LinMot®

RS 232 RS 485

Documentation of the LinRS Interface for the following Controllers:

- **E1100-GP (-HC, -XC)**
- **E1100-RS (-HC, -XC)**
- **E1130-DP (-HC, -XC)**
- **B1100-GP (-HC, -XC)**
- **Series E1200**
- **Series E1400**

LinRS Interface V3.14/V4.2

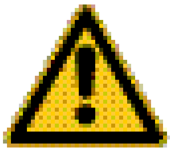
User Manual

Document version 3.12 / June 2010

Table of Content

1 System Overview

The LinMot servo controllers E1100-RS, E1100-GP, E1130-DP, the whole series E1200 and the B1100-GP support the LinRS communication profile. LinRS is a LinMot specific RS protocol to run the E1100 Servo controller over RS232, RS422 or RS485 serial links.



When running the E1100 servo controller over an RS connection with LinRS, the configuration and debugging can be done over the CAN bus link. LinMot-Talk supports an USB to CAN (Part No. 0150-3134) converter for this purpose. It is strongly recommended to use this USB to CAN converter, to avoid programming debugging disappointment.

For the installation and use of the USB to CAN converter refer to document [3].

1.1 References

Ref	Title	Source
1	User Manual Motion Control SW	www.linmot.com
2	LinMot E1100 Servo Controller Configuration over Fieldbus Interfaces	www.linmot.com
3	Installation Guide USB-CAN Converter	www.linmot.com

The documentation is distributed with the LinMot-Talk software.

1.2 Definitions, Items, Shortcuts

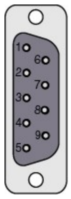
Shortcut	Meaning
LM	LinMot linear motor
OS	Operating System (Software)
MC	Motion Control (Software)
Intf	Interface (Software)
Appl	Application (Software)
UPID	Unique Parameter ID (16 bit)
CT	Command Table
Msg	Message
Cmd	Command

2 Connecting the RS bus

2.1 Pin Out of the COM Connector (X5) (Series E1100 and B1100)

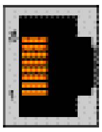
Over this connector the RS232 or the RS422/RS485 serial lines are available. This connector is available with all E1100 series servo controllers.

DSBU 9 male:



Pin 1	RS-485 Tx+	Pin 6	RS-485 Rx-
Pin 2	RS-232 TX	Pin 7	RS-485 Tx-
Pin 3	RS-232 RX	Pin 8	CAN L
Pin 4	RS-485 Rx+	Pin 9	CAN H
Pin 5	GND (100Ω)		

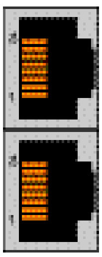
2.2 Pin Out of the System Connector (X19) (Series E1200)



Pin 1	Do not connect
Pin 2	Do not connect
Pin 3	RS232 RX
Pin 4	GND
Pin 5	GND
Pin 6	RS232 TX
Pin 7	Do not connect
Pin 8	Do not connect

2.3 Pin Out of the CMD Connector (X7, X8):

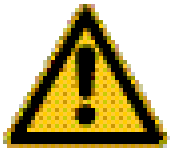
The CMD connector exists only on the E1100-RS(-HC/-XC), E1100-DP(-HC/-XC), series E1200 and B1100-GP(-HC/-XC) controllers, 2xRJ45 with 1:1 connected signals. Standard twisted pairs: 1/2, 3/6, 4/5, 7/8. Use Ethernet cables according the EIA / TIA 568A standard.



Pin 1	RS485 Rx+
Pin 2	RS485 Rx-
Pin 3	RS485 Tx+
Pin 4/5	Ground (1kΩ)
Pin 6	RS485 Tx-
Pin 7	CAN H
Pin 8	CAN L

3 Power Up Behaviour

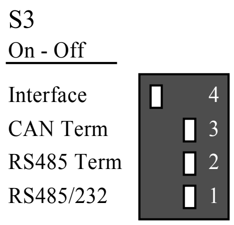
The power up behaviour can be defined over the S3 switches and the S1 and S2 hex switches and the parameter configuration. So the servo controller can be configured over the LinRS protocol. It is possible to setup completely the controller over LinRS when the LinRS software has been installed.



On the B1100 there are no switches to define the baud rate, ID and interface enable, for this reason all configuration has to be done by parameter with LinMotTalk 4 software over CAN-Bus.

3.1 Activating and Deactivating the LinRS Protocol

Over the Interface Switch on the S3.4 switches the LinRS protocol can be activated (position On) or deactivated (position Off). This switch is not available on E1200 series controllers.



If the Interface Switch S3.4 is off during the system is booting, the LinRS will be deactivated for the rest of this power cycle. In this case the servo controller can normally be accessed with the LinMot-Talk software over the RS or CAN link, for configuration and testing.

If the Interface Switch S3.4 is on during the system is booting, the LinRS protocol will be activated. The RS line of the servo controller is no longer available for the LinMot-Talk software, in this case configuring, debugging and testing can only be done over the CAN link. Now switching off the interface to off reactivates the RS line for the LinMot-Talk software, then switching on again, reinstall the LinRS protocol for the RS line, this enables some debugging capabilities without running the LinMot-Talk software over the CAN bus. It is recommended to use the USB to CAN converter, when working with the LinRS protocol for configuring and debugging.

With the RS485/RS232 switch on S3 the bus driver can be selected.

3.2 ID and Baud Rate Selection

With the default parameterization the baud rate can be selected over S1 and the ID is selected over S2.

3.2.1 Baud Rate Selection

The baud rate can be defined over the S1 hex switch (default setting) or by parameter value.

S1 Baud Rate Code Table	
S1 Value	**Selected Baud Rate**
0	Undefined Baud Rate
1	4800Bit/s
2	9600 Bit/s
3	19200 Bit/s
4	38400 Bit/s
5	57600 Bit/s
6	115200 Bit/s
7	Undefined Baud Rate
.	Undefined Baud Rate

3.2.2 ID Selection

Like the baud rate the protocol ID can be defined over the S2 hex switch (default setting), by parameter value or by the S1&S2 hex switches.

S2 ID code table	
S2 Value	**Selected MACID**
0	MACID = 00h
1	MACID = 01h
2	MACID = 02h
.	.
F	MACID = 0Fh

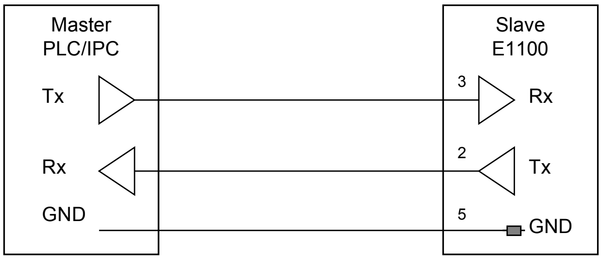
S1&S2 ID code table		
S1 Value	**S2 Value**	**Selected MACID**
0	0	MACID = 00h
0	1	MACID = 01h
0	2	MACID = 02h
.	.	.
1	0	MACID = 10h
.	.	
F	F	MACID = FFh

3.3 RS Topologies

3.3.1 RS232 Topology

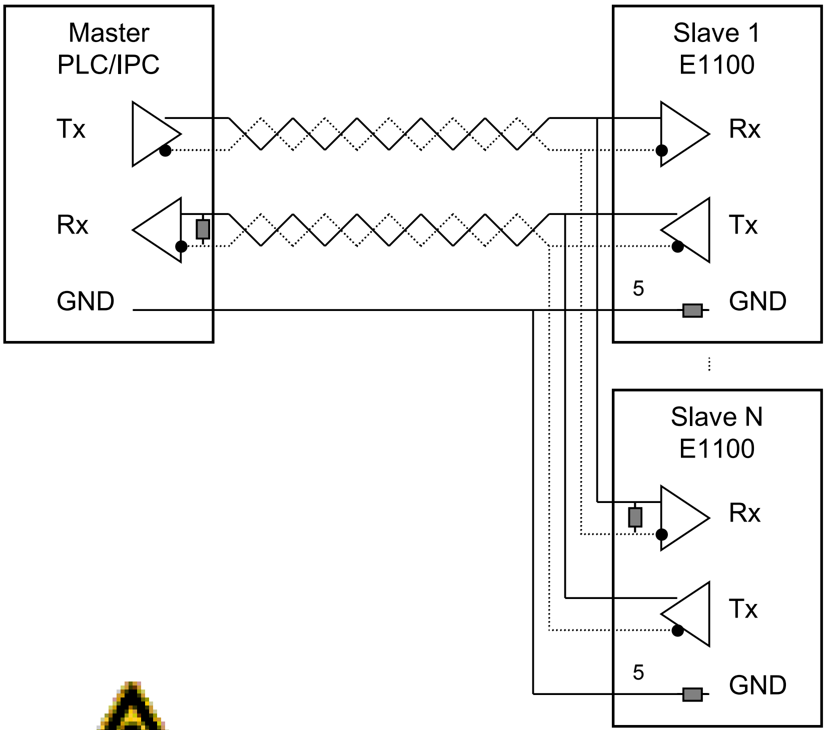
Over a RS232 line only one slave can be connected to the master, controlling several slaves needs several RS232 lines.

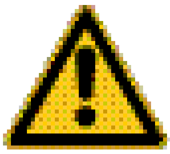
The RS232 serial lines are only on the COM connector X5 available.



3.3.2 RS422 Topology

With the RS422 topology several Slaves can be accessed. The master transmit lines are connected to all slave receives lines and all slave transmit lines are connected with master receive lines. With this topology debugging is quiet easy because master communication and slave communication is separated. The easiest way to loop through the communication lines at slave side, are over the X7/X8 RJ45 connectors with the RS/DP servo controllers. Activate RS485 Term on S3.2 (or S5.2 on E1200 series) on the last slave servo controller.

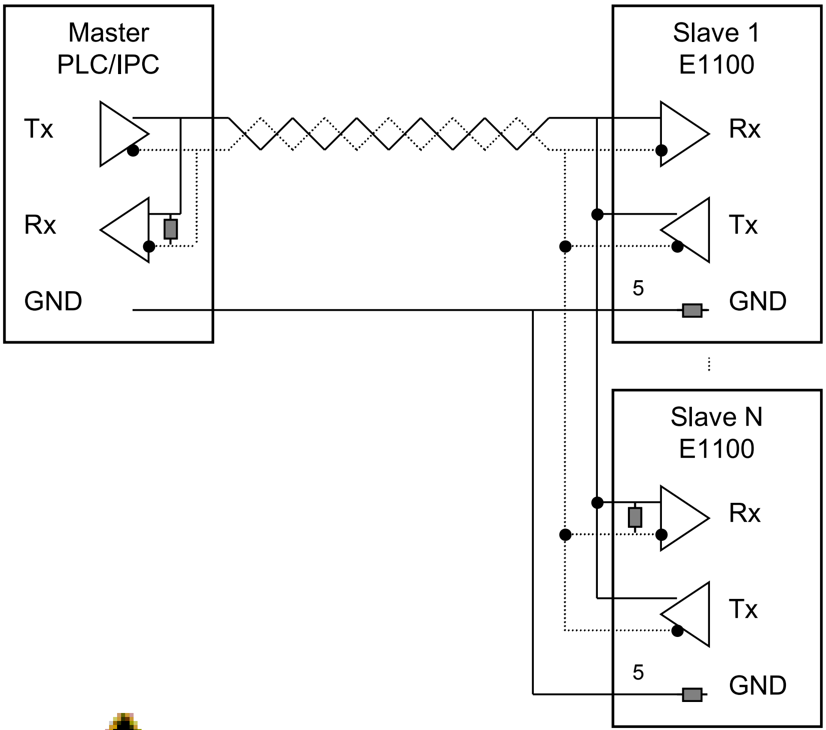


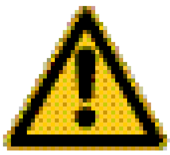


With B1100 the switch S4.1 has to be set to RS485 select!

3.3.3 RS485 Topology

With the RS485 topology also several Slaves can be accessed. The master transmit lines are connected to all slave receives lines and all slave transmit lines are connected with master receive lines. With this topology debugging is quiet easy because master communication and slave communication is separated. The easiest way to loop through the communication lines at slave side, are over the X7/X8 RJ45 connectors with the RS/DP servo controllers, but at the first slave the RS485 AB lines from the master have to be connected with the Rx **and** the Tx signal lines. Activate RS485 Term on S3.2 on the last slave servo controller.





With B1100 the switch S4.1 has to be set to RS485 select!

4 LinRS Message Format

The following tables define the principle structure of a LinRS Message.

Header				Data				Opt. Checksum		End
Start Header	ID	Len	Start Data 0	Data 1 Msg ID Low	Data 2 Msg ID High	…	Data n-1	Check-Sum Low	Check-Sum High	End Tele-gram
01h	0..FFh	2..63	**02h**	0..FFh	0..FFh	0..FFh	0..FFh	0..FFh	0..FFh	**04h**
			Length Count							
	Checksum Calculation									

Byte Nr	Name	Description	Value
0	Start Header	Fix ID at telegram start	01h
1	ID	ID of LinMot Controller	0..FFh
2	Length	Length of telegram data n	2..63
3	Start Data 0	Fix ID at telegram start	02h
4	Data 1 Msg ID Low	Message Sub ID	0..FFh
5	Data 2 Msg ID High	Message Main ID	0..FFh
6	Data 3	Message data 0	0..FFh
..	..	..	0..FFh
n+2	Data n-1	Message data n-1	0..FFh
n+3	Checksum Low	Optional checksum Low Byte	0..FFh
n+4	Checksum High	Optional checksum High Byte	0..FFh
n+5/n+3	End Telegram	Fix ID at telegram end	04h

Data are transmitted lowest byte first highest byte last. Dummy Data could be added to get a fix master transmission length.

4.1 ID

The ID is one byte that defines the address (ID) of the LinMot servo controller. This ID is defined by two Hex Switches or by a parameter.

4.2 Length

The length defines the data length in bytes.

4.3 Data

In the data fields the command specific data are transmitted.

4.4 Checksum

The checksum field is optional. Two different methods are supported:

- Byte wise addition modulo 2^16 (fast and easy)
- CRC CCITT

5 Message Data Definitions

5.1 Message Main ID

The Message object are identified in a first level by following main Message IDs

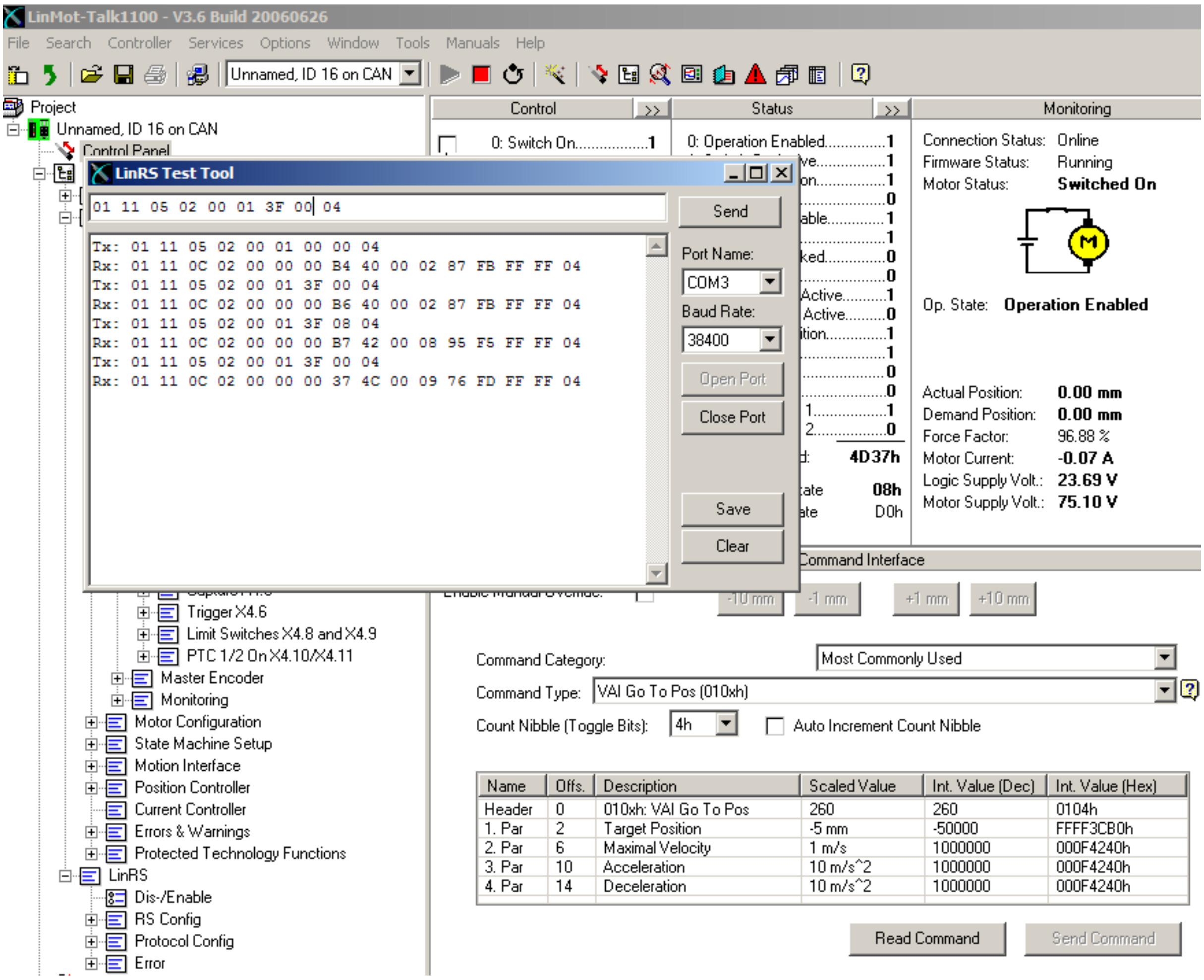
Message Main ID	B1100	Description
00h	Yes	Response Request / Response itself
01h	**Yes**	**Write Control Word**
02h	**Yes**	**Write Motion Command Interface**
03h	Yes	Parameter Message Group with default Response
04h	-	Curve Configuration Message Group
05h	-	Parameter Configuration Message Group
06h	Yes	Program Handling Message Group
07h	-	Read Error Info Message Group
08h	-	Command Table Configuration Message Group

In the easiest way of using the LinRS protocol, only the Messages with the Main IDs (0), 1 and 2 are needed to control the behavior of the servo controller.

The other Main messages IDs are only needed if configuration or setup functionality over the LinRS protocol is needed and supported. In these cases a much deeper integration of the LinMot servo controller into the superior PLC/IPC is supported and needed.

As an alternative to this, LinMot offers a configuration service, which means you can store your configuration with LinMot and order the servo controller with installed firmware and configuration (parameter and curves). In many cases this will be the more cost effective solution.

For debugging reasons and to get familiar with the LinRS protocol the LinMot-Talk has an integrated LinRS test tool (Tools\LinRS test Tool). Together with the USB to CAN converter the steps could be followed directly as shown below.



6 Response Request

The response to the response request is added to the configured response data, or set to the configured reserved space.

Message Main ID	Message Sub ID	Description
00h	00h	Default MC Response Answer (Slave)
00h	01h	Default MC Response Request (Master)
00h	02h	Default MC Response Request with Status Word Request (Master)
00h	03h	Default MC Response Request with Warn Word Request (Master)
00h	04h	Default MC Response Request with State Var Request (Master)
00h	30h	Slave Response to Master Parameter Request
00h	40h	Slave Response to Master Curve Request
00h	*50h*	*Reserved Slave Memory Group Response*
00h	60h	Slave Response to Master Program Handling Request
00h	7yh	Slave Response to Master Read Error Requests
00h	8yh	Slave Response to Master Command Table Request

Every time the controller has accepted a Message, it will respond with a message itself. Normally the response contains the configured data.

6.1 Configuration of the Default Response

The content of the default response can be configured, so the information for the normal operation can be adapted to the application needs. The order of the information is the same as they appear in the LinMot-Talk configuration tool. The Default Response is normally responded within the time >0.5ms..<1.5ms. The bold named entries are configured in default configuration (factory setting) of the LinRS firmware installation.

Name	Format	Description
Communication State	**1 bytes**	Status of communication (Checksum error,..) (Default Cfg)
Status Word	**2 bytes**	Status Word bit coded (Default Cfg)
State Var	**2 bytes**	High byte state number, low byte state depending (Default Cfg)
Error Code	2 bytes	Error Code
Warn Word	2 bytes	Warn Word bit coded
Echo MC Intf Header	2 bytes	Echo of the motion command interface header
Monitoring Channel 1	**4 bytes**	Monitoring Channel 1 Data (Default Cfg)
Monitoring Channel 2	4 bytes	Monitoring Channel 2 Data
MC Response	4 bytes	Place holder for a response request

6.2 Request Default Response Example

The following example documents a default response request, the controller will answer with the configured default response. This request could be used to monitor state changes or direct variable changes.

Request: default response

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Response node ID
2	03h	Telegram length
3	02h	Fix ID start data
4	01h	Message Sub ID (default response request)
5	00h	Message Main ID (Response)
6	04h	Fix ID telegram end

Response: Default Response

Byte Offset	Value	Description
0	01h	Fix ID telegram start
1	11h	MACID
2	0Ch	Data length
3	02h	Fix ID start data
4	00h	Sub ID: Default Response
5	00h	Main ID: Response Message
6	00h	Communication State ok
7	37h	Status Word Low Byte
8	4Ch	Status Word High Byte
9	C2h	State Var Low Byte
10	08h	State Var High Byte (MainState)
11	9Dh	Actual Position Low Word Low Byte
12	FCh	Actual Position Low Word High Byte
13	FFh	Actual Position High Word Low Byte
14	FFh	Actual Position High Word High Byte
15	04h	Fix ID telegram end

Example:
Tx: 01 11 03 02 01 00 04
Rx: 01 11 0C 02 00 00 00 37 4C C2 08 9D FC FF FF 04

6.3 Default MC Response Request with Status Word Request Example

The following example documents a Default MC Response Request with Status Word Request, the controller will answer with the configured default response and adds the Status Word in a 4Byte Container at the end of the data section.

Request: Default MC Response Request with Status Word Request

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Response node ID
2	03h	Telegram length
3	02h	Fix ID start data
4	02h	Message Sub ID (default response request)
5	00h	Message Main ID (Response)
6	04h	Fix ID telegram end

Response: Default MC Response Request with Status Word Request

Byte Offset	Value	Description
0	01h	Fix ID telegram start
1	11h	MACID
2	10h	Data length
3	02h	Fix ID start data
4	00h	Sub ID: Default Response with Status Word
5	00h	Main ID: Response Message
6	00h	Communication State ok
7	37h	Status Word Low Byte
8	4Ch	Status Word High Byte
9	C2h	State Var Low Byte
10	08h	State Var High Byte (MainState)
11	9Dh	Actual Position Low Word Low Byte
12	FCh	Actual Position Low Word High Byte
13	FFh	Actual Position High Word Low Byte
14	FFh	Actual Position High Word High Byte
15	B4h	Low Byte Status Word
16	00h	High Byte Status Word
17	00h	No Meaning
18	00h	No Meaning
19	04h	Fix ID telegram end

Examples:
Tx: 01 11 03 02 02 00 04 ; Request Default Response with Status Word
Rx: 01 11 10 02 00 00 00 B4 00 00 02 D5 6A 10 00 **B4 00** 00 00 04
Tx: 01 11 03 02 03 00 04 ; Request Default Response with Warn Word
Rx: 01 11 10 02 00 00 00 B4 00 00 02 D5 6A 10 00 **80 00** 00 00 04
Tx: 01 11 03 02 04 00 04 ; Request Default Response with State Var
Rx: 01 11 10 02 00 00 00 B4 00 00 02 D5 6A 10 00 **00 02** 00 00 04

7 Write Control Word

With the access to the control Word the main state machine could be controlled.

Message Main ID	Message Sub ID	Description
01h	00h	Write Control Word

With the following examples the first steps in runtime programming should be explained. To this message request the LinMot servo controller will always answer with the default motion response.

7.1 Write Control Word example 1

With this control word command the lock state is cleared

Request: Write Control Word (Clear Control Word)

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	05h	Telegram length
3	02h	Fix ID start data
4	00h	Message Sub ID
5	01h	Message Main ID (Write Control Word)
6	00h	Control Word Low Byte
7	00h	Control Word High Byte
8	04h	Fix ID telegram end

This request is answered with the Default Response.

Example:
Tx: 01 11 05 02 00 01 00 00 04
Rx: 01 11 0C 02 00 00 00 B4 40 00 02 AB 00 00 00 04

7.2 Write Control Word example 2

With this control word command causes a transition to enable state and starts homing.

Request: Write Control Word (Set all Bits for Operation and Home Flag)

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	05h	Telegram length
3	02h	Fix ID start data
4	00h	Message Sub ID
5	01h	Message Main ID (Write Control Word)
6	3Fh	Control Word Low Byte
7	08h	Control Word High Byte
8	04h	Fix ID telegram end

Example:
Tx: 01 11 05 02 00 01 3F 08 04
Rx: 01 11 0C 02 00 00 00 B6 40 00 02 A8 00 00 00 04

To detect when the homing sequence has finished, poll the controller until the low byte in the StateVar is 0Fh and the main state = 09h (homing) high byte of the state var.

Example:
Tx: 01 11 05 02 00 01 3F 08 04
Rx: 01 11 0C 02 00 00 00 B7 22 01 09 65 0E FB FF 04
Tx: 01 11 05 02 00 01 3F 08 04
Rx: 01 11 0C 02 00 00 00 B7 22 01 09 8B A3 F7 FF 04
Tx: 01 11 05 02 00 01 3F 08 04
Rx: 01 11 0C 02 00 00 00 B7 62 0C 09 4D 24 FF FF 04
Tx: 01 11 05 02 00 01 3F 08 04
Rx: 01 11 0C 02 00 00 00 37 4C **0F 09** DA FB FF FF 04 -> Homing finished

7.3 Write Control Word example 3

With this control word command the normal operation is enabled.

Request: Write Control Word (Set all Bits for Operation and Reset Home Flag)

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	05h	Telegram length
3	02h	Fix ID start data
4	00h	Message Sub ID
5	01h	Message Main ID (Write Control Word)
6	3Fh	Control Word Low Byte
7	00h	Control Word High Byte
8	04h	Fix ID telegram end

Example:
Tx: 01 11 05 02 00 01 3F 00 04
Rx: 01 11 0C 02 00 00 00 37 4C 00 09 3C FC FF FF 04

Poll again to make sure main state 08h is reached.

Tx: 01 11 05 02 00 01 3F 00 04
Rx: 01 11 0C 02 00 00 00 37 4C **C0 08** DA FB FF FF 04 -> ‘Operation Enabled’ state reached with homed flag set

8 Write Motion Command Interface

With the access to the Motion Command Interface of the MC-SW [1], the run time motion could be controlled. There are a lot of different motion commands, which are described in [1] for the different needs of the applications.

Message Main ID	Message Sub ID	Description
02h	00h	Write Motion Control Interface

8.1 Write Motion Command Interface example 1

With this motion command a VA-interpolator motion with default parameters for (max. velocity and acceleration and deceleration) to the target position 10mm is defined.

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	09h	Telegram length
3	02h	Fix ID start data
4	00h	Message Sub ID
5	02h	Message Main ID (Motion Command Interface)
6	01h	Motion Cmd Intf Header Low Byte (count =1) Sub ID =0
7	02h	Motion Cmd Intf Header High Byte Master ID =2
8	A0h	Target Position lowest byte
9	86h	Target Position middle low byte
10	01h	Target Position middle high byte
11	00h	Target Position highest byte
12	04h	Fix ID telegram end

Example:
Tx: 01 11 09 02 00 02 01 02 A0 86 01 00 04
Rx: 01 11 0C 02 00 00 00 37 68 A1 08 8B FC FF FF 04

8.2 Write Motion Command Interface example 2

Go back with the same motion command to 0mm.

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	09h	Telegram length
3	02h	Fix ID start data
4	00h	Message Sub ID
5	02h	Message Main ID (Motion Command Interface)
6	02h	Motion Cmd Intf Header Low Byte (count =1) Sub ID =0
7	02h	Motion Cmd Intf Header High Byte Master ID =2
8	00h	Target Position lowest byte
9	00h	Target Position middle low byte
10	00h	Target Position middle high byte
11	00h	Target Position highest byte
12	04h	Fix ID telegram end

Tx: 01 11 09 02 00 02 02 02 00 00 00 00 04
Rx: 01 11 0C 02 00 00 00 37 28 A2 08 5E 81 01 00 04

With the next example the VAI motion command with defined Position, Max Velocity, Acceleration and Deceleration is used. The message length is increased to 15h, to debug the send data push the read button in the Control Panel.

Tx: 01 11 15 02 00 02 03 01 F0 49 02 00 40 42 0F 00 40 42 0F 00 40 42 0F 00 04
Rx: 01 11 0C 02 00 00 00 37 0D D3 08 F3 49 02 00 04

Name	Offs.	Description	Scaled Value	Int. Value (Dec)	Int. Value (Hex)
Header	0	010xh: VAI Go To Pos	259	259	0103h
1. Par	2	Target Position	15 mm	150000	000249F0h
2. Par	6	Maximal Velocity	1 m/s	1000000	000F4240h
3. Par	10	Acceleration	10 m/s^2	1000000	000F4240h
4. Par	14	Deceleration	10 m/s^2	1000000	000F4240h

Read Command   Send Command

Tx: 01 11 15 02 00 02 04 01 B0 3C FF FF 40 42 0F 00 40 42 0F 00 40 42 0F 00 04
Rx: 01 11 0C 02 00 00 00 37 49 94 08 61 3D FF FF 04

Name	Offs.	Description	Scaled Value	Int. Value (Dec)	Int. Value (Hex)
Header	0	010xh: VAI Go To Pos	260	260	0104h
1. Par	2	Target Position	-5 mm	-50000	FFFF3CB0h
2. Par	6	Maximal Velocity	1 m/s	1000000	000F4240h
3. Par	10	Acceleration	10 m/s^2	1000000	000F4240h
4. Par	14	Deceleration	10 m/s^2	1000000	000F4240h

Read Command   Send Command

9 Parameter Group

With the parameter group, parameter can be changed or read. Within the MC SW two different kinds of parameters are supported:

- Live Parameters (during MC runtime, Message Sub ID's 00h and 01h)
- Configuration Parameters

While live parameters can be changed during the MC SW is running the configuration parameters affects the behavior of its SW instance only after a restart of it. A Reset command or Power cycle restarts all SW instances.

The parameters are accessed with a 16 Bit Unique Parameter ID (UPID). All parameters values are mapped in a 4 byte value memory area. With bit parameters the lowest bit of parameter value memory field is relevant, a byte parameter in the lowest byte and word parameter into the two lower bytes.

Message Main ID	Message Sub ID	B1100	Description
03h	00h	Yes	Read RAM value with MC Default Response
03h	01h	Yes	Write RAM value with MC Default Response
03h	*02h*	*Yes*	*Read ROM value with MC Default Response*
03h	*03h*	*Yes*	*Write ROM value with MC Default Response*
03h	*04h*	*-*	*Write RAM and ROM value with MC Default Response*
03h	*05h*	*-*	*Get minimal value of parameter with MC Default Response*
03h	*06h*	*-*	*Get maximal value of parameter with MC Default Response*
03h	*07h*	*-*	*Get default value of parameter with MC Default Response*
03h	*11h*	*-*	*Set OS (Operating System) ROM parameter values to default*
03h	*12h*	*-*	*Set MC (Motion Control) ROM parameter values to default*
03h	*13h*	*-*	*Set Interface ROM parameter values to default*
03h	*14h*	*-*	*Set Application ROM parameter values to default*

9.1 Parameter/Variable Read RAM example

With this command example the RAM value of the UPID 13A2h (P Gain Position Controller, for B1100 controllers it is UPID 6198h) is read. With the default MC response the requested value is added in the last 4 bytes of it.

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	05h	Telegram length
3	02h	Fix ID start data
4	00h	Message Sub ID (Read Ram Value)
5	03h	Message Main ID (Parameter)
6	A2h	UPID Low Byte (P Gain Position Controller)
7	13h	UPID High Byte (P Gain Position Controller)
8	04h	Fix ID telegram end

Tx: 01 11 05 02 00 03 A2 13 04
Rx: 01 11 10 02 00 00 00 37 4C C2 08 AA 06 00 00 **0A 00 00 00** 04

9.2 Parameter Write RAM example

With this command example the RAM value of the UPID 13A2h (P Gain Position Controller, for B1100 controllers it is UPID 6198h) is changed to 11.

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	09h	Telegram length
3	02h	Fix ID start data
4	01h	Message Sub ID (Write Ram Value)
5	03h	Message Main ID (Parameter)
6	A2h	UPID Low Byte (P Gain Position Controller)
7	13h	UPID High Byte (P Gain Position Controller)
8	0Bh	Parameter value low word low byte
9	00h	Parameter value low word high byte
10	00h	Parameter value high word low byte
11	00h	Parameter value high word high byte
12	04h	Fix ID telegram end

Tx: 01 11 09 02 01 03 A2 13 0B 00 00 00 04
Rx: 01 11 0C 02 00 00 00 37 4C C0 08 BF FB FF FF 04

Reading the parameter again shows the changed value:

Tx: 01 11 05 02 00 03 A2 13 04
Rx: 01 11 10 02 00 00 00 37 4C C0 08 BF FB FF FF **0B 00 00 00** 04

Reading the ROM value of the same parameter shows that it is still unchanged 0Ah.

Tx: 01 11 05 02 02 03 A2 13 04
Rx: 01 11 10 02 00 00 00 37 4C C0 08 BF FB FF FF **0A 00 00 00** 04

Write 12 (0Ch) to the ROM value of the same parameter.

Tx: 01 11 09 02 03 03 A2 13 0C 00 00 00 04
Rx: 01 11 0C 02 00 00 00 37 4C C2 08 AB 09 00 00 04

Reading the ROM value of the same parameter shows that it is still unchanged 0Ch. This change will affect the position controller behavior only after a restart of the MC-SW, for this reason it is recommended to change the ROM values only in the stopped MC-SW mode.

Tx: 01 11 05 02 02 03 A2 13 04
Rx: 01 11 10 02 00 00 00 37 4C C2 08 AB 09 00 00 0C 00 00 00 04

The following sequence shows the behavior of the write RAM nad ROM command (Message Sub ID 04h) to the same parameter with UPID 13A2h P-Gain Position controller Set A (UPID 6198h for B1100).

Write 09h to Ram and ROM
Tx: 01 11 09 02 04 03 A2 13 **09 00 00 00** 04
Rx: 01 11 0C 02 00 00 00 37 4C C2 08 95 09 00 00 04

Reading changed RAM value
Tx: 01 11 05 02 00 03 A2 13 04
Rx: 01 11 10 02 00 00 00 37 4C C2 08 EB 09 00 00 09 00 00 00 04

Reading changed ROM value
Tx: 01 11 05 02 02 03 A2 13 04
Rx: 01 11 10 02 00 00 00 37 4C C2 08 95 09 00 00 09 00 00 00 04

10 Parameter Configuration Group

The parameter Configuration Group Messages could be used to read out a configuration, and/or write a configuration. For configuring it is needed to stop the MC-SW of the servo controller first (Program Handling Message Group), and after configuring the controller (re)start the MC-SW again.

Message Main ID	Message Sub ID	Description
05h	00h	Read ROM value
05h	01h	Write ROM value
05h	03h	Get parameter address usage
05h	04h	Get parameter type
05h	05h	Get minimal value of parameter
05h	06h	Get maximal value of parameter
05h	07h	Get default value of parameter
05h	08h	Start Get Modified UPID List
05h	09h	Get Next Modified UPID
05h	0Ah	Start Get UPID List
05h	0Bh	Get Next UPID
05h	0Ch	Set OS (Operating System) ROM parameter values to default
05h	0Dh	Set MC (Motion Control) ROM parameter values to default
05h	0Eh	Set Interface ROM parameter values to default
05h	0Fh	Set Application ROM parameter values to default

Meaning of the bits in the response to the telegram “Get parameter address usage”
Address Usage:

			calculationNot used for Hash				Life Parameter					ROM Write	ROM Read	RAM Write	RAM Read
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Meaning of the code in the response to the telegram “Get parameter type”
Supported Parameter Types:

Type Code	Bit Length	Description
00h	1	BOOL
01h	8	UINT8
02h	8	SINT8
03h	16	UINT16
04h	16	SINT16
05h	32	UINT32
06h	32	SINT32
07h	32	LIN_FLOAT (not used as parameter)
08h	#Char x 8	STRING
09h	#Char x 8	CAP_DIR
0Ah	1	RADIO_DIR_BIT
0Bh	16	RADIO_DIR16
0Ch	8	ENUM_DIR8
0Dh	16	ENUM_DIR16
0Eh	32	STRINGLET (Part of String 4 Characters)
0Fh	32	CAP_DIRLET (Part of String 4 Characters)

10.1 Parameter Configuration Read ROM value example

With command the RAM value of the UPID 13A2h (P Gain Position Controller, UPID 6198h for B1100) is read. With the default MC response the requested value is added in the last 4 bytes of it.

Request: Read ROM value of UPID

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	05h	Telegram length
3	02h	Fix ID start data
4	00h	Message Sub ID (Cfg Read ROM Value)
5	05h	Message Main ID (Parameter Configuration)
6	A2h	UPID Low Byte (P Gain Position Controller)
7	13h	UPID High Byte (P Gain Position Controller)
8	04h	Fix ID telegram end

Configuration Response: Read ROM value of UPID

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	0Ah	Telegram length
3	02h	Fix ID start data
4	50h	Message Sub ID (Cfg Read ROM Value Response)
5	00h	Message Main ID (Response)
6	00h	Communication state
7	A2h	UPID Low Byte (P Gain Position Controller)
8	13h	UPID High Byte (P Gain Position Controller)
9	09h	Parameter value low word low byte
10	00h	Parameter value low word high byte
11	00h	Parameter value high word low byte
12	00h	Parameter value high word high byte
13	04h	Fix ID telegram end

Examples:
Stopping MC SW:
Tx: 01 11 03 02 03 06 04
Rx: 01 11 0C 02 00 00 00 00 00 00 00 87 00 00 00 04 ; MC default response
Reading the P Gain Position Controller Set A (UPID: 13A2h) ROM value:
Tx: 01 11 05 02 00 05 A2 13 04
Rx: 01 11 0A 02 50 00 00 A2 13 **09 00 00 00** 04
Writing 16=00000010h the P Gain Position Controller Set A (UPID: 13A2h) ROM value:
Tx: 01 11 09 02 01 05 A2 13 10 00 00 00 04
Rx: 01 11 0A 02 51 00 00 A2 13 10 00 00 00 04
Reading again the P Gain Position Controller Set A (UPID: 13A2h) ROM value:
Tx: 01 11 05 02 00 05 A2 13 04
Rx: 01 11 0A 02 50 00 00 A2 13 **10 00 00 00** 04

Reading the Parameter address acces of P Gain Position Controller Set A (UPID: 13A2h):
Tx: 01 11 05 02 03 05 A2 13 04
Rx: 01 11 0A 02 53 00 00 A2 13 **0F 01** 00 00 04 ; RAM/ROM Read/Write and life
Reading the Parameter Type of P Gain Position Controller Set A (UPID: 13A2h):
Tx: 01 11 05 02 04 05 A2 13 04
Rx: 01 11 0A 02 54 00 00 A2 13 **03 00** 00 00 04 ; Par Type UINT16
Reading Min Value the P Gain Position Controller Set A (UPID: 13A2h):
Tx: 01 11 05 02 05 05 A2 13 04
Rx: 01 11 0A 02 55 00 00 A2 13 **00 00 00 00** 04
Reading Max Value the P Gain Position Controller Set A (UPID: 13A2h):
Tx: 01 11 05 02 06 05 A2 13 04
Rx: 01 11 0A 02 56 00 00 A2 13 **FF FF 00 00** 04
Reading Default Value the P Gain Position Controller Set A (UPID: 13A2h):
Tx: 01 11 05 02 07 05 A2 13 04
Rx: 01 11 0A 02 57 00 00 A2 13 **0F 00 00 00** 04

10.2 Parameter Configuration read out changed Parameters

This feature is only available on E1100 controller types: With the commands '*Start Get Modified UPID List*' and '*Get Next Modified UPID'* for each SW layer the changed parameters of the actual configuration could be read out the servo controller. With this functionality the whole parameter configuration of the servo controller could be read out and stored in the PC/ PLC.

Each firmware layer has its own range of UPIDs for its parameters.

Layer	UPID Range	Layer name
1	0000h…0EFFh	Operating System
2	1000h…1EFFh	Motion Control SW
3	2000h…2EFFh	Interface Software
4	3000h…3EFFh	Application

In the following example the changed parameters of the Intf SW (LinRS), in the example the 4 listed UPID's are changed:

UPID: 200Eh, Baud Rate Source Select, Value: 00000002h, By Parameter
UPID: 2012h, Baud Rate Parameter Def, Value: 00000008h, 38400 Baud
UPID: 206Ch, MACID Source Select, Value: 00000003h, By Parameter
UPID: 2076h, MACID Parameter Value, Value: 00000011h, MACID

Tx: 01 11 05 02 08 05 **00 20** 04 ; init read out changed Intf Par (LinRS)
Rx: 01 11 0A 02 58 00 00 00 20 00 00 00 00 04
Tx: 01 11 05 02 09 05 00 20 04 ; get next changed Intf parameter
Rx: 01 11 0A 02 59 00 00 0E 20 02 00 00 00 04 ; UPID: 200Eh, Value; 00000002h
Tx: 01 11 05 02 09 05 00 20 04 ; get next changed Intf parameter
Rx: 01 11 0A 02 59 00 00 12 20 08 00 00 00 04 ; UPID: 2012h, Value; 00000008h
Tx: 01 11 05 02 09 05 00 20 04 ; get next changed Intf parameter
Rx: 01 11 0A 02 59 00 00 6C 20 03 00 00 00 04 ; UPID: 206Ch, Value; 00000003h
Tx: 01 11 05 02 09 05 00 20 04 ; get next changed Intf parameter
Rx: 01 11 0A 02 59 00 00 76 20 11 00 00 00 04 ; UPID: 2076h, Value; 00000011h
Tx: 01 11 05 02 09 05 00 20 04 ; get next changed Intf parameter

Rx: 01 11 0A 02 59 00 **C6** C7 20 01 00 00 00 04 ; UPID: 20C7h, Value; 00000001h The Communication state C6h indicates, that this was the last parameter

To read out the changed parameters of MC-SW layer start as follows
Tx: 01 11 05 02 08 05 **00 10** 04 ; init read out changed MC-SW Par
Rx: 01 11 0A 02 58 00 00 00 10 00 00 00 00 04
Tx: 01 11 05 02 09 05 00 00 04 ; get next changed parameter
Rx: 01 11 0A 02 59 00 00 37 10 07 00 00 00 04 ; UPID: 1037h, Value; 00000007h

10.3 Parameter Configuration Read out UPID List

This feature is only available on E1100 controller types: With the commands '*Start Get UPID List*' and '*Get Next UPID'* for each SW layer all parameters of the actual configuration can be read out from the servo controller. With this functionality a parameter list of the servo controller can be read out and stored in the PC/PLC. In a second step with the request read ROM value all the values of the list can be read out.

Request: Get Next UPID

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	05h	Telegram length
3	02h	Fix ID start data
4	0Bh	Message Sub ID (Get Next UPID)
5	05h	Message Main ID (Parameter Configuration)
6	00h	Not used, don't have to be transmitted
7	00h	Not used, don't have to be transmitted
8	04h	Fix ID telegram end

Configuration Response: Get Next UPID

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	0Ah	Telegram length
3	02h	Fix ID start data
4	5Bh	Message Sub ID (Cfg Read ROM Value)
5	00h	Message Main ID (Response)
6	00h	Communication state
7	A2h	Found UPID Low Byte
8	13h	Found UPID High Byte
9	09h	Address Usage low byte of found UPID
10	00h	Address Usage high byte of found UPID
11	00h	Parameter Type low byte of found UPID
12	00h	Parameter Type high byte of found UPID
13	04h	Fix ID telegram end

The following example shows the principle of reading the UPID List of a SW instance, if generating a configuration out of this list all UPIDs with the 'ROM write' address usage bit set have to be read out with the get ROM value command.

Tx: 01 11 05 02 0A 05 00 20 04 ; start Get UPID List Intf SW layer
Rx: 01 11 0A 02 5A 00 00 00 20 00 00 00 00 04
Tx: 01 11 05 02 0B 05 00 20 04
Rx: 01 11 0A 02 5B 00 00 08 20 0D 00 0A 00 04 ; UPID 2008h, AU: 000Dh Type: 000Ah
…
Tx: 01 11 05 02 0B 05 00 20 04
Rx: 01 11 0A 02 5B 00 00 36 21 01 10 03 00 04 ; UPID 2136h, AU: 1001h Type: 0003h
Tx: 01 11 05 02 0B 05 00 20 04
Rx: 01 11 0A 02 5B 00 00 37 21 01 10 03 00 04 ; UPID 2137h, AU: 1001h Type: 0003h
Tx: 01 11 05 02 0B 05 00 20 04
Rx: 01 11 0A 02 5B 00 **C6** 37 21 01 10 03 00 04 ; UPID 2137h, AU: 1001h Type: 0003h

10.4 Parameter Configuration Defaulting SW-Instance Parameters

This feature is only for E1100 controllers: Before writing the parameters of a SW instance it is advised to set all parameters of the corresponding SW instance to default values. This can be done with a single parameter configuration message. The response is given after the defaulting of the SW instance is completed could be more than 1s.

Examples:
Defaulting the parameters of the OS-SW:
Tx: 01 11 05 02 0C 05 00 00 04
Rx: 01 11 0A 02 5C 00 00 00 00 01 00 00 00 04

Defaulting the parameters of the MC-SW:
Tx: 01 11 05 02 0D 05 00 00 04
Rx: 01 11 0A 02 5D 00 00 00 00 02 00 00 00 04

Defaulting the parameters of the Interface SW:
Tx: 01 11 05 02 0E 05 00 00 04
Rx: 01 11 0A 02 5E 00 00 00 00 03 00 00 00 04

Defaulting the parameters of the Application SW:
Tx: 01 11 05 02 0F 05 00 00 04
Rx: 01 11 0A 02 5F 00 00 00 00 04 00 00 00 04

11 Curve Configuration Message Group

Only for E1100 controllers: With the Curve Message Group, curves can be read out or written from/to the servo controller. To store a new curves in the ROM the MC SW layer has to be stopped.

Message Main ID	Message Sub ID	Description
04h	00h	Save Curves from RAM to FLASH (MC SW has to be stopped)
04h	01h	Delete all Curves in RAM
04h	02h	Delete Curve in RAM
04h	04h	Add Curve to RAM (Define Info Block Size and Data Block Size)
04h	05h	Write Curve Info Block
04h	06h	Write Curve Data Block
04h	08h	Read Curve Info Block Size and Data Block Size
04h	09h	Read Curve Info Block
04h	0Ah	Read Curve Data Block

Request: Save Curves from RAM to FLASH

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	03h	Telegram length
3	02h	Fix ID start data
4	00h	Message Sub ID (Save curves from RAM to FLASH)
5	04h	Message Main ID (Curve Message)
6	04h	Fix ID telegram end

Configuration Response: Save Curves from RAM to FLASH

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	0Ah	Telegram length
3	02h	Fix ID start data
4	40h	Message Sub ID (Curve Cfg Response)
5	00h	Message Main ID (Response)
6	00h	Communication state
7	00h	No meaning (low byte curve ID)
8	00h	No meaning (high byte curve ID)
9	09h	No meaning (response data low word low byte)
10	00h	No meaning (response data low word high byte)
11	00h	No meaning (response data high word low byte)
12	00h	No meaning (response data high word high byte)
13	04h	Fix ID telegram end

Example:

Tx: 01 11 03 02 00 04 04 ; timeout 10s
Rx: 01 11 0A 02 40 00 00 00 00 00 00 00 00 04

Request: Delete all Curves in RAM

With this command all curves defined are deleted in RAM.

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	03h	Telegram length
3	02h	Fix ID start data
4	01h	Message Sub ID (Delete all Curves in RAM)
5	04h	Message Main ID (Curve Message)
6	04h	Fix ID telegram end

Configuration Response: Delete all Curves in RAM

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	0Ah	Telegram length
3	02h	Fix ID start data
4	40h	Message Sub ID (Curve Cfg Response)
5	00h	Message Main ID (Response)
6	00h	Communication state
7	00h	No meaning (low byte curve ID)
8	00h	No meaning (high byte curve ID)
9	09h	No meaning (response data low word low byte)
10	00h	No meaning (response data low word high byte)
11	00h	No meaning (response data high word low byte)
12	00h	No meaning (response data high word high byte)
13	04h	Fix ID telegram end

Example:

Tx: 01 11 03 02 01 04 04
Rx: 01 11 0A 02 40 00 00 00 00 00 00 00 00 04

Request: Delete Curve in RAM

With this command the curve with ID 1 defined is deleted in RAM.

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	03h	Telegram length
3	02h	Fix ID start data
4	02h	Message Sub ID (Delete Curve in RAM)
5	04h	Message Main ID (Curve Message)
6	01h	Curve ID low byte
7	00h	Curve ID high byte
8	04h	Fix ID telegram end

Configuration Response: Delete Curve in RAM

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	0Ah	Telegram length
3	02h	Fix ID start data
4	40h	Message Sub ID (Curve Cfg Response)
5	00h	Message Main ID (Response)
6	00h	Communication state
7	01h	low byte curve ID
8	00h	high byte curve ID
9	09h	No meaning (response data low word low byte)
10	00h	No meaning (response data low word high byte)
11	00h	No meaning (response data high word low byte)
12	00h	No meaning (response data high word high byte)
13	04h	Fix ID telegram end

Example:

Tx: 01 11 05 02 02 04 01 00 04

Rx: 01 11 0A 02 40 00 00 01 00 00 00 00 00 04

11.1 Read Curve From Servo Example

Only for E1100 controllers: For a detailed description about the saving structure of a curve refer to [2]. In the following example the curve with ID = 1 is read from the servo controller

Reading Curve 1 Info Block and Data Block size:
Tx: 01 11 05 02 08 04 01 00 04
Rx: 01 11 0A 02 40 00 00 01 00 46 00 54 00 04 ; info block: 46bytes, data block: 54bytes

Reading Curve 1 Info Block Data:
Tx: 01 11 05 02 09 04 01 00 04
Rx: 01 11 0A 02 40 00 **04** 01 00 46 00 03 00 04
Tx: 01 11 05 02 09 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 15 00 04 00 04
Tx: 01 11 05 02 09 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 53 69 6E 52 04
Tx: 01 11 05 02 09 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 69 73 65 00 04
Tx: 01 11 05 02 09 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 05 02 09 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 05 02 09 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 05 02 09 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 01 00 04
Tx: 01 11 05 02 09 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 A0 86 01 00 04
Tx: 01 11 05 02 09 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 1A 00 05 00 04
Tx: 01 11 05 02 09 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 01 03 A0 86 04
Tx: 01 11 05 02 09 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 01 00 00 00 04
Tx: 01 11 05 02 09 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 40 42 04
Tx: 01 11 05 02 09 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 0F 00 00 00 04
Tx: 01 11 05 02 09 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 05 02 09 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 05 02 09 04 01 00 04
Rx: 01 11 0A 02 40 00 **04** 01 00 00 00 00 00 04
Tx: 01 11 05 02 09 04 01 00 04
Rx: 01 11 0A 02 40 00 **00** 01 00 00 00 00 00 04

Reading Curve data 21 Position values:

Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 **04** 01 00 00 00 00 00 04 ; position value 1
Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 0C 18 00 00 04 ; position value 2
Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 98 5F 00 00 04 ; position value 3
Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 E1 D4 00 00 04 ; position value 4
Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 04 75 01 00 04 ; position value 5
Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 0F 3C 02 00 04 ; position value 6
Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 1B 25 03 00 04 ; position value 7
Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 6D 2A 04 00 04 ; position value 8
Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 94 45 05 00 04 ; position value 9
Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 97 6F 06 00 04 ; position value 10
Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 20 A1 07 00 04 ; position value 11
Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 A9 D2 08 00 04 ; position value 12
Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 AC FC 09 00 04 ; position value 13
Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 D3 17 0B 00 04 ; position value 14
Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 25 1D 0C 00 04 ; position value 15
Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 31 06 0D 00 04 ; position value 16
Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 3C CD 0D 00 04 ; position value 17
Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 5F 6D 0E 00 04 ; position value 18
Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 A8 E2 0E 00 04 ; position value 19
Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 **04** 01 00 34 2A 0F 00 04 ; position value 20
Tx: 01 11 05 02 0A 04 01 00 04
Rx: 01 11 0A 02 40 00 **00** 01 00 40 42 0F 00 04 ; position value 21

11.2 Write Curve To Servo Example

Write curve 1 info block size and data block size:
Tx: 01 11 09 02 04 04 01 00 46 00 54 00 04
Rx: 01 11 0A 02 40 00 00 01 00 00 00 00 00 04

Write curve 1 info block data:
Tx: 01 11 09 02 05 04 01 00 46 00 03 00 04
Rx: 01 11 0A 02 40 00 **04** 01 00 00 00 00 00 04
Tx: 01 11 09 02 05 04 01 00 15 00 04 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 05 04 01 00 53 69 6E 52 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 05 04 01 00 69 73 65 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 05 04 01 00 00 00 00 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 05 04 01 00 00 00 00 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 05 04 01 00 00 00 00 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 05 04 01 00 00 00 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 05 04 01 00 A0 86 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 05 04 01 00 1A 00 05 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 05 04 01 00 01 03 A0 86 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 05 04 01 00 01 00 00 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 05 04 01 00 00 00 40 42 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 05 04 01 00 0F 00 00 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 05 04 01 00 00 00 00 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 05 04 01 00 00 00 00 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 05 04 01 00 00 00 00 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 05 04 01 00 00 00 00 00 04
Rx: 01 11 0A 02 40 00 **00** 01 00 00 00 00 00 04

Write curve 1 data:
Tx: 01 11 09 02 06 04 01 00 00 00 00 00 04
Rx: 01 11 0A 02 40 00 **04** 01 00 00 00 00 00 04
Tx: 01 11 09 02 06 04 01 00 0C 18 00 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 06 04 01 00 98 5F 00 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 06 04 01 00 E1 D4 00 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 06 04 01 00 04 75 01 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 06 04 01 00 0F 3C 02 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 06 04 01 00 1B 25 03 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 06 04 01 00 6D 2A 04 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 06 04 01 00 94 45 05 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 06 04 01 00 97 6F 06 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 06 04 01 00 20 A1 07 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 06 04 01 00 A9 D2 08 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 06 04 01 00 AC FC 09 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 06 04 01 00 D3 17 0B 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 06 04 01 00 25 1D 0C 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 06 04 01 00 31 06 0D 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 06 04 01 00 3C CD 0D 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 06 04 01 00 5F 6D 0E 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 06 04 01 00 A8 E2 0E 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 06 04 01 00 34 2A 0F 00 04
Rx: 01 11 0A 02 40 00 04 01 00 00 00 00 00 04
Tx: 01 11 09 02 06 04 01 00 40 42 0F 00 04
Rx: 01 11 0A 02 40 00 **00** 01 00 00 00 00 00 04

12 Command Table Message Group

Only for E1100 controllers: With the Command Table Message Group, Command table Entries can be read out or written from/to the servo controller. To store a new command table in the ROM the MC SW layer has to be stopped.

Message Main ID	Message Sub ID	Description
08h	00h	Save Command Table from RAM to FLASH (MC SW has to be stopped)
08h	01h	Delete all Command Table Entries in RAM
08h	02h	Delete Command table entry
08h	03h	Setup Write Command Table entry in RAM
08h	04h	Write Command Table entry data in RAM
08h	05h	Setup Read Command Table Entry
08h	06h	Read Command Table entry data
08h	07h	Get Command Table defined entry list

Request: Save Command Table from RAM to FLASH

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	03h	Telegram length
3	02h	Fix ID start data
4	00h	Message Sub ID (Save CT from RAM to FLASH)
5	08h	Message Main ID (CT Message)
6	04h	Fix ID telegram end

Configuration Response: Save Command Table from RAM to FLASH

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	0Ah	Telegram length
3	02h	Fix ID start data
4	80h	Message Sub ID (CT Flashing completed Response)
5	00h	Message Main ID (Response)
6	00h	Communication state
7	00h	No meaning (low CT entry ID)
8	00h	No meaning (high CT entry ID)
9	00h	No meaning (response data low word low byte)
10	00h	No meaning (response data low word high byte)
11	00h	No meaning (response data high word low byte)
12	00h	No meaning (response data high word high byte)
13	04h	Fix ID telegram end

Example:

Tx: 01 11 03 02 00 08 04 ; timeout 10s
Rx: 01 11 0A 02 80 00 00 00 00 00 00 00 00 04

Request: Delete all Command Table Entries in RAM

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	03h	Telegram length
3	02h	Fix ID start data
4	01h	Message Sub ID (Delete all CT Entries in RAM)
5	08h	Message Main ID (CT Message)
6	04h	Fix ID telegram end

Configuration Response: Delete all Command Table Entries in RAM

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	0Ah	Telegram length
3	02h	Fix ID start data
4	81h	Message Sub ID (Delete all CT Entries in RAM)
5	00h	Message Main ID (Response)
6	00h	Communication state
7	00h	No meaning (low byte CT entry ID)
8	00h	No meaning (high byte CT entry ID)
9	00h	No meaning (response data low word low byte)
10	00h	No meaning (response data low word high byte)
11	00h	No meaning (response data high word low byte)
12	00h	No meaning (response data high word high byte)
13	04h	Fix ID telegram end

Example:
Tx: 01 11 03 02 01 08 04
Rx: 01 11 0A 02 81 00 00 00 00 00 00 00 00 04

Request: Delete Command Table Entry in RAM

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	03h	Telegram length
3	02h	Fix ID start data
4	02h	Message Sub ID (Delete CT Entry in RAM)
5	08h	Message Main ID (CT Message)
6	02h	Low byte CT entry ID
7	00h	High byte CT entry ID
8	04h	Fix ID telegram end

Configuration Response: Delete Command Table Entry in RAM

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	0Ah	Telegram length
3	02h	Fix ID start data
4	82h	Message Sub ID (CT Flashing completed Response)
5	00h	Message Main ID (Response)
6	00h	Communication state
7	02h	Low byte CT entry ID
8	00h	High byte CT entry ID
9	00h	No meaning (response data low word low byte)
10	00h	No meaning (response data low word high byte)
11	00h	No meaning (response data high word low byte)
12	00h	No meaning (response data high word high byte)
13	04h	Fix ID telegram end

Example:
Tx: 01 11 05 02 02 08 02 00 04
Rx: 01 11 0A 02 82 00 00 02 00 00 00 00 00 04

Request: Get Command Table Defined Entry List

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	05h	Telegram length
3	02h	Fix ID start data
4	07h	Message Sub ID (Delete CT Entry in RAM)
5	08h	Message Main ID (CT Message)
6	00h	Low byte CT entry ID 0..7
7	00h	High byte CT entry ID
8	04h	Fix ID telegram end

Configuration Response: Get Command Table Defined Entry List

Byte Offset	Value	Description
0	01h	Fix ID Telegram start
1	11h	Destination node ID
2	0Ah	Telegram length
3	02h	Fix ID start data
4	82h	Message Sub ID (CT Flashing completed Response)
5	00h	Message Main ID (Response)
6	00h	Communication state
7	00h	Low byte CT entry List ID
8	00h	High byte CT entry ID
9	89h	Entry List 0..7 bit = 0 entry exists, bit = 1 entry not defined
10	00h	Entry List 8..15 bit = 0 entry exists, bit = 1 entry not defined
11	00h	Entry List 16..23 bit = 0 entry exists, bit = 1 entry not defined
12	00h	Entry List 24..31 bit = 0 entry exists, bit = 1 entry not defined
13	04h	Fix ID telegram end

Example:

Tx: 01 11 09 02 07 08 00 00 00 00 00 00 04
Rx: 01 11 0A 02 87 00 00 00 00 89 FF FF FF 04 ; CT entry list 0..31; 1,2,4,5,6 defined
Tx: 01 11 09 02 07 08 01 00 00 00 00 00 04
Rx: 01 11 0A 02 87 00 00 01 00 FF FF FF FF 04 ; CT entry list 32..63; no entry defined
Tx: 01 11 09 02 07 08 02 00 00 00 00 00 04
Rx: 01 11 0A 02 87 00 00 02 00 FF FF FF FF 04 ; CT entry list 64..95; no entry defined
Tx: 01 11 09 02 07 08 03 00 00 00 00 00 04
Rx: 01 11 0A 02 87 00 00 03 00 FF FF FF FF 04 ; CT entry list 96..127; no entry defined
Tx: 01 11 09 02 07 08 04 00 00 00 00 00 04
Rx: 01 11 0A 02 87 00 00 04 00 FF FF FF FF 04 ; CT entry list 128..159; no entry defined
Tx: 01 11 09 02 07 08 05 00 00 00 00 00 04
Rx: 01 11 0A 02 87 00 00 05 00 FF FF FF FF 04 ; CT entry list 160..191; no entry defined
Tx: 01 11 09 02 07 08 06 00 00 00 00 00 04
Rx: 01 11 0A 02 87 00 00 06 00 FF FF FF FF 04 ; CT entry list 192..223; no entry defined
Tx: 01 11 09 02 07 08 07 00 00 00 00 00 04
Rx: 01 11 0A 02 87 00 00 07 00 FF FF FF FF 04 ; CT entry list 224..255; no entry defined

12.1 Read Command Table Entry From Servo Example

Tx: 01 11 05 02 05 08 02 00 04
Rx: 01 11 0A 02 85 00 00 02 00 **40** 00 00 00 04

Tx: 01 11 05 02 06 08 02 00 04
Rx: 01 11 0A 02 86 00 **04** 02 00 01 A7 FF FF 04
Tx: 01 11 05 02 06 08 02 00 04
Rx: 01 11 0A 02 86 00 04 02 00 00 01 40 42 04
Tx: 01 11 05 02 06 08 02 00 04
Rx: 01 11 0A 02 86 00 04 02 00 0F 00 20 A1 04
Tx: 01 11 05 02 06 08 02 00 04
Rx: 01 11 0A 02 86 00 04 02 00 07 00 40 0D 04
Tx: 01 11 05 02 06 08 02 00 04
Rx: 01 11 0A 02 86 00 04 02 00 03 00 40 0D 04
Tx: 01 11 05 02 06 08 02 00 04
Rx: 01 11 0A 02 86 00 04 02 00 03 00 00 00 04
Tx: 01 11 05 02 06 08 02 00 04
Rx: 01 11 0A 02 86 00 04 02 00 00 00 00 00 04
Tx: 01 11 05 02 06 08 02 00 04
Rx: 01 11 0A 02 86 00 04 02 00 00 00 00 00 04
Tx: 01 11 05 02 06 08 02 00 04
Rx: 01 11 0A 02 86 00 04 02 00 00 00 00 00 04
Tx: 01 11 05 02 06 08 02 00 04
Rx: 01 11 0A 02 86 00 04 02 00 00 00 55 6E 04
Tx: 01 11 05 02 06 08 02 00 04
Rx: 01 11 0A 02 86 00 04 02 00 6E 61 6D 65 04
Tx: 01 11 05 02 06 08 02 00 04
Rx: 01 11 0A 02 86 00 04 02 00 64 00 00 00 04
Tx: 01 11 05 02 06 08 02 00 04
Rx: 01 11 0A 02 86 00 04 02 00 00 00 00 00 04
Tx: 01 11 05 02 06 08 02 00 04
Rx: 01 11 0A 02 86 00 04 02 00 00 00 FF FF 04
Tx: 01 11 05 02 06 08 02 00 04
Rx: 01 11 0A 02 86 00 04 02 00 FF FF FF FF 04
Tx: 01 11 05 02 06 08 02 00 04
Rx: 01 11 0A 02 86 00 **00** 02 00 FF FF FF FF 04

12.2 Write Command Table Entry To Servo Example

Only for E1100 controllers: Setup Write Command Table entry 2 data size 40hbytes:
Tx: 01 11 09 02 03 08 02 00 40 00 00 00 04
Rx: 01 11 0A 02 83 00 00 02 00 00 00 00 00 04

Write Command Table entry 2 data :
Tx: 01 11 09 02 04 08 02 00 01 A7 FF FF 04
Rx: 01 11 0A 02 84 00 04 02 00 00 00 00 00 04
Tx: 01 11 09 02 04 08 02 00 00 01 40 42 04
Rx: 01 11 0A 02 84 00 04 02 00 00 00 00 00 04
Tx: 01 11 09 02 04 08 02 00 0F 00 20 A1 04
Rx: 01 11 0A 02 84 00 04 02 00 00 00 00 00 04
Tx: 01 11 09 02 04 08 02 00 07 00 40 0D 04
Rx: 01 11 0A 02 84 00 04 02 00 00 00 00 00 04
Tx: 01 11 09 02 04 08 02 00 03 00 40 0D 04
Rx: 01 11 0A 02 84 00 04 02 00 00 00 00 00 04
Tx: 01 11 09 02 04 08 02 00 03 00 00 00 04
Rx: 01 11 0A 02 84 00 04 02 00 00 00 00 00 04
Tx: 01 11 09 02 04 08 02 00 00 00 00 00 04
Rx: 01 11 0A 02 84 00 04 02 00 00 00 00 00 04
Tx: 01 11 09 02 04 08 02 00 00 00 00 00 04
Rx: 01 11 0A 02 84 00 04 02 00 00 00 00 00 04
Tx: 01 11 09 02 04 08 02 00 00 00 00 00 04
Rx: 01 11 0A 02 84 00 04 02 00 00 00 00 00 04
Tx: 01 11 09 02 04 08 02 00 00 00 55 6E 04
Rx: 01 11 0A 02 84 00 04 02 00 00 00 00 00 04
Tx: 01 11 09 02 04 08 02 00 6E 61 6D 65 04
Rx: 01 11 0A 02 84 00 04 02 00 00 00 00 00 04
Tx: 01 11 09 02 04 08 02 00 64 00 00 00 04
Rx: 01 11 0A 02 84 00 04 02 00 00 00 00 00 04
Tx: 01 11 09 02 04 08 02 00 00 00 00 00 04
Rx: 01 11 0A 02 84 00 04 02 00 00 00 00 00 04
Tx: 01 11 09 02 04 08 02 00 00 00 FF FF 04
Rx: 01 11 0A 02 84 00 04 02 00 00 00 00 00 04
Tx: 01 11 09 02 04 08 02 00 FF FF FF FF 04
Rx: 01 11 0A 02 84 00 04 02 00 00 00 00 00 04
Tx: 01 11 09 02 04 08 02 00 FF FF FF FF 04
Rx: 01 11 0A 02 84 00 00 02 00 00 00 00 00 04

13 Program Handling Message Group

With the program handling message group, the whole servo controller or/and SW instances of it can be accessed.

Message Main ID	Message Sub ID	B1100	Description
06h	01h	Yes	Reset Servo Controller Completely (restart of all SW instances) with response after reset completion
06h	02h	Yes	Reset Servo Controller with immediate response
06h	03h	-	Stop MC- and Application SW
06h	04h	-	Start MC- and Application SW with response after start completion
06h	05h	-	Start MC- and Application SW with immediate response

13.1 Reset Servo Controller with Response after completion

Request: Reset Servo Controller

Byte Offset	Value	Description
0	01h	Fix ID telegram start
1	11h	MACID
2	03h	Data length
3	02h	Fix ID start data
4	01h	Sub ID: Reset Servo Controller with Response after completion
5	06h	Main ID: Program Handling Message Group
6	04h	Fix ID telegram end

Response: Reset Servo Controller

Byte Offset	Value	Description
0	01h	Fix ID telegram start
1	11h	MACID
2	04h	Data length
3	02h	Fix ID start data
4	60h	Sub ID: Program Handling Response
5	00h	Main ID: Response Message
6	00h	Communication State ok
7	04h	Fix ID telegram end

Example:
Tx: 01 11 03 02 01 06 04
Rx: 01 11 04 02 60 00 00 04
The response is given after the reset is completed (ca. 3s)

13.2 Reset Servo Controller with immediate Response

Request: Reset Servo Controller with immediate Response

Byte Offset	Value	Description
0	01h	Fix ID telegram start
1	11h	MACID
2	03h	Data length
3	02h	Fix ID start data
4	01h	Sub ID: Reset Servo Controller with immediate Response
5	06h	Main ID: Program Handling Message Group
6	04h	Fix ID telegram end

Response: Reset Servo Controller with immediate Response

The servo controller answers with the configured default response.

Example:
Tx: 01 11 03 02 02 06 04
Rx: 01 11 0C 02 00 00 00 F6 40 00 00 8A 00 00 00 04
The default response is given immediate, to detect reboot completion poll servo controller until it answers.

13.3 Stop MC- and Application SW

Request: Stop MC- and Application SW

Byte Offset	Value	Description
0	01h	Fix ID telegram start
1	11h	MACID
2	03h	Data length
3	02h	Fix ID start data
4	03h	Sub ID: Stop MC- and Application SW
5	06h	Main ID: Program Handling Message Group
6	04h	Fix ID telegram end

Response: Reset Servo Controller with immediate Response

The servo controller answers with the configured default response.

Example:
Tx: 01 11 03 02 03 06 04
Rx: 01 11 0C 02 00 00 00 00 00 00 00 D9 00 00 00 04

13.4 Start MC- and Application SW with Response after completion

Request: Start MC- and Application SW with Response after completion

Byte Offset	Value	Description
0	01h	Fix ID telegram start
1	11h	MACID
2	03h	Data length
3	02h	Fix ID start data
4	01h	Sub ID: Reset Servo Controller with Response after completion
5	06h	Main ID: Program Handling Message Group
6	04h	Fix ID telegram end

Response: Start MC- and Application SW with Response after completion

Byte Offset	Value	Description
0	01h	Fix ID telegram start
1	11h	MACID
2	04h	Data length
3	02h	Fix ID start data
4	60h	Sub ID: Program Handling Response
5	00h	Main ID: Response Message
6	00h	Communication State ok
7	04h	Fix ID telegram end

Example:
Tx: 01 11 03 02 04 06 04
Rx: 01 11 04 02 60 00 00 04
The response is given after the start is completed (ca. 3s)

13.5 Start MC- and Application SW with immediate Response

Request: Start MC- and Application SW with immediate Response

Byte Offset	Value	Description
0	01h	Fix ID telegram start
1	11h	MACID
2	03h	Data length
3	02h	Fix ID start data
4	01h	Sub ID: Reset Servo Controller with immediate Response
5	06h	Main ID: Program Handling Message Group
6	04h	Fix ID telegram end

Response: Start MC- and Application SW with immediate Response

The servo controller answers with the configured default response.

Example:
Tx: 01 11 03 02 05 06 04
Rx: 01 11 0C 02 00 00 00 F6 40 00 00 8A 00 00 00 04
The default response is given immediate, to detect restart completion poll servo controller until it answers.

14 Read Error Info Message Group

With the Read Error Info message group, error strings and the stored error log of the controller could be read out.

Message Main ID	Message Sub ID	Description
07h	00h	Get error short text of actual error
07h	01h	Get error short text of defined error code
07h	02h	Get error counters of error log and total occurred error
07h	03h	Get error log entry (error code, short text and time)

14.1 Get error short text of actual error

Only for E1100 controllers: With the get error short text of actual error request the slave answers with string with 32 characters, which contains the short text of the actual error code (unused characters are filled with 00h).

Request: Get error short text of actual error

Byte Offset	Value	Description
0	01h	Fix ID telegram start
1	11h	MACID
2	03h	Data length
3	02h	Fix ID start data
4	00h	Sub ID: Get error short text of actual error
5	07h	Main ID: Read Error Info Message Group
6	04h	Fix ID telegram end

Response: Get error short text of actual error

Byte Offset	Value	Description
0	01h	Fix ID telegram start
1	11h	MACID
2	24h	Data length
3	02h	Fix ID start data
4	70h	Sub ID: Response Get error short text of actual error
5	00h	Main ID: Response Message
6	00h	Communication State ok
7	4Eh	First character 'N'
.	..	Characters 2..32
39	04h	Fix ID telegram end

Example:
Tx: 01 11 03 02 00 07 04
Rx: 01 11 24 02 70 00 00 4E 6F 20 45 72 72 6F 72 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 04
The slave responds with the string ‚No Error'.

14.2 Get error short text of defined error code

Only for E1100 controllers: With the get error short text of defined error code request the slave answers with string with 32 characters, which contains the short text of the actual error code (unused characters are filled with 00h).

Request: Get error short text of defined error code

Byte Offset	Value	Description
0	01h	Fix ID telegram start
1	11h	MACID
2	05h	Data length
3	02h	Fix ID start data
4	01h	Sub ID: Get error short text of defined error code
5	07h	Main ID: Read Error Info Message Group
6	01h	Error code low byte
7	00h	Error code High byte (have to be 00h)
8	04h	Fix ID telegram end

Response: Get error short text of defined error code

Byte Offset	Value	Description
0	01h	Fix ID telegram start
1	11h	MACID
2	26h	Data length
3	02h	Fix ID start data
4	71h	Sub ID: Response Get error short text of actual error
5	00h	Main ID: Response Message
6	00h	Communication State ok
7	01h	Error code low byte
8	00h	Error code high byte
9	4Eh	First character ‘N’
.	..	Characters 2..32
41	04h	Fix ID telegram end

Example:
Read Error text of Error Code 01h:
Tx: 01 11 05 02 01 07 01 00 04
Rx: 01 11 26 02 71 00 00 01 00 45 72 72 3A 20 58 34 20 4C 6F 67 69 63 20 53 75 70 70 6C 79 20 54 6F 6F 20 4C 6F 77 00 00 00 00 04
The slave responds with the string ‚Err: X4 Logic Supply Too Low’.

14.3 Get error counters of error log and total occurred errors

With the Get error short text of defined error code request the slave answers with string with 32 characters, which contains the short text of the actual error code (unused characters are filled with 00h).

Request: Get error counters of error log and total occurred erros

Byte Offset	Value	Description
0	01h	Fix ID telegram start
1	11h	MACID
2	03h	Data length
3	02h	Fix ID start data
4	02h	Sub ID: Get error counters
5	07h	Main ID: Read Error Info Message Group
6	04h	Fix ID telegram end

Response: Get error short text of defined error code

Byte Offset	Value	Description
0	01h	Fix ID telegram start
1	11h	MACID
2	08h	Data length
3	02h	Fix ID start data
4	72h	Sub ID: Response Get error short text of actual error
5	00h	Main ID: Response Message
6	00h	Communication State ok
7	15h	Low byte Number of error log entries
8	00h	High byte Number of error log entries
9	4Eh	Low byte Number of occurred errors
10	00h	High byte Number of occurred errors
41	04h	Fix ID telegram end

Example:
Tx: 01 11 03 02 02 07 04
Rx: 01 11 08 02 72 00 00 15 00 90 00 04
The slave responds 21 (15h) Logged errors (=buffer size) and 144 (90h) occurred errors.

14.4 Get error log entry

With the Get error short text of defined error code request the slave answers with string with 32 characters, which contains the short text of the actual error code (unused characters are filled with 00h).

Request: Get error counters of error log and total occurred erros

Byte Offset	Value	Description
0	01h	Fix ID telegram start
1	11h	MACID
2	05h	Data length
3	02h	Fix ID start data
4	03h	Sub ID: Get error log entry
5	07h	Main ID: Read Error Info Message Group
6	00h	Low byte error log entry number (0 = newest)
7	00h	How byte error log entry number
8	04h	Fix ID telegram end

Response: Get error short text of defined error code

Byte Offset	Value	Description
0	01h	Fix ID telegram start
1	11h	MACID
2	0Eh	Data length
3	02h	Fix ID start data
4	73h	Sub ID: Response Get error log entry
5	00h	Main ID: Response Message
6	00h	Communication State ok
7	0Ch	Low byte error code
8	00h	High byte error code
9	3Eh	Low byte low word milli second counter (run time)
10	11h	High byte low word milli second counter (run time)
11	0Eh	Low byte high word milli second counter (run time)
12	00h	High byte high word milli second counter (run time)
13	6Fh	Low byte low word hour counter (run time)
14	02h	High byte low word hour counter (run time)
15	00h	Low byte high word hour counter (run time)
16	00h	High byte high word hour counter (run time)
17	04h	Fix ID telegram end

Example:
Tx: 01 11 05 02 03 07 00 00 04
Rx: 01 11 0E 02 73 00 00 0C 00 3E 11 0E 00 6F 02 00 00 04
The slave responds:
Error code: 000Ch, ‚Err. Pos Lag Standing Too Big’
Milli second counter: 000E113Eh= 921218ms=15min 21s 918ms
Hour counter: 0000026Fh= 623h

15 LinRS Parameters

The E1100 Servo Controllers with loaded LinRS protocol SW have an additional parameter tree branch, which can be configured with the distributed LinMot-Talk software. With these parameters, the LinRS behaviour can be configured. The software LinMot-Talk can be downloaded from http://www.linmot.com under the section download, software & manuals.

Dis-/Enable With the Dis-/Enable parameter the LinMot servo controller can be run without the LinRS going online.

LinRS\ Dis-/Enable	
Disable	Servo controller runs without LinRS.
Enable	Servo controller runs with a LinRS connection, the RS configuration is not port is no longer active! (default)

IMPORTANT: To activate the LinRS Interface, the Dip-Switch S3.4 “Interface” at the bottom of the drive has to be set to “ON” with power up.

RS Config In this section the RS UART behaviour can be configured.
RS Select In this section the RS line type of RS can be configured.
RS Source Select Over the RS select parameter the bus topology is defined (E1100 only).

LinRS\ RS Config\ RS Select\ RS Source Select	
By S3.1	Look at S3.1 for RS232 RS 485 selection (default)
Parameter	Take value from parameter RS Parameter Def.

RS Parameter Def Over the RS select parameter the bus topology is defined.

LinRS\ RS Config\ RS Select\ RS Parameter Def	
RS 485	RS 485 two wire bus topology (default)
RS 422	RS 422 four wire bus topology
RS 232	RS 232 two wire point to point bus topology

Baud Rate In this section the parameters for the baud rate selection are located.
Baud Rate Source Select
Defines if the baud rate is defined over Hex Switch S1 or parameter (E1100 only).

LinRS\ RS Config\ Baud Rate\ Baud Rate Source Select	
By Hex Switch S1	Look at S1 for Baud Rate Selection (default)
By Parameter	Take value from parameter Baud Rate Parameter Definition.

Baud Rate Parameter Definition

The baud rate definition if defined with parameter.

LinRS\ RS Config\ Baud Rate\ Baud Rate Parameter Def	
4800 Bit/s	RS baud rate = 4800 Bit/s
9600 Bit/s	RS baud rate = 9600 Bit/s
19200 Bit/s	RS baud rate = 19200 Bit/s
38400 Bit/s	RS baud rate = 38400 Bit/s
57600 Bit/s	RS baud rate = 57600 Bit/s (default)
115200 Bit/s	RS baud rate = 115200 Bit/s

Stop Bit Defines the stop bit length.

LinRS\ RS Config\ Stop Bit	
1	One stop bit
2	Two bit time stop bit

Parity Defines the parity bit behaviour.

LinRS\ RS Config\ Parity	
None	No parity bit
Even	Even parity bit
Odd	Odd parity bit

Protocol Config In this section the protocol can be configured.

MACID In this section the MAC ID (controller number) can be configured.

ID Source Select

The MACID parameter defines the source of the MACID (Node Address).

E1100: LinRS\ Protocol Config\ MACID\ MACID Source Select B1100: OS\ Communication\ MACID\ MACID Source Select	
By Hex Switch S2	E1100/E1200 only:The MACID is determined by the hex switch S2 (default)
By Hex Switches S1 and S2	E1100/E1200 only: The MACID is determined by the two hex switches S1 and S2
By Parameter	The MACID is determined by parameter setting
By Dig In 1	B1100 only: The MACID is defined by DigIn1 (X13.14) at power up. 0V = ID 0, 24V = ID 1
By Dig In 2..1	B1100 only: The MACID is defined by DigIn2 .. 1 (X13.2 and X13.14) at power up. DigIn2 is the most, DigIn1 the least significant bit. (00b = ID 0, 11b = ID 3)
By Dig In 3..1	B1100 only: The MACID is defined by DigIn3 .. 1 (X13.15, X13.2 and X13.14) at power up. DigIn3 is the most, DigIn1 the least significant bit. (000b = ID 0, 111b = ID 7)
By Dig In 4..1	B1100 only: The MACID is defined by DigIn4 .. 1 (X13.3, X13.15, X13.2 and X13.14) at power up. DigIn4 is the most, DigIn1 the least significant bit. (0000b = ID 0, 1111b = ID 15)
By Dig In 5..1	B1100 only: The MACID is defined by DigIn5 .. 1 (X13.16, X13.3, X13.15, X13.2 and X13.14) at power up. DigIn5 is the most, DigIn1 the least significant bit. (00000b = ID 0, 11111b = ID 31)

By Dig In 6..1	B1100 only: The MACID is defined by DigIn6 .. 1 (X13.4, X13.16, X13.3, X13.15, X13.2 and X13.14) at power up. DigIn6 is the most, DigIn1 the least significant bit. (000000b = ID 0, 111111b = ID 63)
By Dig In 1 + Offset	B1100 only: The MACID is defined by DigIn1 (X14.14) at power up plus the value of 6081h (MACID Parameter Value) as offset. 0V = ID 0, 24V = ID 1 (plus offset).
By Dig In 2..1 + Offset	B1100 only: The MACID is defined by DigIn2 .. 1 (X14.2 and X14.14) at power up plus the value of 6081h (MACID Parameter Value) as offset. DigIn2 is the most, DigIn1 the least significant bit. (00b = ID 0, 11b = ID 3 (plus offset))
By Dig In 3..1 + Offset	B1100 only: The MACID is defined by DigIn3 .. 1 (X14.15, X14.2 and X14.14) at power up plus the value of 6081h (MACID Parameter Value) as offset. DigIn3 is the most, DigIn1 the least significant bit. (000b = ID 0, 111b = ID 7 (plus offset))
By Dig In 4..1 + Offset	B1100 only: The MACID is defined by DigIn4 .. 1 (X14.3, X14.15, X14.2 and X14.14) at power up plus the value of 6081h (MACID Parameter Value) as offset. DigIn4 is the most, DigIn1 the least significant bit. (0000b = ID 0, 1111b = ID 15 (plus offset))
By Dig In 5..1 + Offset	B1100 only: The MACID is defined by DigIn5 .. 1 (X14.16, X14.3, X14.15, X14.2 and X14.14) at power up plus the value of 6081h (MACID Parameter Value) as offset. DigIn5 is the most, DigIn1 the least significant bit. (00000b = ID 0, 11111b = ID 31 (plus offset))
By Dig In 6..1 + Offset	B1100 only: The MACID is defined by DigIn6 .. 1 (X14.4, X14.16, X14.3, X14.15, X14.2 and X14.14) at power up plus the value of 6081h (MACID Parameter Value) as offset. DigIn6 is the most, DigIn1 the least significant bit. (000000b = ID 0, 111111b = ID 63 (plus offset))
Parameter Value	The MACID, when “Parameter” is selected

MACID Parameter Value

The ID parameter defines the source of the MACID.

LinRS\ Protocol Config\ MACID\ MACID Parameter Value	
MACID Parameter Value	The ID, when “Parameter” is selected as ID Source (11h default)

Checksum The checksum parameter defines the checksum generation.

LinRS\ Protocol Config\ Checksum	
None	No checksum is expected or generated (default)
Add	A simple byte wise addition modulo 2^16 (fast and easy)
CRC CCITT	CRC checksum with CCITT polynomial 16 bit

Checksum Start Value This parameter defines the start value of thechecksum generation.

LinRS\ Protocol Config\ Checksum Start Value	
0000h	Checksum start value 0 (default)
FFFFh	Checksum start value 0xFFFF
1D0Fh (CCITT standard)	Standart CCITT start value

Receive Time Out Specifies the byte to byte time out during receiving, if the time out occurs the receive state machine is reset. This behaviour enables a correct receive of the following telegram even if the actual telegram is corrupted.

LinRS\ Protocol Config\ Receive Time Out	
Enable	• Enable • Disable
Time Out	Byte to byte time out value

MC Response Configuration

The response configuration determines the data that is responded from the LinMot servo controller to the PLC. The orders of the data correspond to selection order in the response.

LinRS\ Protocol Config\ MC Response Configuration	
Communication State	LinRS Status byte of communication (Default Selection On)
Status Word	Status Word (Default Selection On)
State Var	State Variable (Default Selection Off)
Error Code	Error Code (Default Selection Off)
Warn Word	Warn Word (Default Selection Off)
MC Cmd Intf Header Echo	MC command interface echo (Default Selection Off)
Monitoring Channel 1	Monitoring Channel 1 Selection (Default On)
Channel 1 UPID	Monitoring Channel 1 UPID
Monitoring Channel 2	Monitoring Channel 2 Selection (Default Off)
Channel 2 UPID	Monitoring Channel 2 UPID
MC Response	4 byte Place holder for MC Response (Default Off)

Error In this section the Error behaviour can be defined.

Error Detection Mask

With the error detection mask a single error can be disabled. Also a LinRS error causes the MC-SW go to the error state.

LinRS\ Error\ Error Detection Mask	
Checksum Error	(Default Selection On)
End Of Telegram Missing	(Default Selection On)
Wrong Msg Main ID	(Default Selection On)
Wrong Msg Sub ID	(Default Selection On)
UPID Not Existing	(Default Selection On)

Respond On Msg With Error

Typically the LinMot servo controller doesn't answer to wrong telegrams, with this mask the response for certain errors can be enabled. With turned on Communication state in the MC default response the error will be responded to the master.

LinRS\ Error\ Respond On Msg With Error	
Checksum Error	(Default Selection On)
End Of Telegram Missing	(Default Selection On)

16 Error

Within the LinRS Intf SW several errors are supported, most of them can be disabled, because they are not fatal. For the motion control specific errors refer to document [1].

16.1 LinRS Error Codes

In the table below the LinRS specific error codes are listed.

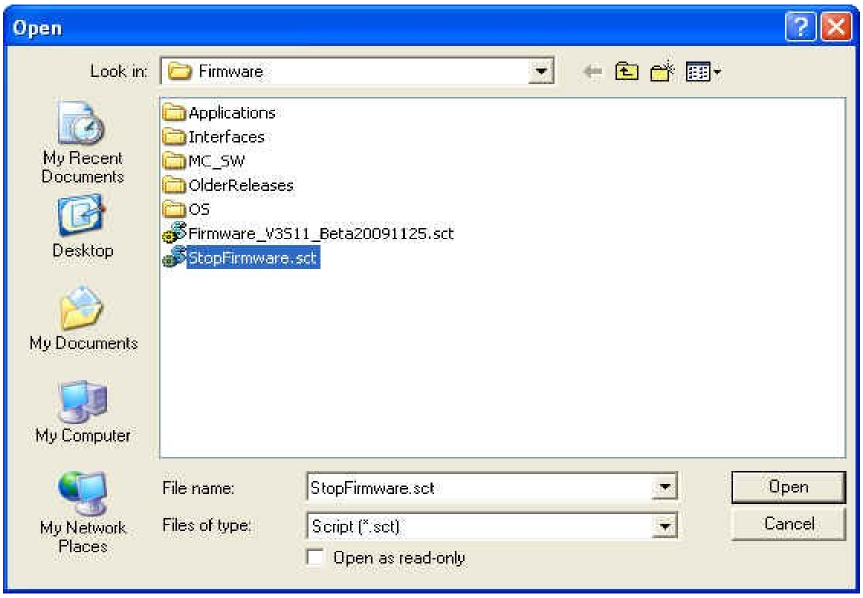
Value	Description
C1h	Checksum error
C2h	Message format error End of Telegram (04h) missing
C3h	Undefined Message Main ID
C4h	Undefined Message Sub ID
C5h	Wrong Baud Rate Defined With S1
C8h	Parameter Unknown UPID
C9h	Parameter Wrong Type

17 Troubleshooting

17.1 Stopping LinRS on B1100 Controllers

If you have installed the LinRS firmware on a B1100 controller the RS-port is occupied by the LinRS, so no configuration or login can be done over RS. It is recommended to use the USB to CAN converter. If you don't have one, you can follow the sequence below to have access over the RS port in the meantime.

1. Power down the B1100 servo
2. Launch the LinMot-Talk software (don't login, it's not possible)
3. Under: File / Install firmware select the stop script and open it



4. select the correct COM port
5. Power on the B1100 controller (24V Logic Supply)

6. The servo is now stopped (Firmware successfully stopped!) during booting, before starting the LinRS. Now the RS COM port is free for login and configuration or diagnostics.

This procedure can also be used to stop other controllers over the RS232 link.

18 Contact Addresses

SWITZERLAND	**NTI AG** Haerdlistr. 15 CH-8957 Spreitenbach	
	Sales and Administration:	+41-(0)56-419 91 91 office@linmot.com
	Tech. Support:	+41-(0)56-544 71 00 support@linmot.com
	Tech. Support (Skype) :	skype:support.linmot
	Fax: **Web:**	+41-(0)56-419 91 92 http://www.linmot.com/

USA	**LinMot, Inc.** 5750 Townline Road Elkhorn, WI 53121	
	Sales and Administration:	877-546-3270 262-743-2555
	Tech. Support:	877-804-0718 262-743-1284
	Fax:	800-463-8708 262-723-6688
	E-Mail: **Web:**	us-sales@linmot.com http://www.linmot-usa.com/

Please visit http://www.linmot.com/ to find the distribution near you.

Smart solutions are…