LOVATO ELECTRIC S.P.A.
24020 GORLE (BERGAMO) ITALIA
VIA DON E. MAZZA, 12
TEL. 035 4282111
TELEFAX (Nazionale): 035 4282200
TELEFAX (International): +39 035 4282400
Web www.LovatoElectric.com
E-mail info@LovatoElectric.com

I454IGB06 17

(D)

SOFTSTARTER

ADXL

MODBUS® KOMMUNIKATIONSPROTOKOLL

ANHANG

(GB)

SOFT STARTER

ADXL

MODBUS COMUNICATION PROTOCOL®

ADDENDUM

CE

MODBUS® PROTOKOLL

Die Softstarter der ADXL-Serie unterstützen die Kommunikationsprotokolle Modbus RTU®, Modbus ASCII® und Modbus TCP®.
Um die Kommunikation zu ermöglichen, müssen die Softstarter ADXL mit dem RS-485-Kommunikations-kartencode EXC1042 (separat zu erwerben) ausgestattet sein.

Dank dieser Funktion ist es möglich, den Status der Geräte zu lesen und sie über eine Überwachungs-software von Lovato Electric (Synergy oder Xpress), eine Standardsoftware von Drittanbietern (SCADA) oder andere intelligente Geräte, die über eine Modbus®-Schnittstelle verfügen (z.B. eine SPS).

PARAMETEREINSTELLUNGEN
Um das Modbus®-Protokoll zu konfigurieren, rufen Sie das Menü 01-SETUP auf und wählen Sie das Menü P08-KOMMUNIKATION: Es kann nur ein Kommunikationsport konfiguriert werden.

Serielle Kommunikation

PAR	Funktion	Standard	Bereich
08.01	*Index Koten*	*1*	*1-255*
08.02	*Baudrate*	*9600*	*1200 2400 4800 9600 19200 38400 57600 115200*
08.03	*Datenformat*	*8 bit – n*	*8 bit, no par. 8 bit, odd 8 bit, even 7 bit, odd 7 bit, even*
08.04	*Stoppbits*	*1*	*1-2*
08.05	*Protokoll*	*Modbus RTU*	*Modbus RTU Modbus ASCII Modbus TCP*

MODBUS® PROTOCOL

The ADXL series of soft starters support the communication protocols Modbus RTU®, Modbus ASCII® and Modbus TCP®.
To allow the communication the soft starters ADXL must be equipped with the RS-485 communication board code EXC1042 (to be purchased separately).

Using this function it is possible to read the device status and to control the units through Lovato Electric software (Synergy and Xpress) or third-party supervision software (SCADA) or through other intelligent devices supporting Modbus®, like PLCs.

PARAMETERS SETTING
To configure the Modbus® protocol, enter in the menu 01-SETUP and then go to the menu P08-COMMMUNICATION: it is possible to configure only one communication port.

Serial comunication

PAR	Function	Default	Range
08.01	*Serial node address*	*1*	*1-255*
08.02	*Baudrate*	*9600*	*1200 2400 4800 9600 19200 38400 57600 115200*
08.03	*Data format*	*8 bit – n*	*8 bit –no par. 8 bit, odd 8 bit, even 7 bit, odd 7 bit, even*
08.04	*Stop bits*	*1*	*1-2*
08.05	*Protocol*	*Modbus RTU*	*Modbus RTU Modbus ASCII Modbus TCP*

MODBUS® RTU PROTOKOLL

Bei Verwendung des Modbus® RTU-Protokolls setzt sich der Aufbau der Kommunikationsnachricht wie folgt zusammen:

T1 T2 T3	*Adresse (8 bit)*	*Funktion (8 bit)*	*Daten (N x 8 bit)*	*CRC (16 bit)*	*T1 T2 T3*

- *Feld „Adresse" enthält Adresse des Slave-Geräts, an die Nachricht gesendet wird*
- *Feld „Funktion" enthält Code der vom Slave auszuführenden Funktion*
- *Feld „Daten" enthält die an den Slave gesendeten Daten oder die vom Slave als Antwort auf eine Frage gesendeten Daten*
- *Feld „CRC" ermöglicht sowohl dem Master als auch dem Slave die Überprüfung auf Übertragungsfehler. Dies ermöglicht es, im Falle einer Störung auf der Übertragungsleitung die gesendete Nachricht zu ignorieren, um Probleme sowohl auf der Master- als auch auf der Slave-Seite zu vermeiden.*
- *Die Sequenz T1 T2 T3 entspricht der Zeit, in der auf dem Kommunikationsbus keine Daten ausgetauscht werden dürfen, damit die angeschlossenen Geräte das Ende einer Nachricht und den Beginn der nächsten erkennen können. Diese Zeit muss 3,5 Zeichen lang sein.*

Das Gerät misst die Zeit, die zwischen dem Empfang eines Zeichens und dem nächsten verstrichen ist. Überschreitet diese Zeit die für die Übertragung von 3,5 Zeichen erforderliche Zeit, gilt bezogen auf die eingestellte Baudrate das nächste Zeichen als Beginn einer neuen Nachricht.

MODBUS® FUNKTIONEN

Die verfügbaren Funktionen sind:

03 = Read input register	*Erlaubt Auslesen der Messwerte des Geräts*
04 = Read input register	*Erlaubt Auslesen der Messwerte des Geräts*
06 = Preset single register	*Erlaubt Schreiben von Parametern*
07 = Read exception	*Erlaubt Auslesen des Gerätestatus*
17 = Report slave ID	*Erlaubt Lesen von Geräteinformationen*

Wenn Sie beispielsweise vom ADXL-Softstarter mit der Adresse 01 den Wert der Thyristortemperatur lesen möchten, der sich an der Stelle 0FB2 Hex befindet, ist die zu sendende Nachricht wie folgt:

01	*04*	*0F*	*B1*	*00*	*02*	*22*	*F8*

Erläuterung:
01= Slave Adresse
04 = Modbus Funktion "Read Input Register"
0FB1 = Adresse des benötigten Registers (die den Wert der Thyristor-Temperatur enthält) um eins verringert
00 02 = Anzahl der zu lesenden Register ab Adresse 2D
22 F8 = CRC Prüfzahl

Die Antwort des Geräts ist wie folgt:

01	*04*	*04*	*04*	*00*	*00*	*01*	*10*	*3B*	*C3*

Dove:
01= Geräteadresse (Slave 01)
04 = Vom Master angefragte Funktion
04 = Byte-Anzahl, die vom Gerät gesendet wurde
00 00 01 10 = Temperatur als Hex-Zahl dargestellt = 272 → 27.2 °C
3B C3 = CRC Prüfzahl

FUNKTION 04: READ INPUT REGISTER

Mit Funktion 04 können Sie eine oder mehrere aufeinanderfolgende Größen im Speicher lesen. Die Adresse jeder Menge ist in Tabelle 2 aufgeführt. Gemäß dem Modbus®-Standard muss die in der Nachricht angegebene Adresse um 1 gegenüber der in der Tabelle angegebenen tatsächlichen Adresse verringert werden. Ist die angeforderte Adresse nicht in der Tabelle enthalten oder ist die Anzahl der benötigten Register größer als die erlaubte Anzahl, gibt das Gerät eine Fehlermeldung zurück (siehe Fehlertabelle).

MODBUS® RTU PROTOCOL

If one selects the Modbus® RTU protocol, the communication message has the following structure:

T1 T2 T3	*Address (8 bit)*	*Function (8 bit)*	*Data (N x 8 bit)*	*CRC (16 bit)*	*T1 T2 T3*

- *The Address field holds the serial address of the slave destination device.*
- *The Function field holds the code of the function that must be executed by the slave.*
- *The Data field contains data sent to the slave or data received from the slave in response to a query.*
- *The CRC field allows the master and slave devices to check the message integrity. If a message has been corrupted by electrical noise or interference, the CRC field allows the devices to recognize the error and thereby to ignore the message.*
- *The T1 T2 T3 sequence corresponds to a time in which data must not be exchanged on the communication bus to allow the connected devices to recognize the end of one message and the beginning of another. This time must be at least 3.5 times the time required to send one character.*

The device measures the time that elapses from the reception of one character and the following. If this time exceeds the time necessary to send 3.5 characters at the selected baudrate, then the next character will be considered as the first of a new message.

MODBUS® FUNCTIONS

The available functions are:

03 = Read input register	*Allows to read the device measures.*
04 = Read input register	*Allows to read the device measures.*
06 = Preset single register	*Allows writing parameters*
07 = Read exception	*Allows to read the device status*
17 = Report slave ID	*Allows to read information about the device.*

For instance, to read the value of the temperature of the thyristors, which resides at location 0FB2 Hex, from the ADXL with serial address 01, the message to send is the following:

01	*04*	*0F*	*B1*	*00*	*02*	*22*	*F8*

Where:
01= Slave address.
04 = Modbus® function 'Read input register'.
0FB1 = Address of the required register (which contains the value of the thyristors temperature) decreased by one.
00 02 = Number of registers to be read beginning from address 2D.
22 F8 = CRC Checksum.

The device answer is the following:

01	*04*	*04*	*00*	*00*	*01*	*10*	*3B*	*C3*

Where:
01 = device address (Slave 01).
04 = Function requested by the master.
04 = Number of bytes sent by the device.
00 00 01 10 = Hex value of the temperature = 272 → 27.2 °C.
3B C3 = CRC checksum.

FUNCTION 04: READ INPUT REGISTER

The Modbus® function 04 allows to read one or more consecutive registers from the slave memory. The address of each measure is given in the table 2. As for Modbus® standard, the address in the query message must be decreased by one from the effective address reported in the table. If the measure address is not included in the table or the number of requested registers exceeds the acceptable max number, the device will return an error code (see error table).

Anforderung Master:

Slave-Adresse	*01h*
Funktion	*04h*
MSB Registeradresse	*00h*
LSB Registeradresse	*0Fh*
MSB Registernummer	*00h*
LSB Registernummer	*08h*
LSB CRC	*C1h*
MSB CRC	*CFh*

Im o.g. Beispiel werden on Slave 01 insgesamt 8 aufeinander folgende Register ab Adresse 10h angefordert. Dann werden die Register von 10h bis 17h zurückgemeldet. Der Befehl endet wie üblich mit dem CRC-Prüfsummenwert.

Rückmeldung Slave:

Slave-Adresse	*01h*
Funktion	*04h*
Byte-Anzahl	*10h*
MSB Register 10h	*00h*
LSB Register 10h	*00h*
----------	*----*
MSB Register 17h	*00h*
LSB Register 17h	*00h*
LSB CRC	*55h*
MSB CRC	*2Ch*

Die Antwort setzt sich immer aus der Slave-Adresse, der vom Master angeforderten Funktion und den benötigten Registerdaten zusammen. Die Antwort endet immer mit dem CRC-Prüfsummenwert.

FUNKTION 06: PRESET SINGLE REGISTER
Mit dieser Funktion können Sie in die Register schreiben. Beispielsweise ist es möglich, die Setup-Parameter einzustellen. Liegt der eingestellte Wert nicht innerhalb der Minimal- und Maximalwerte der Tabelle, antwortet das Gerät mit einer Fehlermeldung. Wird ein Parameter von einer nicht vorhandenen Adresse angefordert, wird dieser mit einer Fehlermeldung beantwortet.

Anforderung Master:

Slave-Adresse	*01h*
Funktion	*06h*
MSB Registeradresse	*4Fh*
LSB Registeradresse	*FFh*
MSB Daten	*00h*
LSB Daten	*01h*
LSB CRC	*6Eh*
MSB CRC	*EEh*

Rückmeldung Slave: *die Antwort ist ein Echo der Frage, d.h. die Adresse der zu ändernden Daten und der neue Wert des Parameters werden an den Master gesendet.*

FUNKTION 07: READ EXCEPTION STATUS
Mit dieser Funktion können Sie den Status des Softstarters auslesen.

Anforderung Master:

Slave-Adresse	*01h*
Funktion	*07h*
LSB CRC	*41h*
MSB CRC	*E2h*

Die folgende Tabelle zeigt die Bedeutung des vom ADXL als Antwort gesendeten Bytes:

BIT	BEDEUTUNG
0	*Start Eingang geschlossen*
1	*Stop Eingang geschlossen*
2	*Sperrbedingung aktiv*
3	*Startbedingung vorhanden*
4	*Startbedingung vorhanden + Start aktiv*
5	*Alarm aktiv*
6	*Motor in Betrieb*
7	*Globaler Alarmausgang aktiv*
8	*Bypass aktiv*

Master query:

Slave address	*01h*
Function	*04h*
MSB address	*00h*
LSB address	*0Fh*
MSB register number	*00h*
LSB register number	*08h*
LSB CRC	*C1h*
MSB CRC	*CFh*

In the above example, slave 01 is requested for 8 consecutive registers beginning with address 10h. Thus, registers from 10h to 17h will be returned. As usual, the message ends with the CRC checksum.

Slave response:

Slave address	*01h*
Function	*04h*
Byte number	*10h*
MSB register 10h	*00h*
LSB register 10h	*00h*
----------	*----*
MSB register 17h	*00h*
LSB register 17h	*00h*
LSB CRC	*55h*
MSB CRC	*2Ch*

The response is always composed of the slave address, the function code requested by the master and the contents of the requested registers. The answer ends with the CRC.

FUNCTION 06: PRESET SINGLE REGISTER
This function allows to write in the registers. For instance, it is possible to change setup parameters. If the value is not in the correct range, the device will answer with an error message. In the same way, if the parameter address is not recognised, the device will send an error response.

Master message*:*

Indirizzo slave	*01h*
Funzione	*06h*
MSB Indirizzo registro	*4Fh*
LSB Indirizzo registro	*FFh*
MSB Dato	*00h*
LSB Dato	*01h*
LSB CRC	*6Eh*
MSB CRC	*EEh*

Slave response: *the slave response is an echo to the query, that is the slave sends back to the master the address and the new value of the variable.*

FUNCTION 07: READ EXCEPTION STATUS
This function allows to read the status of the power factor controller.

Master query:

Slave address	*01h*
Function	*07h*
LSB CRC	*41h*
MSB CRC	*E2h*

The following table gives the meaning of the status byte sent by the ADXL as answer:

BIT	MEANING
0	*Start input closed*
1	*Stop input closed*
2	*Block condition activated*
3	*Start conditions present*
4	*Start conditions present + start active*
5	*Alarm active*
6	*Motor running*
7	*Global alarm output active*
8	*Bypass active*

FUNKTION 17: REPORT SLAVE ID
Mit dieser Funktion können Sie den Gerätetyp identifizieren.

Anforderung Master.

Slave-Adresse	*01h*
Funktion	*11h*
LSB CRC	*C0h*
MSB CRC	*2Ch*

Rückmeldung Slave:

Slave-Adresse	*01h*
Funktion	*11h*
Byte-Zähler	*08h*
Daten 01 (Typ) ❶	*01h*
Daten 02 (Software-Revision)	*02h*
Daten 03 (Hardware-Revision)	*00h*
Daten 04 (Parameter-Revision)	*00h*
Daten 05 (Gerätetyp) ❷	*05h*
Daten 06 (reserviert)	*00h*
Daten 07 (reserviert)	*00h*
Daten 08 (reserviert)	*00h*
LSB CRC	*86h*
MSB CRC	*88h*

❶
1 - 01h = ADXL0030600
2 - 02h = ADXL0045600
3 - 03h = ADXL0060600
4 - 04h = ADXL0075600
5 - 05h = ADXL0085600
6 - 06h = ADXL0115600
7 - 07h = ADXL0135600
8 - 08h = ADXL0162600
9 - 09h = ADXL0195600
10 - 0Ah = ADXL0250600
11 – 0Bh = ADXL0320600

❷
5 – 05h= Serie AXDL

FEHLERMELDUNGEN
Erhält der Slave eine falsche Nachricht, signalisiert er dem Master den Zustand, indem er mit einer Nachricht bestehend aus der angeforderten Funktion in ODER mit 80 Hex gefolgt von einem Fehlercode antwortet. Die folgende Tabelle zeigt die vom Slave an den Master gesendeten Fehlercodes:

TABELLE 1: FEHLERCODES

CODE	FEHLER
01	*Ungültige Funktion*
02	*Ungültige Registeradresse*
03	*Parameter außerhalb des gültigen Bereichs*
04	*Vorgang kann nicht ausgeführt werden*
06	*Slave beschäftigt, Funktion aktuell nicht verfügbar*

FUNZIONE 17: REPORT SLAVE ID
This function allows to identify the device type.

Master query.

Slave address	*01h*
Function	*11h*
LSB CRC	*C0h*
MSB CRC	*2Ch*

Slave response:

Slave address	*01h*
Function	*11h*
Byte count	*08h*
*Data 01 (type)*❶	*01h*
Data 02 (software revision)	*02h*
Data 03 (hardware revision)	*00h*
Data 04 (parameter revision)	*00h*
Data 05 (type of device) ❷	*05h*
Data 06 (reserved)	*00h*
Data 07 (reserved)	*00h*
Data 08 (reserved)	*00h*
LSB CRC	*86h*
MSB CRC	*88h*

❶
1 - 01h = ADXL0030600
2 - 02h = ADXL0045600
3 - 03h = ADXL0060600
4 - 04h = ADXL0075600
5 - 05h = ADXL0085600
6 - 06h = ADXL0115600
7 - 07h = ADXL0135600
8 - 08h = ADXL0162600
9 - 09h = ADXL0195600
10 - 0Ah = ADXL0250600
11 – 0Bh = ADXL0320600

❷
5 – 05h= AXDL series

ERRORS
In case the slave receives an incorrect message, it answers with a massage composed by the queried function ORed with 80 Hex, followed by an error code byte. In the following table are reported the error codes sent by the slave to the master:

TABLE 1: ERROR CODES

CODE	ERROR
01	*Invalid function*
02	*Invalid address*
03	*Parameter out of range*
04	*Function execution impossible*
06	*Slave busy, function momentarily not available*

MODBUS® ASCII PROTOKOLL

Das Modbus®-ASCII-Protokoll wird normalerweise in Anwendungen verwendet, die eine Kommunikation über ein Modem erfordern. Die zur Verfügung stehenden Funktionen und Adressen sind die gleichen wie bei der RTU-Version, jedoch werden die Zeichen in ASCII übertragen und die Beendigung der Nachricht erfolgt nicht rechtzeitig, sondern mit Carriage Return-Zeichen.
Wenn Parameter 08.05 als Modbus®-ASCII-Protokoll ausgewählt ist, sieht der Aufbau der Kommunikationsnachricht auf dem entsprechenden Kommunikationsport wie folgt aus:

:	Adresse 2 chars	Funktion 2 chars	Daten (N chars)	LRC 2 chars	CR LF

- *Das Adressfeld enthält die Adresse des Slave-Geräts, an das die Nachricht gesendet wird.*
- *Das Funktionsfeld enthält den Code der vom Slave auszuführenden Funktion.*
- *Das Datenfeld enthält die an den Slave gesendeten Daten oder die vom Slave als Antwort auf eine Frage gesendeten Daten.*
- *Das LRC-Feld ermöglicht sowohl dem Master als auch dem Slave die Überprüfung auf Übertragungsfehler. Dies ermöglicht es, im Falle einer Störung auf der Übertragungsleitung die gesendete Nachricht zu ignorieren, um Probleme sowohl auf der Master- als auch auf der Slave-Seite zu vermeiden.*
- *Die Nachricht endet immer mit den Steuerzeichen CRLF (0D 0A).*

Beispiel:

Wenn Sie beispielsweise den Temperaturwert am Standort 0FB2h aus dem Gerät mit der Adresse 01 auslesen möchten, lautet die zu sendende Nachricht:

:	01	04	0F	B1	00	02	39	CRLF

Erklärung:
: = ASCII 3Ah = Nachrichten-Starttrennzeichen
01 = Slave-Adresse
04 = Modbus® Funktion "Read input register"
0F B1 = Standortadresse (Temperatur) um eine Einheit verringert
00 02 = Anzahl der zu lesenden Register ab Adresse 04
39 = LRC-Prüfsumme
CRLF = ASCII 0Dh 0Ah = Nachrichten-Endtrennzeichen

Die Antwort des Geräts ist wie folgt:

:	01	04	04	00	00	00	FF	F8	CRLF

Erklärung:
: = ASCII 3Ah = Nachrichten-Starttrennzeichen
01 = ADXL-Adresse (Slave 01).
04 = Vom Master benötigte Funktion
04 = Anzahl der vom Slave gesendeten Bytes
00 00 00 FF = Hexadezimalwert der gemessenen Temperatur (=25.5°C)
F8 = LRC-Prüfsumme
CRLF = ASCII 0Dh 0Ah = Nachrichten-Endtrennzeichen

MODBUS® ASCII PROTOCOL

The Modbus® ASCII protocol is normally used in application that require to communicate through a couple of modems. The functions and addresses available are the same as for the RTU version, but the transmitted characters are in ASCII and the message end is delimited by Carriage return / Line Feed instead of a transmission pause. If one selects the parameter 08.05 as Modbus® ASCII protocol, the communication message on the correspondent communication port has the following structure:

:	Address (2 chars)	Function (2 chars)	Dates (N chars)	LRC (2 chars)	CR LF

- *The Address field holds the serial address of the slave destination device.*
- *The Function field holds the code of the function that must be executed by the slave.*
- *The Data field contains data sent to the slave or data received from the slave in response to a query.*
- *The LRC field allows the master and slave devices to check the message integrity. If a message has been corrupted by electrical noise or interference, the LRC field allows the devices to recognize the error and thereby ignore the message.*
- *The message terminates always with CRLF control character (0D 0A).*

Example:

For instance, to read the value of the temperature, which resides at location 0FB2h from the slave with serial address 01, the message to send is the following:

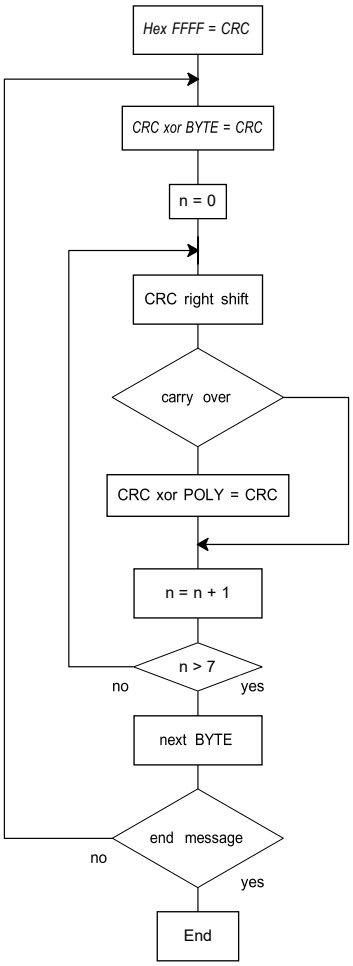
:	01	04	0F	B1	00	02	39	CRLF

Whereas:
: = ASCII 3Ah message start delimiter
01 = slave address.
04 = Modbus® function 'Read input register'
0F B1 = Address of the required register (temperature) decreased by one.
00 02 = Number of registers to be read beginning from address 04.
39 = LRC Checksum.
CRLF = ASCII 0Dh 0Ah = Message end delimiter

The answer of the device is the following:

:	01	04	04	00	00	00	FF	F8	CRLF

Where:
: = ASCII 3Ah message start delimiter.
01 = ADXL address (Slave 01).
04 = Function requested by the master.
04 = Number of bytes sent by the slave.
00 00 00 FF = Hex value of the measured temperature (=25.5°C).
F8 = LRC checksum
CRLF = ASCII 0Dh 0Ah = Message end delimiter.



BERECHNUNG VON CRC (PRÜFSUMME für RTU)
Beispiel der CRC-Berechnung:
Frame = 0207h

Initialisierung CRC	*1111*	*1111*	*1111*	*1111*	
Erstes Byte laden			*0000*	*0010*	
xor mit erstem frame-Byte ausführen	*1111*	*1111*	*1111*	*1101*	
1. right shift ausführen	*0111*	*1111*	*1111*	*1110*	**1**
Carry=1, Lastpolynom	*1010*	*0000*	*0000*	*0001*	
xor mit dem Polynom ausführen	*1101*	*1111*	*1111*	*1111*	
2. right shift ausführen	*0110*	*1111*	*1111*	*1111*	**1**
Carry=1, Lastpolynom	*1010*	*0000*	*0000*	*0001*	
xor mit dem Polynom ausführen	*1100*	*1111*	*1111*	*1110*	
3. right shift ausführen	*0110*	*0111*	*1111*	*1111*	**0**
4. right shift ausführen	*0011*	*0011*	*1111*	*1111*	**1**
Carry=1, Lastpolynom	*1010*	*0000*	*0000*	*0001*	
xor mit dem Polynom ausführen	*1001*	*0011*	*1111*	*1110*	
5. right shift ausführen	*0100*	*1001*	*1111*	*1111*	**0**
6. right shift ausführen	*0010*	*0100*	*1111*	*1111*	**1**
Carry=1, Lastpolynom	*1010*	*0000*	*0000*	*0001*	
xor mit dem Polynom ausführen	*1000*	*0100*	*1111*	*1110*	
7. right shift ausführen	*0100*	*0010*	*0111*	*1111*	**0**
8. right shift ausführen	*0010*	*0001*	*0011*	*1111*	**1**
Carry=1, Lastpolynom	*1010*	*0000*	*0000*	*0001*	
Zweites Byte des frame's laden			*0000*	*0111*	
xor mit dem zweiten des frame's laden	*1000*	*0001*	*0011*	*1001*	
1. right shift ausführen	*0100*	*0000*	*1001*	*1100*	**1**
Carry=1, Lastpolynom	*1010*	*0000*	*0000*	*0001*	
xor mit dem Polynom ausführen	*1110*	*0000*	*1001*	*1101*	
2. right shift ausführen	*0111*	*0000*	*0100*	*1110*	**1**
Carry=1, Lastpolynom	*1010*	*0000*	*0000*	*0001*	
xor mit dem Polynom ausführen	*1101*	*0000*	*0100*	*1111*	
3. right shift ausführen	*0110*	*1000*	*0010*	*0111*	**1**
Carry=1, Lastpolynom	*1010*	*0000*	*0000*	*0001*	
xor mit dem Polynom ausführen	*1100*	*1000*	*0010*	*0110*	
4. right shift ausführen	*0110*	*0100*	*0001*	*0011*	**0**
5. right shift ausführen	*0010*	*0100*	*0000*	*1001*	**1**
Carry=1, Lastpolynom	*1010*	*0000*	*0000*	*0001*	
xor mit dem Polynom ausführen	*1001*	*0010*	*0000*	*1000*	
6. right shift ausführen	*0100*	*1001*	*0000*	*0100*	**0**
7. right shift ausführen	*0010*	*0100*	*1000*	*0010*	**0**
8. right shift ausführen	*0001*	*0010*	*0100*	*0001*	**0**
CRC Ergebnis	**0001**	**0010**			
0100	**0001**				
	12h	**41h**			

Beachte: *Byte 41h wird zuerst gesendet (auch wenn es das LSB ist), dann wird 12h gesendet*

LRC BERECHNUNG (PRÜFSUMME für ASCII)
Beispiel der LRC Berechnung:

Adresse	*01*	*00000001*
Funktion	*04*	*00000100*
Start-Adresse hi.	*00*	*00000000*
Start-Adresse lo.	*00*	*00000000*
Registeranzahl	*08*	*00001000*
	Summe	*00001101*
	1. Ergänzung	*11110010*
	+ 1	*00000001*
	2. Ergänzung	*11110101*
LRC Ergebnis		
		F5

CRC CALCULATION (CHECKSUM for RTU)
Example of CRC calculation:
Frame = 0207h

CRC initialization	*1111*	*1111*	*1111*	*1111*	
Load the first byte			*0000*	*0010*	
Execute xor with the first Byte of the frame	*1111*	*1111*	*1111*	*1101*	
Execute 1st right shift	*0111*	*1111*	*1111*	*1110*	**1**
Carry=1,load polynomial	*1010*	*0000*	*0000*	*0001*	
Execute xor with the polynomial	*1101*	*1111*	*1111*	*1111*	
Execute 2nd right shift	*0110*	*1111*	*1111*	*1111*	**1**
Carry=1,load polynomial	*1010*	*0000*	*0000*	*0001*	
Execute xor with the polynomial	*1100*	*1111*	*1111*	*1110*	
Execute 3rd right shift	*0110*	*0111*	*1111*	*1111*	**0**
Execute 4th right shift	*0011*	*0011*	*1111*	*1111*	**1**
Carry=1,load polynomial	*1010*	*0000*	*0000*	*0001*	
Execute xor with the polynomial	*1001*	*0011*	*1111*	*1110*	
Execute 5th right shift	*0100*	*1001*	*1111*	*1111*	**0**
Execute 6th right shift	*0010*	*0100*	*1111*	*1111*	**1**
Carry=1,load polynomial	*1010*	*0000*	*0000*	*0001*	
Execute xor with the polynomial	*1000*	*0100*	*1111*	*1110*	
Execute 7th right shift	*0100*	*0010*	*0111*	*1111*	**0**
Execute 8th right shift	*0010*	*0001*	*0011*	*1111*	**1**
Carry=1,load polynomial	*1010*	*0000*	*0000*	*0001*	
Load the second byte of the frame			*0000*	*0111*	
Execute xor with the Second byte of the frame	*1000*	*0001*	*0011*	*1001*	
Execute 1st right shift	*0100*	*0000*	*1001*	*1100*	**1**
Carry=1,load polynomial	*1010*	*0000*	*0000*	*0001*	
Execute xor with the polynomial	*1110*	*0000*	*1001*	*1101*	
Execute 2nd right shift	*0111*	*0000*	*0100*	*1110*	**1**
Carry=1,load polynomial	*1010*	*0000*	*0000*	*0001*	
Execute xor with the polynomial	*1101*	*0000*	*0100*	*1111*	
Execute 3rd right shift	*0110*	*1000*	*0010*	*0111*	**1**
Carry=1,load polynomial	*1010*	*0000*	*0000*	*0001*	
Execute xor with the polynomial	*1100*	*1000*	*0010*	*0110*	
Execute 4th right shift	*0110*	*0100*	*0001*	*0011*	**0**
Execute 5th right shift	*0010*	*0100*	*0000*	*1001*	**1**
Carry=1,load polynomial	*1010*	*0000*	*0000*	*0001*	
Execute xor with the polynomial	*1001*	*0010*	*0000*	*1000*	
Execute 6th right shift	*0100*	*1001*	*0000*	*0100*	**0**
Execute 7th right shift	*0010*	*0100*	*1000*	*0010*	**0**
Execute 8th right shift	*0001*	*0010*	*0100*	*0001*	**0**
CRC Result	**0001**	**0010**		**0100**	
	0001				
	12h	**41h**			

Note: *The byte 41h is sent first(even if it is the LSB), then12h is sent.*

LRC CALCULATION (CHECKSUM for ASCII)
Example of LRC calculation:

Address	*01*	*00000001*
Function	*04*	*00000100*
Start address hi.	*00*	*00000000*
Start address lo.	*00*	*00000000*
Number of registers	*08*	*00001000*
	Sum	*00001101*
	1. complement	*11110010*
	+ 1	*00000001*
	2. complement	*11110101*
LRC result		
		F5

TABELLE 2:
MESSDATEN DES KOMMUNIKATIONSPROTOKOLLS
(verwendbar mit den Funktionen 03 und 04)

TABLE 2:
MEASURES SUPPLIED BY SERIAL COMMUNICATION PROTOCOL
(To be used with functions 03 and 04)

ADRESSE ADDRESS	WORDS	MESSWERT	MEASURE	EINHEIT UNIT	FORMAT
06h	*2*	*Spannung L3-L1*	*L3-L1 Voltage*	*V / 100*	*Unsigned long*
08h	*2*	*Strom L1*	*L1 current*	*A / 10000*	*Unsigned long*
0Ah	*2*	*Strom L2*	*L2 current*	*A / 10000*	*Unsigned long*
0Ch	*2*	*Strom L3*	*L3 current*	*A / 10000*	*Unsigned long*
14h	*2*	*Wirkleistung L1*	*L1 active power*	*kW / 100000*	*Signed long*
16h	*2*	*Wirkleistung L2*	*L2 active power*	*kW / 100000*	*Signed long*
18h	*2*	*Wirkleistung L3*	*L3 active power*	*kW / 100000*	*Signed long*
26h	*2*	*Phasenleistungsfaktor L1*	*L1 power factor*	*/ 10000*	*Signed long*
28h	*2*	*Phasenleistungsfaktor L2*	*L2 power factor*	*/ 10000*	*Signed long*
2Ah	*2*	*Phasenleistungsfaktor L3*	*L3 power factor*	*/ 10000*	*Signed long*
32h	*2*	*Frequenz*	*Frequency*	*Hz / 1000*	*Unsigned long*
3Ah	*2*	*Gesamte Wirkleistung*	*Total active power*	*kW / 10000*	*Signed long*
40h	*2*	*Gesamter Phasenleistungsfaktor*	*Total power factor*	*/ 10000*	*Signed long*
76h	*2*	*Maximale Spannung*	*Max current*	*A / 10000*	*Unsigned long*
78h	*2*	*Drehmoment*	*Torque*	*%/ 10*	*Unsigned long*
7Ah	*2*	*Maximaler Momentanstrom %*	*Maximum instantaneous current %*	*% / 10*	*Unsigned long*
F80h	*2*	*Gesamtstundenzähler*	*Motor hour meter*	*h / 3600*	*Unsigned long*
F82h	*2*	*Startzähler (Anzahl Startvorgänge)*	*Start counter*		*Unsigned long*
F84h	*2*	*Status* ❶	*Status*		*Unsigned integer*
FB0h	*2*	*Thermischer Motorstatus*	*Motor thermal status*	*%*	*Unsigned long*
FB2h	*2*	*Thyristortemperatur*	*Thyristors temperature*	*°C / 10*	*Unsigned long*
1B20h	*4*	*Gesamte Wirkenergie*	*Total active energy*	*kWh / 100*	*Unsigned long*
2100h – bit 0	*1*	*Eingang 1*	*Input 1*	*bool*	*Unsigned integer*
2100h– bit 1	*1*	*Eingang 2*	*Input 2*	*bool*	*Unsigned integer*
2100h– bit 2	*1*	*Eingang 3*	*Input 3*	*bool*	*Unsigned integer*
2141h – bit 0	*1*	*Ausgang 1*	*Output 1*	*bool*	*Unsigned integer*
2141h – bit 1	*1*	*Ausgang 2*	*Output 2*	*bool*	*Unsigned integer*
2141h – bit 2	*1*	*Ausgang 3*	*Output 3*	*bool*	*Unsigned integer*
2180h - bit 0	*1*	*Remote 1*	*Remote 1*	*bool*	*Unsigned integer*
2180h – bit 1	*1*	*Remote 2*	*Remote 2*	*bool*	*Unsigned integer*
2180h – bit 2	*1*	*Remote 3*	*Remote 3*	*bool*	*Unsigned integer*
2180h – bit3	*1*	*Remote 4*	*Remote 4*	*bool*	*Unsigned integer*
21C0h – bit 0	*1*	*Grenzwert 1*	*Limit 1*	*bool*	*Unsigned integer*
21C0h – bit 1	*1*	*Grenzwert 2*	*Limit 2*	*bool*	*Unsigned integer*
21C0h – bit 2	*1*	*Grenzwert 3*	*Limit 3*	*bool*	*Unsigned integer*
21C0h – bit 3	*1*	*Grenzwert 4*	*Limit 4*	*bool*	*Unsigned integer*

❶ *Bedeutung des Statusbits – Meaning of status bit:*

BIT	BEDEUTUNG	MEANING
0	*Keine Leistung*	*Absence of power*
1	*Vorwärmphase*	*Preheating*
2	*Anlasser bereit*	*Starter ready*
3	*verzögerter Start*	*Start delay*
4	*Kickstart*	*Kick start*
5	*Anlauframpe*	*Acceleration ramp*
6	*Spannungsbegrenzer*	*Current limit*
7	*Drehzahlbegrenzer*	*Torque limit*
8	*Betrieb*	*Run*
9	*Bypass geschlossen*	*Bypass closed*
10	*Auslauframpe*	*Deceleration ramp*
11	*Schutz gesperrt*	*Protections inhibited*
12	*Freilauf*	*Freewheel*
13	*Alarm*	*Alarm*
14	*Motorauswahl*	*Motor selection*

TABELLE 3:
BEFEHLE
(verwendbar mit Funktion 06)

TABLE 3:
COMMANDS
(To be used with function 06)

ADRESSE ADDRESS	WORDS	BEFEHL	COMMAND	WERT VALUE	FORMAT
2FF0h	1	C01 Wartungsintervall zurücksetzen	C01 Reset maintenance service interval.	0	Unsigned int
2FF0h	1	C02 Thermischen Status zurücksetzen	C02 Thermal status reset.	1	Unsigned int
2FF0h	1	C03 Zähler für Startanzahl zurücksetzen	C03 Reset the number of startings counter.	2	Unsigned int
2FF0h	1	C04 Motorstundenzähler zurücksetzen	C04 Reset the motor's hour meter.	3	Unsigned int
2FF0h	1	C05 Energiezähler zurücksetzen	C05 Reset the energy counters.	4	Unsigned int
2FF0h	1	C06 LIM Variablen mit Speicher zurücksetzen	C06 Reset LIM variables with memory.	5	Unsigned int
2FF0h	1	C11 AUTOSET- Assistenten wiederholen	C11 Repeat the AUTOSET wizard.	❶ 10	Unsigned int
2FF0h	1	C12 Auf Werkseinstellung zurücksetzen	C12 Restore the factory default settings.	❶ 11	Unsigned int
2FF0h	1	C13 Setup-Parameter Kopie speichern	C13 Save a copy of the setup parameters.	❶ 12	Unsigned int
2FF0h	1	C14 Setup-Parameter Kopie wiederherstellen	C14 Restore a copy of the setup parameters.	❶ 13	Unsigned int
2FF0h	1	C15 Testlauf mit niedriger Motorleistung	C15 Test with low power motor.	14	Unsigned int
2FF0h	1	C16 Speicher der Ereignisliste löschen	C16 Reset the events list.	15	Unsigned int

❶ **ACHTUNG***: Nach Verwendung dieses Befehls wird empfohlen, den REBOOT-Befehl zu senden (Wert 5 mit Funktion 06 an Adresse 2F03h schreiben).*
ATTENTION*: after using of this command it is recommended to send REBOOT command (write with function 06 the value 5 at the address 2F03h).*

TABELLE 4:
EREIGNIS

TABLE 4:
EVENTS

ADRESSE *ADDRESS*	WORDS	MAßNAHME	MEASURE	EINHEIT UNIT	FORMAT
5030h	1	EVENTS POINTER zeigt letzte aufgezeichnete Ereignisse an (LSB)/ EVENTS COUNTER zeigt alle gespeicherten Ereignisse an (MSB)	EVENTS POINTER last event stored (LSB)/ EVENTS COUNTER total events stored (MSB)		Unsigned int
5032h	28	Beschreibung der Ereignisse in aktueller Sprache	Event description using current language		Unsigned int
Vorgehen zum Lesen von Ereignissen *1 - Lesen Sie 1 Register mit Funktion 04 an Adresse 5030h, das höchstwertige Byte (MSB) gibt an, wie viele Ereignisse gespeichert sind (Wert zwischen 0 und 60), das niederwertigste Byte (LSB) wird jedes Mal erhöht, wenn ein Ereignis gespeichert wird (Wert zwischen 0 und 60). Sobald 60 Ereignisse gespeichert wurden, bleibt das MSB auf 60, während das LSB auf 0 zurückkehrt und dann weiter ansteigt.* *2- Legen Sie den Index des zu lesenden Ereignisses fest (weniger als die maximale Anzahl gespeicherter Ereignisse). Dazu müssen Sie die Funktion 06 an Adresse 5030h ausführen und angeben, welches Ereignis gelesen werden soll.* *3- Lesen Sie 28 Register (mit nur einer Funktion 04) bei Adresse 5032h.* *Der zurückgemeldete Wert ist eine Zeichenfolge aus ASCII-Zeichen, die dieselbe Beschreibung des Ereignisses anzeigen, die auf der ADXL-Anzeige sichtbar ist.* *Der Index des zu lesenden Ereignisses wird nach dem Lesen des 5032h-Registers automatisch erhöht, um das Herunterladen von Ereignissen zu beschleunigen.* *4- Wenn Sie das nächste Ereignis lesen möchten, führen Sie Schritt 3 aus. Wenn Sie ein anderes Ereignis lesen möchten, führen Sie Schritt 2 aus.*			**Procedure for events reading** *1 - Perform the read of 1 register by using the function 04 at address 5030h, the most significant byte (MSB) indicates how many events are stored (value between 0 to 60), the least significant byte (LSB) is incremented each time an event is saved (value between 0 to 60). Once stored the 60 events the MSB will remain at 60 while the LSB will back to zero and after will continue to increase.* *2- Set the index of the event that you want to read (less than the maximum number of events stored), to do this perform the function 06 at 5030h, specifying which event read.* *3- Perform a read of 28 registers (with a single function 04) at address 5032h. The value returned is a string of ASCII characters, showing the same event description visible on the display of the ADXL. The index of the event to be read is incremented automatically after a reading of the register 5032h, in order to speed up the download of events* *4- If you want to read the next event performing step 3, if you want to read any other event do step 2.*		

PARAMETEREINSTELLUNGEN - ***PARAMETER SETTING***
ADXL-Parameter werden gemäß den folgenden Regeln gelesen/modifiziert. ADXL parameters are read/modified according to the following rules.

ADRESSE *ADDRESS*	WORDS	BEDEUTUNG *MEANING*	FUNKTION *FUNCTION*	BEISPIEL *EXAMPLE*
5000h	1	Auswahl Menü-Nummer Menu number selection	4 read – 6 write	Wert 1 eintragen, um Menü 1 auszuwählen. Write value 1 to select the menu number 1.
5001h	1	Auswahl Untermenü-Nummer (wenn vorhanden) Sub-menu number selection (if present)	4 read – 6 write	Wert 1 eintragen, um Untermenü 1 auszuwählen. Write value 1 to select the sub-menu number 1.
5002h	1	Auswahl Parameter-Nummer Parameter number selection	4 read – 6 write	Wert 2 eintragen, um Parameter 2 auszuwählen. Write value 2 to select the parameter number 2.
5004h	1	Parameter-Wert Parameter value	4 read – 6 write	Wert des Parameters eintragen. Insert the value of the parameter.
2F03h	1	Im Flash-Speicher abspeichern Save to flash memory	6 write	Wert=5. Value=5.

BEISPIEL / EXAMPLE

Stellen Sie den Wert von Parameter P01.04 (Beschleunigungszeit) auf 10 Sekunden ein.
Set to 10 seconds the value of the parameter P01.04 (acceleration time).

Schritt 1*: Menu 01 wählen*
Step 1*: Select menu 01*

MASTER *Funktion / Function* = 6
Adresse / Address = 5000H (5000H – 0001H =4FFFH**)**
Wert / Value = 1 (01H)

01	*06*	*4F*	*FF*	*00*	*01*	*6E*	*EE*

ADXL *Funktion / Function* = 6
Adresse / Address = 5000H (5000H – 0001H =4FFFH**)**
Wert / Value = 1 (01H)

01	*06*	*4F*	*FF*	*00*	*01*	*6E*	*EE*

(Anmerkung: In diesem Fallbeispiel ist es nicht erforderlich, das Untermenü auf Adresse 5001 einzustellen)
(Note: In this example it is not necessary to set the sub-menu number at address 5001)

Schritt 2*: Parameter 04 wählen*
Step 2*: Select parameter 04*

MASTER *Funktion / Function* = 6
Adresse / Address = 5002H (5002H – 0001H =5001H**)**
Wert / Value = 4 (04H)

01	*06*	*50*	*01*	*00*	*04*	*C8*	*C9*

ADXL *Funktion / Function* = 6
Adresse / Address = 5002H (5002H – 0001H =5001H**)**
Wert / Value = 4 (04H)

01	*06*	*50*	*01*	*00*	*04*	*C8*	*C9*

Schritt 3*: Wert 10 einstellen*
Step 3*: Set value 10*

MASTER *Funktion / Function* = 6
Adresse / Address = 5004H (5004H – 0001H =5003H**)**
Wert / Value = 10 (0AH)

01	*06*	*50*	*03*	*00*	*0A*	*E8*	*CD*

ADXL *Funktion / Function* = 6
Adresse / Address = 5004H (5004H – 0001H =5003H**)**
Wert / Value = 10 (0AH)

01	*06*	*50*	*03*	*00*	*0A*	*E8*	*CD*

Schritt 4*: Speichern und Neustart*
Step 4*: Saving and reboot*

MASTER *Funktion / Function* = 6 (06H)
Adresse / Address = 2F03H (2F03H – 0001H =2F02H**)**
Wert / Value = 5 (05H)

01	*6*	*2F*	*02*	*00*	*05*	*E0*	*DD*

ADXL *Das Gerät speichert die Parameter und führt den Neustart durch (Modbus sendet keine Antwort).*
The device saves and reboots (no Modbus response message will be received).