers can be exchanged or modified by specific protocols to adapt the protocol to specific needs. To ensure the performance of the total protocol it is only necessary to keep the communication link to the layer above and the layer below.



2.1 Applikation = application

The application level is the user level. This is the top level layer for the user. This layer provides various functionalities, see chapter "application specific data".

2.2 Transport = transport

The transport layer is the control layer that ensures the control and back up of data transferred between the application layer and the transmission (Ethernet) layer. Protocol examples for this layer are TCP and UDP.

2.3 Vermittlung = medium

This layer ensures the data exchange. It controls the communication between the partners in terms of logic and timing, independent from the topology and the medium. Normally internet protocol IP is used for this layer.

2.4 Ethernet

Under the medium layer there is the communication area. In a local network this can be e.g. the Ethernet.

3 Description of the protocol

3.1 MODBUS Protocol

Modbus is a user protocol for the exchange of information between intelligent Modbus controllers, independent of the structure of the net. The Modbus protocol is associated to the application level of the OSI reference model. It supports master slave communication between intelligent equipment.

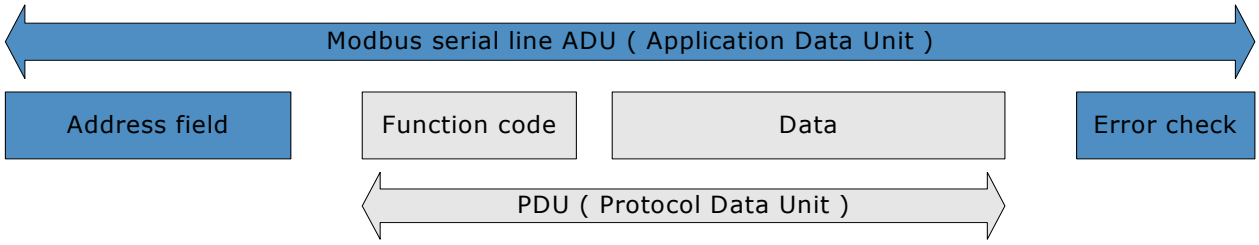
The Modbus protocol defines the type of message which the master station can use, independent of the network through which the Modbus controller will communicate. It describes how the Modbus controller starts the communication with another controller, how this responds to questions and how errors are recognised and documented. Modbus protocol works on the basis of question and answer. It offers several services which are specified by a function code. During communication the Modbus protocol defines how each controller receives its individual address and the way each controller can sort out the messages related to it. Whenever an answer is needed, the controller builds a message and sends it with the Modbus protocol to the related station.

Because this Modbus protocol can be enclosed in other frames, the Modbus is often used in an industrial Ethernet and has the reserved port number 502 in the TCP/IP stack. The TCP/IP version is called Modbus-TCP/IP.

The Modbus protocol defines a simple protocol data unit (PDU) which is independent of the layers below. The above level application data unit (ADU) contains additional areas which are related to the used network.

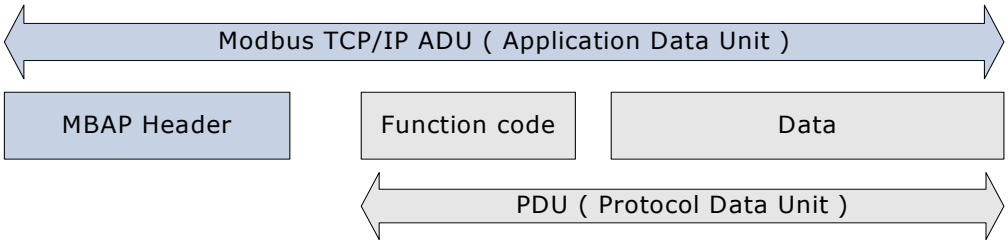
3.1.1 MODBUS RTU

When using the Modbus protocol for the serial communication the function code, data, slave address and CRC checksum are transmitted in addition (see "Modbus over serial line specification and implementation guide V1.02")



3.1.2 MODBUS TCP/IP protocol

Modbus TCP/IP Protocol uses the MBAP Header (Modbus Application Protocol Header). The data transfer is administrated by the TCP protocol in the layer below (see "MODBUS Messaging on TCP/IP Implementation Guide V1.0b")



Structure of the MBAP header		
No. of bytes	**Function**	**Description**
2	Transaction identifier	Code for transmission
2	Protocol identifier	Protocol identifier = 0
2	Length	No. of following bytes
1	Unit identifier	slave address

3.2 MODBUS data transmission

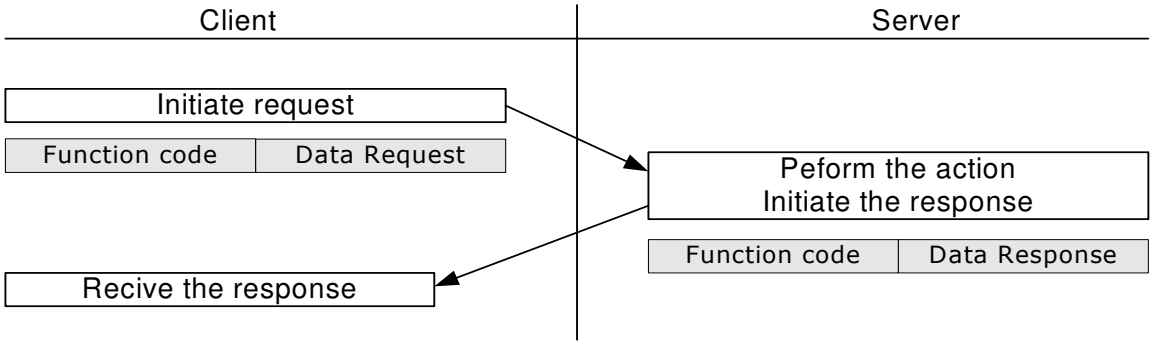
The application data unit „ADU" from the MODBUS telegram is composed by the client which initiates the transmission. The function code defines the type of function performed. The data area of the telegram includes additional information which is needed to perform the function, .e.g. register number and quantity of registers. The length of the protocol data area (function Code + data) is restricted to 253 bytes in the MODBUS protocol.

Possible status of the MODBUS data transmission:

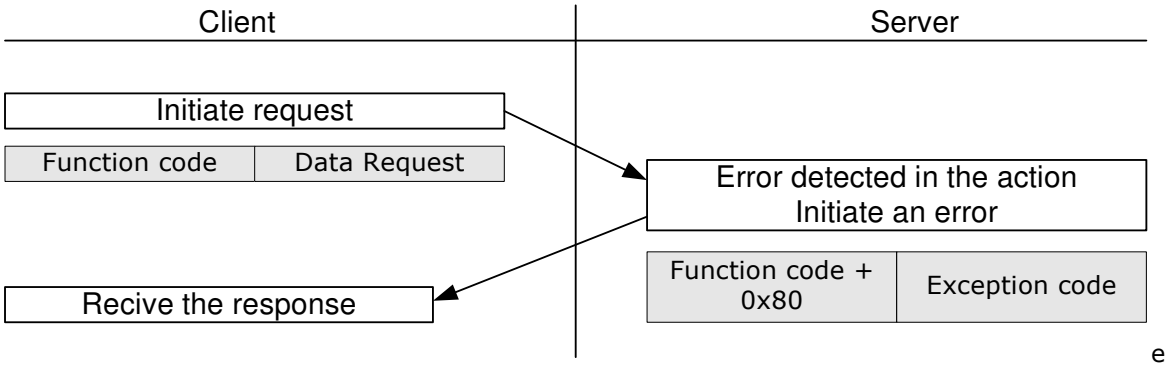
- The server receives the question from the client without any communication error. The server can process this request and sends a normal response message with the same function code and accordant data back.
- The server receives, because of a communication error, no question from the client. In this case the server will send no response message back. The client must monitor this error with a timeout control.
- The server receives the question from the client but recognises a communication error (parity, CRC, a.s.o.). In this case no response message is send back. The client must monitor this error with a timeout control.
- The server receives the question from the client without communication error, but is not able to process this request. E.g. due to a not supported function or register number. In this case the response message with a function code + 0x80 and an exception code is sent back. The exception code is described in the following table.

List of MODBUS Exception Codes		
Code	**Function**	**Description**
1	**ILLEGAL FUNCTION**	Not supported function code
2	**ILLEGAL DATA ADDRESS**	Register number not valid
3	**ILLEGAL DATA VALUE**	User data not in allowed limits

Error free MODBUS data exchange



MODBUS data exchange with error



e

3.2.1 Response times

After the reception without error the requests are processed from the controller right away. The time to process and to send an answer depends on the type of request, these times are very different. They can be from a millisecond (read variable) to some seconds (Switch from single modus to list modus). In general the processing of a request that needs read/write of the internal flash memory takes much longer than a read/write of a single value.

3.3 Function codes

The function code defines the type of the function to perform. The range of allowed codes is 1...255, while the range from 128 to 255 is reserved for exception responses (error answers). For the "exception responses" a 0x80 is added to the original function code.

The following list shows the function codes supported by the apsolute.

List of MODBUS function codes		
function code	**Identification**	**Description**
4	**read register**	MODBUS standard function code to read one or more read registers
101	**user specific function code**	User specific function code used by apsolute. The functions are defined according to the message command that is transmitted in the additional message header in the data area.

The Modbus standard function code "4" is used to identify the apsolute on the MODBUS. To achieve this registers are implemented which are described in the following chapter. The total special functions of the apsolute are realised with the user specific function code "101".

3.4 MODBUS register

With the Modbus standard function code 4 the following registers can be read

List of MODBUS registers				
Register no.	**Description**	**Content**	**NO. of bytes**	**R/W**
1	Manufacturer name	„APS "	16	RO
11	Product description	„apsolute V1 "	16	RO
21	Serial number	„00000000 "	16	RO
31	Version string	„V2.00.0 31.12.2007 "	32	RO

3.4.1 Example to read version string

Request of version string		
Data area	**Value (Hex)**	**Description**
Function code	0x04	Read register
Starting address Hi	0x00	Starting address = Register No. – 1
Starting address Lo	0x1E	
Quantity of register Hi	0x00	Quantity of register = NO. of Bytes / 2
Quantity of register Lo	0x10	

Attention: The starting address is the register no.-1, because the MODBUS standard numbers the registers from 1 to 255, but the addresses of these registers start with 0.

Attention: A register always consists of one word with 2 bytes. The quantity of registers is therefore half of the number of bytes.

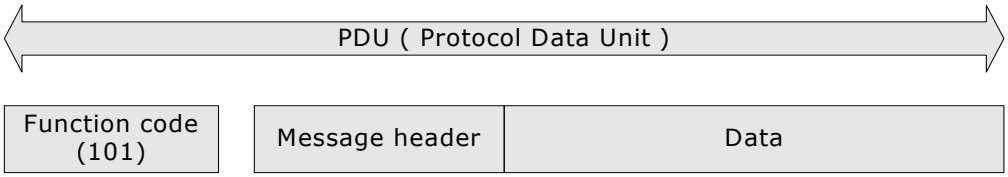
Response of version string		
Data area	**Value (Hex)**	**Description**
Function code	0x04	Read register
Byte count	0x20	32 Bytes
manufacturer	„V2.00.0 31.12.2007 “ (ASCII-characters)	32 characters of the version string

Attention: The content is not a C-String; it is an array of characters. The array is filled with blanks and contains no terminating zero.

Attention: In the request the number of registers is addressed. In the answer the number of bytes is addressed.

4 Application specific data transfer

For the application specific Modbus data transfer the user specific function code 101 and in addition a 4 byte long message header is used. The length of a MODBUS message is restricted to 253 bytes and therefore the data area is restricted to 249 bytes.



.

The message header is defined as following:

Structure of message headers		
No. of Bytes	**Identification**	**Description**
1	Command number	Command number, defines the type of the application specific function to perform. The functions and command numbers are shown in the tables below.
1	Status	Additional status information for the execution of the application specific function
2	Identifier	The parameter „Identifier" can be used by the client software to allocate the received response messages explicitly to the request messages sent. The identifier is sent back from the apsolute unchanged.

In principle the application specific data exchange works according to the request/response principle, meaning the client always sends a request to the apsolute and the apsolute answers with a response.

Attention: when reading the following chapter please note: with a Get Value the value of the variable to be read is not put in the request sent. In this case the value is only put in the received response. With a Set Value the value to be written is put in the sent request. In this case the response only confirms the achieved message.

The following list includes all values of the additional status information for the application specific functions.

List of Status (application specific message header)	
Value	**Description**
0	No error
1	Unknown command
2	Unknown drive or drive not ready
3	Unknown or not valid folder
4	Unknown file
5	Error when reading the file
6	Error when writing the file
7	Unknown variable
8	Unknown string
9	Illegal index
10	Flow line buffer of variable text is full
11	Illegal value
12	Value cannot be read (write only) Value cannot be written (read only)
13	Internal data error

4.1 Application specific functions

The following list includes all application specific functions and the command numbers which can be used by the apsolute with a user specific function code

List of commands		
Command number	**Function**	**Description**
1	**Get_Volume info**	Read drive information The drive number is send from the client to the apsolute. Then the drive information is send to the client with number of free bytes, defective bytes and the total available bytes on the drive.
2	**Get_Dir**	Read directory. The directory name is send from the client to the apsolute. Then the file names or sub directories with size and time are send from the apsolute to the client.
3	**Get_File**	Read a file. The filename is send to the apsolute from the client. Then the file name, number of data blocks and the data content is send in binary form from the apsolute to the client.
4	**Put_File**	Send/ write a file. The file name, number of data blocks and the data content is send in binary form from the client to the apsolute.
5	**Del_File**	Delete file The client sends the file name to the apsolute.
6	**Get_Value**	Read the value of a variable. The client sends a definite number of the variable to the apsolute. If this corresponds to an array, the index or indexes are send to the apsolute. Then the apsolute sends the value of the variable to the client.

List of commands		
Command number	**Function**	**Description**
7	**Set_Value**	Receive and write the value of a variable. The client sends a definite number of the variable to the apsolute. If this addresses an array, the index or indexes are sent to the apsolute. In addition the client sends the value of the variable. The apsolute responds with the value of the variable set.
8	**Get_String**	Read and send the value of a string. The client sends the number of the string to the apsolute. If this indicates an array, the index or many indices are send in addition. After this the apsolute sends the content of the string to the client.
9	**Set_String**	Receive and write the value of a string. The client sends the number of the string to the apsolute. If this is indicates an array of strings, the index or many indices are send in addition. In addition the content of the string is send to the apsolute. The apsolute responds with the set value of the string to the client.
10	**Reboot**	Restart This command is send from the client to the apsolute to perform a restart (reboot) on the apsolute.

4.2 Application specific data

This chapter contains merely a list of all variables of the apsolute that can be read or written by the client with the command Get_Value and Set_Value.

A detailed description, explaining how the data area of the message is composed, is given in chapter 5.

0=Application status					
Var. No.	**Par. No**	**Status bits**	**Description**	**Bytes**	**R/W**
0	1	1(MSB)	Restart This bit is set by the apsolute after a restart/reboot. Only the client can erase this bit	2	RW
		2	Active error This bit is set by the apsolute in case of an active error. This bit is reset automatically if no active error is present.		
		3	Log book entry This bit is set by the apsolute if there is any event that is registered in the log book. Only the client can erase this bit.		
		4 to 7	Print start for respective print group This bit is set by the apsolute, as soon as the print start is performed and the next "select_message" function can be initiated by the client. Only the client can erase this bit.		
		8 to 11	Print end for respective print group This bit is set by the apsolute as soon as the print is finished. Only the client can erase this bit.		
		12 to 16	Reserved		

1=Activation of single print group					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
1	1	Number of print group	1 to 4	1	WO
	2	Activation of print group	0 = OFF 1 = ON	1	
1=Activation of more print groups					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
1	1	Number of print group (0=all)	0	1	WO
	2	Activation of the 1. print group	0 = OFF 1 = ON 255 = unchanged	1	
	3	Activation of the 2. print group		1	
	4	Activation of the 3. print group		1	
	5	Activation of the 4. print group		1	

2=Status of single print group					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
2	1	Number of print group	1 to 4	1	RO
	2	Status of print group	0 = OFF 1 = ON 2 = PRINT 3 = FAULTY	1	
2=Status of all print groups					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
2	1	Number of print group (0=all)	0	1	RO
	2	Status of 1. print group	0 = OFF 1 = ON 2 = PRINT 3 = FAULTY	1	
	3	Status of 2. print group		1	
	4	Status of 3. print group		1	
	5	Status of 4. print group		1	

3=Start/Stop print in a single print group					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
3	1	Number of print group	1 to 4	1	WO
	2	Status of print group	0 = STOP (stop print) 1 = DTOP (Start print, triggered with DTOP) 2 = PRINT ENABLE	1	
3=Start/Stop of more print groups					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
3	1	Number of print group	0	1	WO
	2	Status of 1. print group	0 = STOP (stop print) 1 = DTOP (Start print, triggered with DTOP) 2 = PRINT ENABLE 255 = unchanged	1	
	3	Status of 2. print group		1	
	4	Status of 3. print group		1	
	5	Status of 4. print group		1	

10=Ink Level of a Cartridge					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
10	1	Number of print head	1 to 4	1	RW
	2	Ink level	0 to 4000 (1/10 ml)	2	
10=Ink Level of all cartridges					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
10	1	Number of print head	0	1	RW
	2	Ink level of 1. print head	0 to 4000 (1/10 ml)	2	
	3	Ink level of 2. print head		2	
	4	Ink level of 3. print head		2	
	5	Ink level of 4. print head		2	

11=Prints Remaining of one print head					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
11	1	Number of print head	1 to 4	1	RO
	2	Number of prints remaining	0 to 999.9999	4	
11=Prints Remaining of all print heads					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
11	1	Number of print head (0=all)	0	1	RO
	2	Number of prints remaining for 1. print head	0 to 999.9999	4	
	3	Number of prints remaining for 2. print head		4	
	4	Number of prints remaining for 3. print head		4	
	5	Number of prints remaining for 4. print head		4	

12=Prints for 10 ml of one print head					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
12	1	Number of print head	1 to 4	1	RO
	2	Number of prints	0 to 999.9999	4	
12=Prints for 10 ml of all print heads					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
12	1	Number of print head (0=all)	0	1	RO
	2	Amount of prints for 1. print head	0 to 999.9999	4	
	3	Amount of prints for 2. print head		4	
	4	Amount of prints for 3. print head		4	
	5	Amount of prints for 4. print head		4	

13=Status of print head					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
13	1	Number of print head	1 to 4	1	RO
	2	Status of print head	0 = OK 1 = print head not connected 2 = Internal error 3 = Comm. error 4 = no head data available 5 = head software error	1	
13=Status of all print heads					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
13	1	Number of print head (0=all)	0	1	RO
	2	Status of 1. print head	0 = OK 1 = print head not connected 2 = Internal error 3 = Comm. error 4 = no head data available 5 = head software error	1	
	3	Status of 2. print head		1	
	4	Status of 3. print head		1	
	5	Status of 4. print head		1	

.

14=Activation of nozzle rows of one print head					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
14	1	Number of print head	1 to 4	1	RW
	2	Nozzle row of print head	0 = Left 1 = Right 2 = Toggle 3 = Both*	1	
14=Activation of nozzle rows of more print heads					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
14	1	Number of print head (0=all)	0	1	RW
	2	Nozzle row of 1. print head	0 = Left 1 = Right 2 = Toggle 3 = Both* 255 = unchanged	1	
	3	Nozzle row of 2. print head		1	
	4	Nozzle row of 3. print head		1	
	5	Nozzle row of 4. print head		1	

*not supported from software today

15=Activation of spitting function of one print head					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
15	1	Number of print head	1 to 4	1	RW
	2	Spitting option of print head	0 = OFF 1 = ON	1	
15=Activation of spitting function of more print heads					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
15	1	Number of print head (0=all)	0	1	RW
	2	Spitting option of 1. print head	0 = OFF 1 = ON 255 = unchanged	1	
	3	Spitting option of 2. print head		1	
	4	Spitting option of 3. print head		1	
	5	Spitting option of 4. print head		1	

16=Activation of ink level alarm for one print head					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
16	1	Number of print head	1 to 4	1	RW
	2	Status activation of print head	0 = OFF 1 = ON	1	
16=Activation of ink level alarm for more print heads					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
16	1	Number of print head (0=all)	0	1	RW
	2	Status activation of 1. print head	0 = OFF 1 = ON 255 = unchanged	1	
	3	Status activation of 2. print head		1	
	4	Status activation of 3. print head		1	
	5	Status activation of 4. print head		1	

17=Ink Level Alarm Limit for one print head					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
17	1	Number of print head	1 to 4	1	RW
	2	Ink level limit of print head	0 to 500 (1/10 ml)	2	
17=Ink Level Alarm for more print heads					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
17	1	Number of print head (0=all)	0	1	RW
	2	Ink level limit of 1. print head	0 to 500 (1/10 ml)	2	
	3	Ink level limit of 2. print head		2	
	4	Ink level limit of 3. print head		2	
	5	Ink level limit of 4. print head		2	

18=Activate spitting on one print head*					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
18	1	Number of print head	1 to 4	1	WO
	2	Duration of spitting	0 to 9999 (ms)	2	
18=Activate spitting on all print heads *					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
18	1	Number of print head (0=all)	0	1	WO
	2	Duration of spitting 1. print head	0 to 9999 (ms)	2	
	3	Duration of spitting 2. print head		2	
	4	Duration of spitting 3. print head		2	
	5	Duration of spitting 4. print head		2	

*Spitting is only performed when print head is not active (inactive).

19=Horizontal adjustment of one print head					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
19	1	Number of print head	1 to 4	1	RW
	2	Horizontal adjustment	-10 to 10 (dots)	1	
19=Horizontal adjustment for more print heads					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
19	1	Number of print head (0=all)	0	1	RW
	2	Horizontal adjustment of 1. print head	-10 to 10 (dots)	1	
	3	Horizontal adjustment of 2. print head		1	
	4	Horizontal adjustment of 3. print head		1	
	5	Horizontal adjustment of 4. print head		1	

20=Vertical adjustment of one print head					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
20	1	Number of print head	1 to 4	1	RW
	2	Vertical adjustment	-10 to 10 (dots)	1	
20=Vertical adjustment for more print heads					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
20	1	Number of print head (0=all)	0	1	RW
	2	Vertical adjustment of 1. print head	-10 to 10 (dots)	1	
	3	Vertical adjustment of 2. print head		1	
	4	Vertical adjustment of 3. print head		1	
	5	Vertical adjustment of 4. print head		1	

30=Counter Value					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
30	1	Counter number	1 to 10	1	RW
	2	Counter value*	-1999999999 to 1999999999	4	

31=Counter Increment					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
31	1	Counter number	1 to 10	1	RW
	2	Counter increment *	-999 to 999	2	

32=Counter Start/End Value					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
32	1	Counter number	1 to 10	1	RW
	2	Start value*	-1999999999 to 1999999999	4	
	3	End value*		4	

*The counter value is written, when loading the message which includes the counter, according to the definition of the values in the message.

40=Forward margin of one print group					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
40	1	Number of print group	1 to 4	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Forward margin	0 to 10000 (1/10mm)	2	
40=Forward margin of all print groups					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
40	1	Number of print group (0=all)	0	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Forward margin of 1. group	0 to 10000 (1/10mm)	2	
	4	Forward margin of 2. group		2	
	5	Forward margin of 3. group		2	
	6	Forward margin of 4. group		2	

41=End margin of one print group					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
41	1	Number of print group	1 to 4	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	End margin	0 to 10000 (1/10mm)	2	
41=End Margin of all print groups					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
41	1	Number of print group (0=all)	0	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	End margin of 1. group	0 to 10000 (1/10mm)	2	
	4	End margin of 2. group		2	
	5	End margin of 3. group		2	
	6	End margin of 4. group		2	

42=Print start mode of one print group*					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
42	1	Number of print group	1 to 4	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Print start mode	0 = EXTERNAL 1 = INTERNAL	1	
42=Print start mode of all print groups*					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
42	1	Number of print group	0	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Print start mode of 1. group	0 = EXTERNAL 1 = INTERNAL 255 = unchanged	1	
	4	Print start mode of 2. group		1	
	5	Print start mode of 3. group		1	
	6	Print start mode of 4. group		1	

* Start mode can only be changed if print group is not active (inactive).

43=Print speed mode of one print group*					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
43	1	Number of print group	1 to 4	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Print speed mode	0 = EXTERNAL 1 = INTERNAL	1	
43=Print speed mode of all print groups*					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
43	1	Number of print group (0=all)	0	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Print speed mode of 1. group	0 = EXTERNAL 1 = INTERNAL 255 = unchanged	1	
	4	Print speed mode of 2. group		1	
	5	Print speed mode of 3. group		1	
	6	Print speed mode of 4. group		1	

* Print speed mode can only be changed if print group is not active (inactive).

44=Production speed of one print group					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
44	1	Number of print group	1 to 4	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Production speed	0 to 300 (m/min)	2	
44=Production speed of all print groups					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
44	1	Number of print group (0=all)	0	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Production speed of 1. group	0 to 300 (m/min)	2	
	4	Production speed of 2. group		2	
	5	Production speed of 3. group		2	
	6	Production speed of 4. group		2	

45=Speed modification of one print group					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
45	1	Number of print group	1 to 4	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Speed modification	-100 to 100 (%)	2	
45=Speed modification of all print groups					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
45	1	Number of print group (0=all)	0	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Speed modification of the 1. group	-100 to 100 (%)	2	
	4	Speed modification of the 2. group		2	
	5	Speed modification of the 3. group		2	
	6	Speed modification of the 4. group		2	

46=Space between elements of one print group*					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
46	1	Number of print group	1 to 4	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Space between elements	0 to 10000 (1/10 mm)	2	
46=Space between elements for all print groups*					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
46	1	Number of print group (0=all)	0	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Space between elements of 1. group	0 to 10000 (1/10 mm)	2	
	4	Space between elements of 2. group		2	
	5	Space between elements of 3. group		2	
	6	Space between elements of 4. group		2	

* This parameter can only be changed if print group is not active (inactive).

47=Distance between prints of one print group*					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
47	1	Number of print group	1 to 4	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Distance between prints	0 to 10000 (1/10 mm)	2	
47=Distance between prints for all print groups*					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
47	1	Number of print group (0=all)	0	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Distance between prints of 1. group	0 to 10000 (1/10 mm)	2	
	4	Distance between prints of 2. group		2	
	5	Distance between prints of 3. group		2	
	6	Distance between prints of 4. group		2	

* This parameter can only be changed if the print group is not active (inactive).

48=Number of prints per object of one print group*					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
48	1	Number of print group	1 to 4	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Number of prints per object	1 to 100, 100=permanent	2	
48=Number of prints per object for all print groups*					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
48	1	Number of print group (0=all)	0	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Number of prints per object of 1. group	1 to 100, 100=permanent	2	
	4	Number of prints per object of 2. group		2	
	5	Number of prints per object of 3. group		2	
	6	Number of prints per object of 4. group		2	

* This parameter can only be changed if print group is not active (inactive).

49=Print direction of one print group					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
49	1	Number of print group	1 to 4	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Print direction	0 = NORMAL 1 = BACKWARD	1	
49=Print direction for more print groups					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
49	1	Number of print group (0=all)	0	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Print direction of 1. group	0 = NORMAL 1 = BACKWARD 255 = unchanged	1	
	4	Print direction of 2. group		1	
	5	Print direction of 3. group		1	
	6	Print direction of 4. group		1	

50=Horizontal orientation of one print group*					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
50	1	Number of print group	1 to 4	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Horizontal orientation	0 = NORMAL 1 = FORWARDS	1	
50=Horizontal orientation for more print groups*					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
50	1	Number of print group (0=all)	0	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Horizontal orientation of 1. group	0 = NORMAL 1 = FORWARDS 255 = unchanged	1	
	4	Horizontal orientation of 2. group		1	
	5	Horizontal orientation of 3. group		1	
	6	Horizontal orientation of 4. group		1	

* This parameter can only be changed if print group is not active (inactive).

51=Vertical orientation of one print group*					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
51	1	Number of print group	1 to 4	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Vertical orientation	0 = NORMAL 1 = MIRRORED	1	
51=Vertical orientation of more print groups*					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
51	1	Number of print group (0=all)	0	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Vertical orientation of 1. group	0 = NORMAL 1 = MIRRORED 255 = unchanged	1	
	4	Vertical orientation of 2. group		1	
	5	Vertical orientation of 3. group		1	
	6	Vertical orientation of 4. group		1	

* This parameter can only be changed if print group is not active (inactive).

53=Vertical resolution of one print group*					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
53	1	Number of print group	1 to 4	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Vertical resolution	0 = 600, dpi 1 = 300, dpi 2 = 300, dpi, alternate	1	
53=Vertical resolution of morel print groups*					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
53	1	Number of print group (0=all)	0	1	RW
	2	Destination of value	0 = actual parameter 1 = default parameter	1	
	3	Vertical resolution of 1. group	0 = 600, dpi 1 = 300, dpi 2 = 300, dpi, alternate 255 = unchanged	1	
	4	Vertical resolution of 2. group		1	
	5	Vertical resolution of 3. group		1	
	6	Vertical resolution of 4. group		1	

* This parameter can only be changed if print group is not active (inactive).

61=Set user level					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
61	1	User level	0 = settings 1 = service	1	RW

62=Lock keyboard					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
62	1	Status lock	0 = free 1 = locked	1	RW

70=Digital inputs					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
70	1	Input number	1 to 48	1	RO
	2	Input state	0 = input not set 1 = input set	1	

71=Digital outputs					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
71	1	Output number	1 to 48	1	RW
	2	Output state	0 = not activated 1 = activated	1	

80=Error state					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
80	1	Error state	0 = no error 1 = active errors 2 = old errors 3 = new + active errors	1	RO
	2	Number of errors	0 to max	1	

81=Error quit					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
81	1	quit new errors	0	1	WO

82= Status of Error list					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
82	1	Modification flag of the error list (*1)	0 = no modification 1 = error list modified	1	RO
	2	Number of active errors in the error list	n	1	
	3	Number of new / modified errors in the history (*2)	m	1	
	4	Number of all errors in the history (*3)	i	1	

(*1) A modification flag is set with every modification to the error list. The client-software can read out any active error with the command „10 = read active error". The modification flag is set back with Modbus variable 81 (error quit) after registration of the errors , if no new modifications take place in the error list anymore.

*2) Every new or modified entry in the error history is marked as „not read" for the remote-interface at first. The client-software can read out every new / "not read" error entry from the error history with command "20 = read new entries from error history". After the client-software has read an error entry, this is marked as "read". Thus the client-software can create its own virtually endless error history.

(*3) The client software can read out all error entries from the errror history with the command „21 = read all entries from error history".After the client-software has read an error entry, this is marked as "read".

90=Switch Single/List mode*					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
90	1	Mode	0 = single mode 1 = list mode (external message selection)	1	WO

- This parameter can only be changed if print group is not active (inactive).

91=Date / Time					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
91	1	Time	Seconds since 1.1.1970	4	RW

4.3 Application specific strings of the apsolute

This chapter merely contains a listing of all apsolute strings, which can be read and written by the client with Get_String and Set_String. A detailed description how the data of a message is composed is given in chapter 5.

1=Load print message to print*					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
1	1	Group number	1 to 4	1	WO
	2	Filename of message to print	Filename without file extension, inclusive termination Zero (0x00)	to 16	

- This parameter can only be changed if print group is not active (inactive).

2=Ink Type					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
2	1	Group number	1 to 4	1	RW
	2	Name of ink	String, including terminating Zero (0x00)	to 249	

3=Variable Text for all print groups					
Var. No.	**Par. No.**	**Description**	**Value**	**Bytes**	**R/W**
3	1	Field Name of variable text	String (fixed length), inclusive determination Zero (0x00)	20 (fixed)	WO
	2	Number of prints*	0 = permanent >0 = no. of prints	2	
	3	Variable text	String, including terminating Zero (0x00)	to 227	

The variable texts are written in a FIFO (Buffer). If the parameter "number of prints" is 0 the FIFO is not used and the variable text is printed until apsolute receives a new string (variable text).

If the number of prints is >0 the FIFO is used. In this case the parameter "number of prints" defines how often the variable text is printed before the next variable text is taken from the FIFO.

The size of the FIFO is 16 entries for each variable text set up. If the FIFO is full a buffer overflow is reported in the status of the Response Message (status 10=FIFO for variable text is full). In this case the same variable text has to be sent again after a time out.

To avoid that a print message cannot be printed, because there is no variable text in the FIFO, the transmission rate and the time between transmission have to be adjusted well, so that the FIFO is permanently filled.

Attention: with **"variable text for all print groups"** all variable texts with the declared text name are searched and set in all print groups. This will cause problems if the printing of all these groups is not synchronized. When they are not synchronized several FIFO will overflow and be free again at different times. Therefore it is recommended to use the "**variable text for a single print group",** which writes directly into the FIFO of one print group. The "variable text for a single print group" is described in the following.

4 = Variable text for a single print group				
Var. Nr.	**Description**	**Value**	**Bytes**	**R/W**
1	Group number	1…4	1	WO
2	Amount of prints	0 = permanent >0 = no. of prints	2	
3	Sequence number	Continuous sequence number	2	
4	Text name	String incl. the terminating 0 (*1).	20 set	
5	Variable text	String incl. the terminating 0 (*2).	To 200	

With this function all variable texts with the declared text name will only be searched and set in the assigned print group, respective written into the FIFO.

Attention: the sequence number is a continuous number which has to be incremented by the client after every successful transmission of the variable text. If apsolute receives 2 messages with variable texts and the same sequence number in succession the last variable text is not accepted. This sequence number insures that the same text is not written into the FIFO twice, if repeating of the message should occur because of mistakes in the MODBUS protocol layer. In this case the status reports the value 0= "no error" and the value 0 for the "amount of written strings" is reported back.

5 = Text name of variable texts				
Var. Nr.	**Description**	**Value**	**Bytes**	**R/W**
1	Group number	1….4	1	RO
2	Number of the variable text	0….N	1	
3	Text name	String incl. the terminating 0 (*1)	20 set	

With this function the names of the variable texts loaded in one print group can be detected. The number of the variable text is a continuous number of the texts in the order they are stored in the apsolute. If no variable text is stored with the number the request is responded to with status 11 = illegal value. As soon as this status is reported, the end of the list of variable texts is reached. Therefore if the loaded print message does not contain any variable text, the number 0 is already answered with status 11 = illegal value.

(*1) If the text name including the terminating 0 is < 20 bytes, the remaining characters are not evaluated.

(*2) The terminating 0 of the variable text is added to the data length and has to be transmitted.

10 = Read active error				
Var. Nr.	**Description**	**Value**	**Bytes**	**R/W**
1	Entry number	0…n	1	RO
2	Timestamp	Seconds since 1.1.1970	4	
3	Error number	Error number	2	
4	Entry status	1 = active error 2 = old error 3 = new + active error	1	
5	Entry text	String incl. the terminating 0 (*1)	X	

(*1) The terminating 0 of the string is added to the data length.

This function allows the active error to be read from the error list. The entry number is a continuous number of the new entry in the error list. If no more new error entries are stored with the number the request is responded to with status "11 = illegal value". As soon as this status is reported, the end of the error list is reached. Therefore if the error list does not contain any error entry, the number 0 is already answered with status "11 = illegal value".

20 = Read new entries from error history				
Var. Nr.	**Description**	**Value**	**Bytes**	**R/W**
1	Entry number	0…n	1	RO
2	Timestamp	Seconds since 1.1.1970	4	
3	Error number	Error number	2	
4	Entry status	1 = active error 2 = old error 3 = new + active error	1	
5	Entry text	String incl. the terminating 0 (*1)	X	

(*1) The terminating 0 of the string is added to the data length.

With this function the new error entries to the error history can be read. The entry number is a continuous number of the new entry in the history. If no more new error entries are stored with the number the request is responded to with status "11 = illegal value". As soon as this status is reported, the end of the error list is reached. Therefore if the error list does not contain any error entry, the number 0 is already answered with status "11 = illegal value".

21 = Read all entries from error history				
Var. Nr.	**Description**	**Value**	**Bytes**	**R/W**
1	Entry number	0…n	1	RO
2	Timestamp	Seconds since 1.1.1970	4	
3	Error number	Error number	2	
4	Entry status	1 = active error 2 = old error 3 = new + active error	1	
5	Entry text	String incl. the terminating 0 (*1)	X	

(*1) The terminating 0 of the string is added to the data length.

With this function the error entries to the error history can be read. The entry number is a continuous number of the entry in the history list. If no more error entries are stored with the number the request is responded to with status "11 = illegal value". As soon as this status is reported, the end of the error list is reached. Therefore if the error list does not contain any error entry, the number 0 is already answered with status "11 = illegal value".

5 Description of the application specific data transfer

This chapter describes the structure of all messages with the function code 101 as well as the procedure of the application specific data transmission.

The chapter contains the formal composition of the Request and the Response and gives an example for an application. This example refers to MODBUS RTU. Because of this, the Slave Address and the CRC checksum are included. With MODBUS TCP the MBAP header has to be added instead of that.

In the table for the request- and response-telegrams the reference data is underlayed in grey.

The transmission of complex data types (16-bit-word and 32-bit-word) always starts with the data byte of higher value.

5.1 Read drive information

This message is send from the client to the apsolute to ask for the drive information

Drive information, request				
Range	**Par. No.**	**Description**	**Value**	**Bytes**
Message-Header	1	Command number	1, **Get_Volumeinfo**	1
	2	Status	0 to 10	1
	3	Identifier		2
Data	1	Drive number	0 – Drive A, int. Flash 1 – Drive B, CMOS-Drive 2 – Drive C, int CF 3 – Drive D, ext. CF	1

On request the apsolute sends the following message to the client:

Drive information, Response				
Range	**Par. No.**	**Description**	**Value**	**Bytes**
Message-Header	1	Command number	1, **Get_Volumeinfo**	1
	2	Status	0 to 9	1
	3	Identifier		2
Data	1	Drive number	0 – Drive A, int. Flash 1 – Drive B, CMOS-Drive 2 – Drive C, int CF 3 – Drive D, ext. CF	1
	2	Total size	X	4
	3	Free memory	Y	4
	4	Defect bytes	Z	4

When the apsolute can not read the drive information or when the drive is not present, apsolute sends the response message back without any data. The status parameter of the header contains information about the reason for the error (see description of application specific message header).

5.1.1 Example for reading drive information

With the following message the drive information of the drive internal CF drive is inquired.

Example: Request, Drive information		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave Address
Function code	0x65	Function code 101
Command number	0x01	Get_Volumeinfo
Status	0x00	Status
Identifier Hi	0x00	Identifier
Identifier Lo	0x00	
Number of drive	0x02	2 – drive C, int. CF
CRC Lo	0xBE	CRC
CRC Hi	0x34	

With the following message the drive information of the drive "C" is reported back.

Example: Response, Drive information		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave Address
Function code	0x65	Function code 101
Command number	0x01	Get_Volumeinfo
Status	0x00	Status
Identifier Hi	0x00	
Identifier Lo	0x00	
Number of drive	0x02	2 – drive C, int. CF
1. Byte, total size Hi	0x07	Total size = 127 029 248 byte
2. Byte	0x9F	
3. Byte	0xE8	
4. Byte, total size Lo	0x00	
5. Byte, free memory Hi	0x07	Free memory = 123 900 928 byte
6. Byte	0x89	
7. Byte	0xB0	
8. Byte, free memory Lo	0x00	
9. Byte, defective bytes Hi	0x00	Defective bytes = 0 byte
10. Byte	0x00	
11. Byte	0x00	
12. Byte, defective bytes Lo	0x00	
CRC Lo	0xBF	
CRC Hi	0xA6	

5.2 Read Directory information

This message sends the client to get the content of a directory from a drive. In the first request the Query index = 0 (First). When the response index from the apsolute is one more then the query index, there is more information available and the client should ask again with the query index equal response index. When the response index from the apsolute is equal to the query index, then no more file and directory information are available.

Folder information, request				
Range	**Par. No.**	**Description**	**Value**	**Bytes**
Message-Header	1	Command number	2, **Get_Dir**	1
	2	Status	0	1
	3	Identifier		2
Data	1	Query index	0 = First >0 = Next (sequential number)	2
	2	Folder name incl. drive	String, including terminating Zero (0x00)	to 247

On this request apsolute sends the following response:

Folder Information, Response				
Range	**Par. No.**	**Description**	**Value**	**Bytes**
Message-Header	1	Command number	2, **Get_Dir**	1
	2	Status	0 to 9	1
	3	Identifier		2
Data	1	Response index	Response index = query index + 1 ➔ more information available or Response index = query index ➔ last data set, no more data available	2
	2	Attributes	File- or directory attributes- bit coded: 0x01 – read-only 0x02 – hidden 0x04 – system 0x10 – directory 0x20 – archive	1
	3	File size	Byte	4
	4	Date	Time in seconds since 1.1.1970	4
	5	File- or subdirectory name	String, including terminating Zero (0x00)	to 96

When the drive or directory name is not present, apsolute sends an response message without data back. The status parameter of the header contains the information about the reason of the problem (see description of application specific message header).

5.2.1 Example for read folder information

Example: Request, Folder information (1. request)		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave Address
Function code	0x65	Function code 101
Command number	0x02	Get_Dir
Status	0x00	Status
Identifier Hi	0x00	Identifier
Identifier Lo	0x00	
Query index Hi	0x00	Query index
Query index Lo	0x03	
Folder name 1. byte	0x63	c:/messages/
Folder name 2. byte	0x3A	
Folder name 3. byte	0x2F	
Folder name 4. byte	0x6D	
Folder name 5. byte	0x65	
Folder name 6. byte	0x73	
Folder name 7. byte	0x73	
Folder name 8. byte	0x61	
Folder name 9. byte	0x67	
Folder name 10. byte	0x65	
Folder name 11. byte	0x73	
Folder name 12. byte	0x2F	
Folder name 13. byte	0x00	Zero character
CRC Lo	0xBD	CRC
CRC Hi	0x1D	

Example: Response, Folder Information (response index=request index+1, free entity)		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave Address
Function code	0x65	Function code 101
Command number	0x02	Get_Dir
Status	0x00	Status
Identifier Hi	0x00	
Identifier Lo	0x00	
Query index Hi	0x00	Query index
Query index Lo	0x04	
Attributes	0x20	0x20 – Archive
1. Byte, file size Hi	0x00	File Size = 679 Byte
2. Byte, file size	0x00	
3. Byte, file size	0x02	
4. Byte, file size Lo	0xA7	
1. Byte, date Hi	0x47	Date = 26.10.2007 18:46:22
2. Byte, date	0x22	
3. Byte, date	0x35	
4. Byte, date Lo	0xFE	
1. Byte	0x5A	Sub folder / file name:
2. Byte	0x5F	Z_APS_WPNT.MSG
3. Byte	0x41	
4. Byte	0x50	
5. Byte	0x53	
6. Byte	0x5F	
7. Byte	0x57	
8. Byte	0x50	
9. Byte	0x4E	
10. Byte	0x54	
11. Byte	0x2E	
12. Byte	0x4D	
13. Byte	0x53	

Example: Response, Folder Information (response index=request index+1, free entity)		
Range	**Value (Hex)**	**Description**
14. Byte	0x47	Sub folder / file name
15. Byte	0x00	Zero character
CRC Lo	0x19	
CRC Hi	0x3C	

Note: the first filename in every folder is "."

Example: Request, Folder Information (2. request)		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x02	Get_Dir
Status	0x00	Status
Identifier Hi	0x00	Identifier
Identifier Lo	0x01	
Query index Hi	0x00	Query index
Query index Lo	0x01	
Folder name 1. byte	0x63	c:/messages/
Folder name 2. byte	0x3A	
Folder name 3. byte	0x2F	
Folder anme 4. byte	0x6D	
Folder name 5. byte	0x65	
Folder name 6. byte	0x73	
Folder name 7. byte	0x73	
Folder name 8. byte	0x61	
Folder name 9. byte	0x67	
Folder name 10. byte	0x65	
Folder name 11. byte	0x73	
Folder name 12. byte	0x2F	

Example: Request, Folder Information (2. request)		
Range	**Value (Hex)**	**Description**
Folder name 13. byte	0x00	Zero character
CRC Lo	0x7F	CRC
CRC Hi	0x4F	

Example: Response, Folder Information (response index=2, free entity)		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x02	Get_Dir
Status	0x00	Status
Identifier Hi	0x00	
Identifier Lo	0x01	
Response index Hi	0x00	Response index
Response index Lo	0x02	
Attributes	0x10	0x10 – Directory
1. Byte, file size Hi	0x00	File size = 0 byte
2. Byte, file size	0x00	
3. Byte, file size	0x00	
4. Byte, file size Lo	0x00	
1. Byte, date Hi	0x47	Date
2. Byte, date	0x42	
3. Byte, date	0xA6	
4. Byte, date Lo	0x8D	
1. Data byte	0x2E	.
2. Data byte	0x2E	.
3. Data byte	0x00	Zero character
CRC Lo	0x8B	
CRC Hi	0xF1	

Example: Request, Folder information (3. request)		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x02	Get_Dir
Status	0x00	Status
Identifier Hi	0x00	Identifier
Identifier Lo	0x02	
Request index Hi	0x00	Request index
Request index Lo	0x02	
Folder name 1. byte	0x63	c:/messages/
Folder name 2. byte	0x3A	
Folder name 3. byte	0x2F	
Folder name 4. byte	0x6D	
Folder name 5. byte	0x65	
Folder name 6. byte	0x73	
Folder name 7. byte	0x73	
Folder name 8. byte	0x61	
Folder name 9. byte	0x67	
Folder name 10. byte	0x65	
Folder name 11. byte	0x73	
Folder name 12. byte	0x2F	
Folder name 13. byte	0x00	Zero character
CRC Lo	0x3C	CRC
CRC Hi	0xBD	

Example: Response. Folder information (Response index=request index, no free entity)		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x02	Get_Dir
Status	0x00	Status
Identifier Hi	0x00	
Identifier Lo	0x02	
Response index Hi	0x00	Response index
Response index Lo	0x02	
Attributes	0x20	0x20 – File, with archive-attribute
1. Byte, file size Hi	0x00	File size = 632 byte
2. Byte, file size	0x00	
3. Byte, file size	0x02	
4. Byte, file size Lo	0x78	
1. Byte, date Hi	0x45	Date
2. Byte, date	0x23	
3. Byte, date	0xB5	
4. Byte, date Lo	0xCB	
1. Data byte	0x41	APS_npnt.MSG
2. Data byte	0x50	
3. Data byte	0x53	
4. Data byte	0x5F	
5. Data byte	0x6E	
6. Data byte	0x70	
7. Data byte	0x6E	
8. Data byte	0x74	
9. Data byte	0x2E	
10. Data byte	0x4D	
11. Data byte	0x53	
12. Data byte	0x47	
13. Data byte	0x00	Zero character

Example: Response. Folder information (Response index=request index, no free entity)		
Range	**Value (Hex)**	**Description**
CRC Lo	0x54	
CRC Hi	0x1B	

5.3 Read File

This message sends the client to the apsolute to request the content of a file. In the request the file name and the block number is transmitted. The file content is transmitted in blocks of 128 Bytes. When on the last request the number of data is less then 128 Bytes, the number of transmitted bytes is send back

Read File, Request				
Range	**Par. No.**	**Description**	**Value**	**Bytes**
Message-Header	1	Command number	3, **Get_File**	1
	2	Status	0	1
	3	Identifier		2
Data	1	Block number	x	2
	2	File name	String, including terminating Zero (0x00)	to 96

On request the apsolute sends the following message to the client:

Read File, Response				
Range	**Par. No.**	**Description**	**Value**	**Bytes**
Message-Header	1	Command number	3, **Get_File**	1
	2	Status	0 to 13	1
	3	Identifier		2
Data	1	Block number	x	2
	2	Number of transmitted bytes	0 = EOF (end of file) or 1 to 128	1
	3	Block content		to 128

If the file is not available or there is a read error, apsolute sends the response message without data back. The status parameter of the header contains the information about the reason of the problem (see description of application specific message header).

5.3.1 Example for Datatransmission, Read File

Example: Read File, Request (Blocknumber = 0)		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x03	Get_File
Status	0x00	Status
Identifier Hi	0x00	Identifier
Identifier Lo	0x03	
Blocknumber Hi	0x00	Blocknumber
Blocknumber Lo	0x00	
File name 1. byte	0x64	d:/messages/text.msg
File name 2. byte	0x3A	
File name 3. byte	0x2F	
File name 4. byte	0x6D	
File name 5. byte	0x65	
File name 6. byte	0x73	
File name 7. byte	0x73	
File name 8. byte	0x61	
File name 9. byte	0x67	
File name 10. byte	0x65	
File name 11. byte	0x73	
File name 12. byte	0x2F	
File name 13. byte	0x74	
File name 14. byte	0x65	
File name 15. byte	0x78	
File name 16. byte	0x74	
File name 17. byte	0x2E	
File name 18. byte	0x6D	
File name 19. byte	0x73	
File name 20. byte	0x67	

Example: Read File, Request (Blocknumber = 0)		
Range	**Value (Hex)**	**Description**
File name 21. Byte	0x00	Zero character
CRC Lo		CRC
CRC Hi		

Example: Read File, Response (Blocknumber = 0)		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x03	Get_File
Status	0x00	Status
Identifier Hi	0x00	
Identifier Lo	0x03	
Blocknumber Hi	0x00	Blocknumber
Blocknumber Lo	0x00	
Amount bytes read	0x80	128 Byte read
128 Data bytes		128 Bytes data with the block content
CRC Lo		
CRC Hi		

5.4 Write file

This message is send from the client to the apsolute to write data into a file. The request contains the file name, block number, number of bytes to write and the data content. The data content is transmitted in blocks of 128 bytes. The number of data bytes indicates the number of bytes to write. The block content should be 128 even for the last block.

Example: Write File, Request				
Range	**Par. No.**	**Description**	**Value**	**Bytes**
Message-Header	1	Command number	4, **Put_File**	1
	2	Status	0	1
	3	Identifier		2
Data	1	Block number	X	2
	2	Number of data bytes to write	128, Normal < 128, last block	1
	3	Block content		128
	4	File name	String, including terminating Zero (0x00)	to 96

On request apsolute sends the following message to the client:

Example: Write File, Response				
Range	**Par. No.**	**Description**	**Value**	**Bytes**
Message-Header	1	Command number	4, **Put_File**	1
	2	Status	0 to 13	1
	3	Identifier		2
Data	1	Block number	x	2
	2	No. of written bytes	0 to 128	1

If the file to write can not be opened, the apsolute sends an response message back with no. of written bytes = 0. The status parameter of the header contains information about the reason for the error (see description of application specific message header).

5.4.1 Example for Datatransmission, Write File

Example: Write File, Request (Blocknumber = 0)		
Range	**Value(Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x04	Put_File
Status	0x00	Status
Identifier Hi	0x00	Identifier
Identifier Lo	0x04	
Blocknumber Hi	0x00	Blocknumber
Blocknumber Lo	0x00	
Amount data bytes	0x80	128 Byte
128 Data bytes		128 Bytes Data with block content
File name 1. byte	0x63	c:/messages/text.msg
File name 2. byte	0x3A	
File name 3. byte	0x2F	
File name 4. byte	0x6D	
File name 5. byte	0x65	
File name 6. byte	0x73	
File name 7. byte	0x73	
File name 8. byte	0x61	
File name 9. byte	0x67	
File name 10. byte	0x65	
File name 11. byte	0x73	
File name 12. byte	0x2F	
File name 13. byte	0x74	
File name 14. byte	0x65	
File name 15. byte	0x78	
File name 16. byte	0x74	

Example: Write File, Request (Blocknumber = 0)		
Range	**Value(Hex)**	**Description**
File name 17. byte	0x2E	
File name 18. byte	0x6D	
File name 19. byte	0x73	
File name 20. byte	0x67	
File name 21. byte	0x00	Zero character
CRC Lo		CRC
CRC Hi		

Example: Write File, Response (Blocknumber = 0)		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x04	Put_File
Status	0x00	Status
Identifier Hi	0x00	
Identifier Lo	0x04	
Blocknumber Hi	0x00	Blocknumber
Blocknumber Lo	0x00	
Amount bytes read	0x80	128 Byte written
CRC Lo		
CRC Hi		

5.5 Read variable

This message is send from the client to the apsolute to ask for the value of a variable

Read variable, Request				
Range	**Par. No.**	**Description**	**Value**	**Bytes**
Message-Header	1	Command number	6, **Get_Value**	1
	2	Status	0	1
	3	Identifier		2
Data	1	Quantity of variables	n, limited by maximal length of data area 240 byte	1
	1. Var	Var.no. 1. variable	44 (Production speed)	1
		Par.no. 1. variable	0 (All groups)	1
			0 (Active parameter)	1
	2. Var.	Var.no. 2. variable	10 (Ink level)	1
		Par.no. 2. variable	1 (First printhead)	1
	n. Var.	Var.no. n. variable	91 (Date / time)	1

The amount of variables, variable-code number and parameter are examples

On request apsolute sends the following response:

Read variable, Response				
Range	**Par. No.**	**Description**	**Value**	**Bytes**
Message-Header	1	Command number	6, **Get_Value**	1
	2	Status	0 bis 13	1
	3	Identifier		2
Daten	1	Quantity of variables	n, limited by maximal length of the data area 240 byte	1
	1. Var	Var.no. 1. variable	44 (Production speed)	1
		Par.no. 1. variable	0 (All groups)	1
			0 (Active parameter)	1
		Data 1. variable	10	2
			20	2
			30	2
			40	2
	2. Var	Var.no. 2. variable	10 (Ink level)	1
		Par.no. 2. variable	1 (First printhead)	1
		Data 2. variable	420 [1/10 ml]	2
	n. Var.	Var.no. n. variable	91 (Date / time)	1
		Data n. variable	1234567890	4

The amount of variables, variable-code number and parameter are examples

5.5.1 Example for Datatransmission, Read variable

Request, Forward Margin		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x06	Get_Value
Status	0x00	Status
Identifier Hi	0x00	Identifier
Identifier Lo	0x00	
Number of variables	0x02	Number of variables
Var.No. 1. variable	0x28	Forward margin
Par.No. 1. of 1. variable	0x00	Group number, all groups
Par.No. 2. of 1. variable	0x00	Active values
Var.No. 2. variable	0x29	Endtext margin
Par.No. 1. of 2. variable	0x00	Group number, all groups
Par.No. 2. of 2. variable	0x00	Active values
CRC Lo	0x67	CRC
CRC Hi	0x4E	

Example: Response, Forward and Endtext Margin		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Comand number	0x06	Get_Value
Status	0x00	
Identifier Hi	0x00	
Identifier Lo	0x00	
Number of variables	0x02	Number of variables
Var.no. 1. variable	0x28	Forward margin
Par.no.1 of 1. variable	0x00	Group number, all groups
Par.no.2 of 1. variable	0x00	Active values
1. Byte	0x00	
2. Byte	0x32	
3. Byte	0x00	
4. Byte	0x32	
5. Byte	0x00	
6. Byte	0x32	
7. Byte	0x00	
8. Byte	0x32	
Var.no. 2. Variable	0x29	Endtext margin
Par.no. 1of 2. Variable	0x00	Group number, all groups
Par.no. 2 of 2. Variable	0x00	Active values
1. Byte	0x00	
2. Byte	0x32	
3. Byte	0x00	
4. Byte	0x32	
5. Byte	0x00	
6. Byte	0x32	
7. Byte	0x00	
8. Byte	0x32	

Example: Response, Forward and Endtext Margin		
Range	**Value (Hex)**	**Description**
CRC Lo	0xEE	
CRC Hi	0x8B	

Example: Request, Counter Parameter		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x06	Get_Value
Status	0x00	Status
Identifier Hi	0x00	Identifier
Identifier Lo	0x00	
Number of variables	0x03	3 Variables
Var.no. 1. variable	0x1E	30: Counter value
Par.no. 1 of 1. variable	0x01	Counter number, 1
Var.no. 2. variable	0x1F	31: Counter increment value
Par.no. 1 of 2. variable	0x01	Counter number, 1
Var.no. 3. variable	0x20	32: Counter limits
Par. no. 1 of 3. variable	0x01	Counter number, 1
CRC Lo	0x10	CRC
CRC Hi	0x04	

Example: Response, Counter parameter		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x06	Get_Value
Status	0x00	
Identifier Hi	0x00	
Identifier Lo	0x00	
Number of variables	0x03	3 Variables
Var.no. 1. variable	0x1E	30: Counter value
Par.no. 1 of 1. variable	0x01	Counter number, 1
1. Data byte, Hi	0x00	Counter value = 5
2. Data byte	0x00	
3. Data byte	0x00	
4. Data byte, Lo	0x05	
Var.no. 2. variable	0x1F	31: Counter increment value
Par.no. 1 of 2. variable	0x01	Counter number, 1
1. Data byte, Hi	0x00	Counter increment = 1
2. Data byte, Lo	0x01	
Var.no. 3. variable	0x20	32: Counter limits
Par.no. 1 of 3. variable	0x01	Counter number, 1
1. Data byte, Hi	0x00	Counter start value = 0
2. Data byte	0x00	
3. Data byte	0x00	
4. Data byte, Lo	0x00	
5. Data byte, Hi	0x00	Counter end value = 9
6. Data byte	0x00	
7. Data byte	0x00	
8. Data byte, Lo	0x09	
CRC Lo	0xB0	
CRC Hi	0x93	

Example: Request, Status of Error list		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x06	Get_Value
Status	0x00	Status
Identifier Hi	0x00	Identifier
Identifier Lo	0x00	
Number of variables	0x01	Number of variables
Var. No. variable	0x52	Status of error list

Example: Response, Status of Error list		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x06	Get_Value
Status	0x00	
Identifier Hi	0x00	
Identifier Lo	0x00	
Number of variables	0x01	Number of variables
Var. No.	0x52	Status of error list
Modification flag	0x01	Error list is modified
Number of active errors	0x02	2 active errors
Number of new errors in history	0x00	
Number of all errors in history	0x00	

5.6 Write variable

This message is sent from the client to the apsolute in order to write the variables

Write variable, Request				
Range	**Par. No.**	**Description**	**Value**	**Bytes**
Message-Header	1	Command number	7, **Set_Value**	1
	2	Status	0	1
	3	Identifier		2
Data	1	Quantity of variables	n, limited by maximal length of data area 240 bytes	1
	1. Variable	Var.No. 1. variable	44 (Production speed)	1
		Par.No. 1 of 1. variable	0 (all groups)	1
			0 (active parameter)	1
		Data of variable	10	2
			20	2
			30	2
			40	2
	2. Var.	Var.no. 2. variable	17 (Ink level alarm)	1
		Par.no. 1 of 2. variable	1 (First printhead)	1
		Data of variable	5 (ml)	2
	n. Var.	Var.no. n. variable	91 (date / time)	1
		Data of variable	1234567890	4

The amount of variables, variable-code number, parameter and data are examples

The composition of the data structure is explained in the chapter „read variable".

On request apsolute sends the following response to the client:

Variable write, Response				
Range	**Par. No.**	**Description**	**Value**	**Bytes**
Message-Header	1	Command number	7, **Set_Value**	1
	2	Status	0 to 13	1
	3	Identifier		2
Data	1	Quantity of written variables	N	1

When there is an error in writing the variables, apsolute sends backa response without data. The status parameter of the header contains the information about the error code (see description of application specific message header).

5.6.1 Examples of datatransmission for write variable

Example:Request, activate print group		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x07	Set_Value
Status	0x00	Status
Identifier Hi	0x00	Identifier
Identifier Lo	0x00	
Number of variables	0x01	Number of variables
Var.No. 1. variable	0x01	Activate print group
Par.No. 1 of 1. variable	0x00	Group number, all groups
1. Data byte	0x01	Activate 1. print head group
2. Data byte	0xFF	2. print head group unchanged
3. Data byte	0xFF	3. print head group unchanged
4. Data byte	0xFF	4. print head group unchanged
CRC Lo	0xB6	CRC
CRC Hi	0xFF	

Example: Response, activate print head group		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x07	Set_Value
Status	0x00	No error
Identifier Hi	0x00	
Identifier Lo	0x00	
Number of variables	0x01	Number of set variables
CRC Lo	0x76	
CRC Hi	0x35	

Example: Request, start printing		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x07	Set_Value
Status	0x00	Status
Identifier Hi	0x00	Identifier
Identifier Lo	0x00	
Number of variables	0x01	Number of variables
Var.no. 1. variable	0x03	Start print head group
Par.no. 1 of 1. variable	0x01	Group number, 1. Group
1. Data byte	0x01	1. D-Top, print once
CRC Lo	0xC6	CRC
CRC Hi	0xDA	

Example: Response, start printing		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x07	Set_Value
Status	0x00	No error
Identifier Hi	0x00	
Identifier Lo	0x00	
Number of variables	0x01	Number of set variables
CRC Lo	0x76	
CRC Hi	0x35	

Example: Request, activate and start print head group		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x07	Set_Value
Status	0x00	Status
Identifier Hi	0x00	Identifier
Identifier Lo	0x00	
Number of variables	0x02	Number of variables
Var. no. 1. variable	0x01	Activate print head group
Par.no. 1 of 1. variable	0x01	Group number, 1. group
1. Data byte	0x01	Activate 1. print head group
Var.no. 2. variable	0x03	Start print head group
Par.no. 1 of 2. variable	0x01	Print head group, 1. group
1. Data byte	0x02	2. Print enable, start print continuously
CRC Lo		CRC
CRC Hi		

Example: Response, activate and start print head group		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x07	Set_Value
Status	0x00	No error
Identifier Hi	0x00	
Identifier Lo	0x00	
Number of variables	0x02	Number of set variables
CRC Lo		
CRC Hi		

5.7 Read String

This command is send from the client to the apsolute to read the content of a string.

Read String, Request				
Range	**Par. No.**	**Description**	**Value**	**Bytes**
Message-Header	1	Command number	8, **Get_String**	1
	2	Status	0	1
	3	Identifier		2
Data	1	Number of strings	n, limited by maximum length of data area 240 bytes	1
	1. String	Var.No. 1. string	2 (ink identification)	1
		Number of data bytes	1	1
		Par.no 1. string	1 (print head number)	1
	n. String	Var.no. n. strings	2 (ink identification)	1
		Number of data bytes	1	1
		Par.no. n. string	2 (print head number)	1

The amount of strings, string identification and parameter are examples

On request apsolute sends the following response to the client:

Read String, Response				
Range	**Par. No.**	**Description**	**Value**	**Bytes**
Message-Header	1	Command number	8, **Get_String**	1
	2	Status	0 bis 13	1
	3	Identifier		2
Data	1	Quantity of strings	n, limited by maximal length of the data area 240 byte	1
	1. Sting	Var.No. 1. string	2 (ink identification)	1
		Number of data bytes	8	1
		Par.no. 1. string	1 (print head number)	1
		Data of string	WC30BK	6
		Termination	0	1
	n. Sting	Var.no. n. string	2 (ink identification)	1
		Number of data bytes	8	1
		Par.no. n. string	2 (print head number)	1
		Data of string	EC01BK	6
		Termination	0	1

The amount of strings, string identification, parameter and data are examples

5.8 Write string

This message is send by the client to write a string in the apsolute.

Write String, Request				
Range	**Par. No.**	**Description**	**Value**	**Bytes**
Message-Header	1	Command number, Set_String	9	1
	2	Status	0	1
	3	Identifier		2
Data	1	Number of Strings	n, limited by maximum length of data area 240 bytes	1
	1. String	Var. no. 1. String	1 (File name of message)	1
		Quantity of data bytes	10	1
		Par.no. 1. string	1 (Group number)	1
		Data of string	aps_npnt	8
		Termination	0	1
	n. String	Var. no. n. string	1 (file name of message)	1
		Quantity of data bytes	10	1
		Par.no. n. string	2 (Group number)	1
		Data of string	aps_wpnt	8
		Termination	0	1

The amount of strings, string identification, parameter and data are examples

The composition of the data structure is explained in the chapter „read string"

On request apsolute sends the following message

Write String, Response				
Range	**Par. No.**	**Description**	**Value**	**Bytes**
Message-Header	1	Command number	9, **Set_String**	1
	2	Status	0 to 13	1
	3	Identifier		2
Data	1	Quantity of written strings	n	1

In case of an error when writing a string, apsolute sends an response without data. The status parameter of the header contains information about the reason of this error (see description of application specific message header).

The variable texts using the FIFO buffer (parameter "number of prints" > 0) may only be transmitted separately. Because of the necessity to acknowledge the sent variable text (e.g. status byte of Response telegram = 10, FIFO buffer full) this condition has to be kept in all cases.

5.8.1 Examples for write string

Example: Load message vtext for printing, Request		
Data Area	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x09	Set_String
Status	0x00	Status
Identifier Hi	0x00	Identifier
Identifier Lo	0x00	
Number of variables	0x02	Number of strings
Var. no. 1.String	0x01	1: File name
Number of data bytes	0x07	7 Bytes will follow
Par.no. 1. String	0x01	1. Print group
1. Data byte	0x76	V
2. Data byte	0x74	T
3. Data byte	0x65	E
4. Data byte	0x78	X
5. Data byte	0x74	T
6. Data byte	0x00	Zero character
Var.no. 2.String	0x01	1: File name
Number of data bytes	0x07	7 Bytes will follow
Par.no. 2. String	0x02	2. Print group
1. Data byte	0x76	V
2. Data byte	0x74	T
3. Data byte	0x65	E
4. Data byte	0x78	X
5. Data byte	0x74	T
6. Data byte	0x00	Zero character
CRC Lo		CRC
CRC Hi		

Example: load message for printing, Response		
Data Area	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x09	Set_String
Status	0x00	No error
Identifier Hi	0x00	
Identifier Lo	0x00	
Number of variables	0x02	Number of set variables
CRC Lo		
CRC Hi		

Example: Transmit variable text, Request		
Data Area	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x09	Set_String
Status	0x00	Status
Identifier Hi	0x00	Identifier
Identifier Lo	0x00	
Number of variables	0x01	Number of strings
Var.No.of 1.string	0x03	3: variable text
Number of data bytes	0x1D	29 Bytes will follow
Name 1. Byte	0x76	v, element name of variable text
Name 2. Byte	0x74	t
Name 3. Byte	0x65	e
Name 4. Byte	0x78	x
Name 5. Byte	0x74	t
Name 6. Byte	0x00	Zero terminated
Name 7. Byte	0x00	
Name 8. Byte	0x00	
Name 9. Byte	0x00	
Name 10. Byte	0x00	
Name 11. Byte	0x00	
Name 12. Byte	0x00	
Name 13. Byte	0x00	
Name 14. Byte	0x00	
Name 15. Byte	0x00	
Name 16. Byte	0x00	
Name 17. Byte	0x00	
Name 18. Byte	0x00	
Name 19. Byte	0x00	
Name 20. Byte	0x00	

1. Parameter Hi	0x00	Number of prints, 0: permanent
1. Parameter Lo	0x00	
1. Data byte	0x35	5, content of variable text
2. Data byte	0x35	5
3. Data byte	0x36	6
4. Data byte	0x36	6
5. Data byte	0x37	7
6. Data byte	0x37	7
7. Data byte	0x00	Zero terminated
CRC Lo	0xFE	CRC
CRC Hi	0xFC	

Example: Transmit variable text, Response		
Data area	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Functioncode	0x65	Function code 101
Command number	0x09	Set_String
Status	0x00	No error
Identifier Hi	0x00	
Identifier Lo	0x00	
Number of variables	0x01	Number of set variables
CRC Lo	0x1F	
CRC Hi	0xF4	

Example: Request, Read active error		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x08	Get_String
Status	0x00	Status
Identifier Hi	0x00	Identifier
Identifier Lo	0x00	
Number of strings	0x01	Number of strings
Var. No. string	0x0A	Read active error
Number of data bytes	0x01	1 byte will follow
Entry number	0x00	Start with latest new error

Example: Response, Read active error		
Range	**Value (Hex)**	**Description**
Address field	0x01	Slave address
Function code	0x65	Function code 101
Command number	0x08	Get_String
Status	0x00	
Identifier Hi / Lo	0x00 0x00	
Number of string	0x01	Number of string
Var. No.	0x0A	Read active error
Number of data bytes	0x20	32 bytes will follow
Next entry number	0x01	Next error with entry number 1 is available
Time stamp	0x49 0xDD 0xF4 0x83	Time stamp: 09.04.2009 13:13:39
Error number	0x13 0xB4	Error number 5044
Error status	0x03	new + active error
Error text	0x50 0x72 0x69 0x6E 0x74 0x68 0x65 0x61 0x64 0x20 0x31 0x20 0x6E 0x6F 0x74 0x20 0x70 0x72 0x65 0x73 0x65 0x6E 0x74 0x00	Printhead 1 not present