LinMot®

CANopen

Documentation of the CANopen Interface of the following Controllers:

- **E1100-CO (-HC,-XC)**
- **E1100-GP (-HC, -XC) (with CANopen Firmware loaded)**
- **B1100-GP (-HC, -XC) (with CANopen Firmware loaded)**

CANopen Interface 3.14

User Manual

Document version 3.14a / mk, May 2011

Table of Content

1 System overview

The LinMot CANopen controllers support the communication profile CiA DS301.
Further information on CANopen can be found under: http://www.can-cia.de/

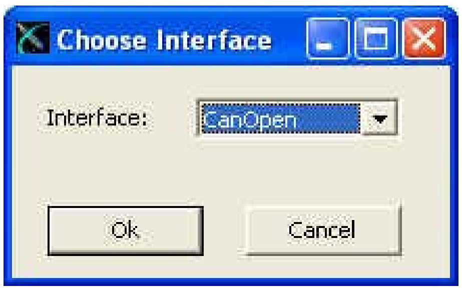
The following resources are available:
- 3 TxPDO
- 3 RxPDO
- 1 TxSDO
- 1 RxSDO

The supported protocols include:
- NMT Error Control (Node Guarding Protocol or Heartbeat Protocol)
- TxPDO (Transmission type 254, 250 and 1-240)
- RxPDO (Transmission type 254, 250 and 1)
- SDO Upload and Download
- NMT (Start, Stop, Enter PreOp, Reset Node, Reset Communication)
- Boot-Up Message

The baud rate can be selected by parameter or directly by BTR (bit timing register).

2 Installation on Servo Controller

For installing the CANopen firmware on the servo controller, start the LinMot-Talk software and press the install firmware button . Choose the file “Firmware_Buildxxxxxxxx.sct” and press “Open“. The wizard will guide you through the installation. When asking for the application software choose “CANopen”:

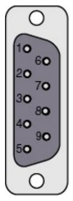

Choose Interface
Interface: CanOpen

Press ok and follow the rest of the wizard.

3 Connecting the CAN bus

3.1 *Pin assignment of the COM Connector (X5)*

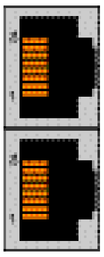
D-SUB 9 male:



Pin 1	RS-485 Y	Pin 6	RS-485 B
Pin 2	RS-232 TX	Pin 7	RS-485 Z
Pin 3	RS-232 RX	**Pin 8**	**CAN L**
Pin 4	RS-485 A	**Pin 9**	**CAN H**
Pin 5	**GND**		

3.2 *Pin assignment of the CMD Connector (X7, X8)*

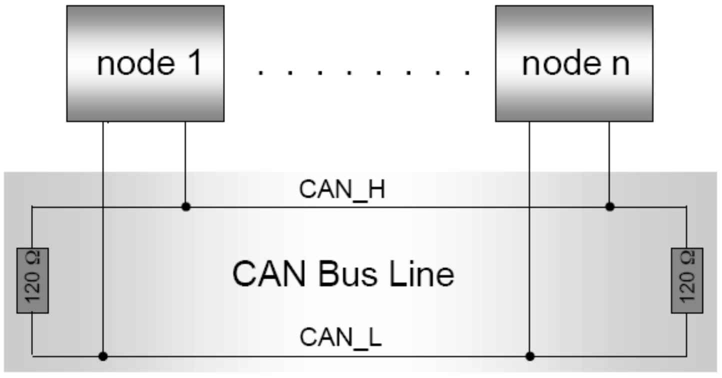
On the E1100-RS-xx, E1100-DP-xx and B1100-GP-xx controllers, the CMD connectors can be used to connect to the CANopen bus.
These are RJ45 connectors with 1:1 connected signals.
Use Ethernet cables according the EIA / TIA 568A standard.



Pin 1	RS485 A
Pin 2	RS485 B
Pin 3	RS485 Y
Pin 4/5	**Ground**
Pin 6	RS485 Z
Pin 7	**CAN H**
Pin 8	**CAN L**

3.3 CAN Termination

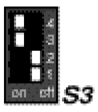
The CANbus must be terminated by two 120 Ohm resistors at both ends of the bus line, according the following figure:



For easy installation, the LinMot CANopen controller has built in termination resistors, which can be activated, if the LinMot controller is at the end of the bus line, and if there is no termination in the connector.

3.3.1 E1100

S3
ON – OFF
Interface
CAN Term
RS485 Term
RS485/232

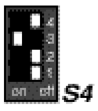


The built in termination resistor for the CAN bus can be activated by setting the DIP switch “CAN Term” to “ON”.

ATTENTION: For normal operation S3.4 (Interface) has to be set to ON!

3.3.2 B1100

S4
ON – OFF
Bootstrap
CAN Term
RS485 Term
RS485/232



The built in termination resistor for the CAN bus can be activated by setting the DIP switch “CAN Term” to “ON”.

ATTENTION: For normal operation S4.4 (Bootstrap) has to be set to OFF!

4 CANopen Parameters

The CANopen servo controllers have an additional parameter tree branch (Parameters → CANopen Interface), which can be configured with the distributed LinMot-Talk software. With these parameters, the CANopen behaviour can be defined. The LinMot-Talk software 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 CANopen going online. So in a first step the system can be configured and run without any bus connection.

CANopen Interface\ Dis-/Enable	
Disable	Servo controller runs without CANopen.
Enable	Servo controller runs only with a CANopen connection.

⚠ **IMPORTANT**: To activate the CANopen interface on E1100 controllers, the DIP switch “Interface” at the bottom of the drive has to be set to “ON”. This is not necessary for controllers of the B1100 series.

Baud Rate In this section the parameters for the baud rate selection are located.

Baud Rate Source Select

Defines the source of the baud rate definition.

E1100: CANopen Interface\ Baud Rate \Baud Rate Source Select B1100: OS\Communication\ CAN Configuration\ Baud Rate\ Baud Rate Source Select	
By Hex Switch S1 [1]	E1100 only: CAN bus baud rate dependent on S1 0 = By BTR 1 = 125 kBit/s 2 = 250 kBit/s 3 = 500 kBit/s 4 = 1 Mbit/s
By Parameter	The CAN bus baud rate is selected by the “Baud Rate Parameter”: - 125 kBit/s [1] - 250 kBit/s [2] - 500 kBit/s [3] - 1 Mbit/s [4]
By BTR	CAN bus baud rate is defined according to the Bit Timing Register
By DigIn 5 & 6 [2]	The baud rate is defined through the state of DigIn5 and DigIn6 at startup. DigIn6 is the most, DigIn5 the least significant bit. **DigIn6** **DigIn5** **Baud Rate** 0 0 125kBaud 0 1 250kBaud 1 0 500kBaud 1 1 1MBaud

Baud Rate BTR Value

For special applications where no standard setting for the baud rate works this parameter defines the bit timing for the CAN bus. The setting of the baud rate by Bit Timing Register is only necessary on special bus configurations: For example, if there are devices on the bus that have slow optocouplers.

Baud Rate Parameter Definition

The baud rate parameter defines the CAN bus baud rate for the CANopen connection.

CANopen Interface\ Baud Rate\ Baud Rate Parameter Definition	
125 kBit/s	CAN bus baud rate = 125 kBit/s
250 kBit/s	CAN bus baud rate = 250 kBit/s
500 kBit/s	CAN bus baud rate = 500 kBit/s
1 Mbit/s	CAN bus baud rate = 1 Mbit/s

[1] Parameter **not** available on controllers of the B1100 series.
[2] Parameter **only** available on controllers of the B1100 series.

MACID

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

MACID Source Select

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

E1100: CANopen Interface\ MACID\ MACID Source Select B1100: OS\ Communication\ MACID\ MACID Source Select	
By Hex Switch S2	E1100 only:The MACID is determined by the hex switch S2
By Hex Switches S1 and S2	E1100 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

Is the ID, when "By Parameter" is selected as source.

⚠ **E1100**
With the default settings, the MAC-ID and the baud rate are selected by the two rotary hex switches S1 and S2.

⚠ **B1100**
With the default settings, the MAC-ID and the baud rate are both selected by Parameter. The default values are 500kBit/s as baud rate and 63 (3Fh) for the MACID.

PDO Mapping

TxPDO 1 These parameters define the mapping of the transmit PDO 1. Four words can be mapped in total.

CANopen Interface\ PDO Mapping\ TxPDO 1	
Status Word [1W]	If this Boolean parameter is set, the status word is transmitted with TxPDO 1 (see variable 1D51h (E1100) / 6061h (B1100)).
State Var [1W]	If this Boolean parameter is set, the state var (high byte = state no. / low byte = sub state) is transmitted with TxPDO 1 (see variable 1B62h / 6968h (B1100)).
Logged Error Code [1W]	If this Boolean parameter is set, the logged error code is transmitted with TxPDO 1 (see variable 1D96h (E1100) / 6976h (B1100)).
Warn Word [1W]	If this Boolean parameter is set, the warn word (= bit coded warnings) is transmitted with TxPDO 1 (see variable 1D8Eh (E1100) / 6068h (B1100)).
Demand Current [1W]	If this Boolean parameter is set, the demand current value (= motor current) is transmitted with TxPDO 1 (see variable 1B93h (E1100) / E9E7h (B1100)).
Actual Position low word [1W]	If this Boolean parameter is set, the lower 16 bit of the actual position (32 bit value, see variable 1B8Dh (E1100) / F4D9h (B1100)) is transmitted with TxPDO 1.
Actual Position high word [1W]	If this Boolean parameter is set, then the higher 16 bit of the actual position (32 bit value, see variable 1B8Dh (E1100) / F4D9h (B1100)) is transmitted with TxPDO 1.
By UPID	This parameter can be used for free mapping of any parameter or variable to TxPDO 1 (mapping through Unique Parameter ID = UPID, 0 = no mapping). The corresponding data size in TxPDO 1 is either 1 word, if parameter or variable type is 16 bit or less, or 2 words, if the type is 32 bit.

TxPDO 2 These parameters define the mapping of the transmit PDO 2. Four words can be mapped in total.

CANopen Interface\ PDO Mapping\ TxPDO 2	
Motion Cmd Status [1W]	Feedback of the motion command header (toggle, etc?)
Actual Position 16 Bit [1W]	If this Boolean parameter is set, the actual motor position in 16 bit format is transmitted with TxPDO 2 (see variable 1B95h (E1100) / E9A5h (B1100)).
Demand Current [1W]	If this Boolean parameter is set, the demand current value (= motor current) is transmitted with TxPDO 2 (see variable 1B93h (E1100) / E9E7h (B1100)).
Demand Position 16 Bit [1W]	If this Boolean parameter is set, the demand position in 16 bit format is transmitted with TxPDO 2 (position setpoint, see variable 1B94h (E1100) / E9A4h (B1100)).
By UPID	This parameter can be used for free mapping of any parameter or variable to TxPDO 2 (mapping through Unique Parameter ID = UPID, 0 = no mapping). The corresponding data size in TxPDO 2 is either 1 Word, if parameter or variable type is 16 bit or less, or 2 Words, if the type is 32 bit.

TxPDO 3 These parameters define the mapping of the transmit PDO 3. Four words can be mapped in total.

CANopen Interface\ PDO Mapping\ TxPDO 3	
By UPID	This parameter can be used for free mapping of any parameter or variable to TxPDO 3 (mapping through Unique Parameter ID = UPID, 0 = no mapping). The corresponding data size in TxPDO 3 is either 1 Word, if parameter or variable type is 16 bit or less, or 2 Words, if the type is 32 bit.

RxPDO 1 These parameters define the mapping of the receive PDO 1. Four words can be mapped in total.

CANopen Interface\ PDO Mapping\ RxPDO 1	
Control Word [1W]	If this Boolean parameter is set, the control word has to be transmitted with RxPDO 1 (see variable 1D52h (E1100) / 6062h (B1100)).
Motion Cmd Header + Par Byte 0..3 [3W]	Motion command interface (Header and the first 4 bytes of the command parameters).
By UPID	For free mapping, every parameter or variable can be mapped by its UPID (Unique Parameter ID). The size is either 1 word, if type is 16 bit or less, or 2 words, if the type is 32 bit.

RxPDO 2 These parameters define the mapping of the receive PDO 2. Four words can be mapped in total.

CANopen Interface\ PDO Mapping\ RxPDO 2	
Motion Cmd Header + Par Byte 0..5 [4W]	Motion command interface (header and the first 6 bytes of the command parameters)
By UPID	For free mapping, every parameter or variable can be mapped by its UPID (Unique Parameter ID). The size is either 1 word, if type is 16 bit or less, or 2 words, if the type is 32 bit.

RxPDO 3 These parameters define the mapping of the receive PDO 3. Four words can be mapped in total.

CANopen Interface\ PDO Mapping\ RxPDO 3	
CMD Slave Header + Par Byte 6..7 [2W]	Command interface (slave header and byte 6..7 of the parameters)
CMD Slave Header + Par Byte 6..9 [3W]	Command interface (slave header and byte 6..9 of the parameters)
CMD Slave Header + Par Byte 6..11 [4W]	Command interface (slave header and byte 6..11 of the parameters)
Direct Par X [1W] [3]	Direct parameter channel for setting live parameters during runtime (only 16 bit parameters).
Direct Par X UPID [4]	UPID (Unique Parameter ID) of the selected parameter
By UPID	For free mapping, every parameter or variable can be mapped by its UPID (Unique Parameter ID). The size is either 1 word, if type is 16 bit or less, or 2 words, if the type is 32 bit.

[3] Parameter not available on controllers of the B1100 series.
[4] Parameter not available on controllers of the B1100 series.

PDO Configuration

TxPDO 1..3 These parameters define the bus parameters of the transmit PDO 1..3.

TxPDO 1..3 Enable
Selector for enabling/disabling the transmit PDO 1..3.

CANopen Interface\ PDO Configuration\ TxPDO 1..3\ TxPDO 1..3 Enable	
Disable	The PDO is deactivated
Enable	The PDO is activated

Transmission Type
This defines the transmission type according to DS 301. Default Value is 254 (Asynchronous with inhibit Time). Types 1-240 (cyclic synchronous) are supported as well.
If any of the TxPDOs has a synchronous transmission mode set, all RxPDOs are automatically evaluated synchronously.
The transmission type 250 is LinMot specific (it is reserved according to DS301). If transmission Type 250 is selected, the Transmit PDO is sent immediately after reception of the corresponding Receive PDO (TxPDO 1 corresponds to RxPDO 1). It can be used to realize a simple Poll-Request / Poll-Respond type bus structure. The “Legacy Sync WatchDog” feature can be used for monitoring (RxPDO 1 takes the function of the Sync).

Inhibit Time
Defines the minimal time between two send events.

Event Time
Defines the maximum time between two send events.

RxPDO 1..3 These parameters define the bus parameters of the receive PDO 1..3.

CANopen Interface\ PDO Configuration\ RxPDO 1..3	
Disable	The PDO is deactivated
Enable	The PDO is activated

CANopen Interface\ PDO Configuration\	
RxPDO 3 COB ID	0 = Default Mapping for RxPDO 3 COB ID xx= COB ID for RxPDO 3 (manual configuration)

Evaluate RxPDOs on SYNC with all TxPDOs asynchronous

These parameters defines the evaluation of RxPDOs on SYNC-messages.

CANopen Interface\ PDO Configuration\ Evaluate RxPDOs on SYNC with all TxPDOs asynchronous	
Disable	RxPDOs are not synchronously evaluated
Enable	RxPDOs are synchronously evaluated

This parameter is only in effect if all TxPDOs are configured for asynchronous transmission. If any of the TxPDOs has a synchronous transmission mode set, all RxPDOs are automatically evaluated synchronously. This setting can be used if one wants to send RxPDOs synchronously (e.g. for streaming-modes) but the response TxPDOs should only be transmitted asynchronously.

NMT Error Control

Nodeguarding Protocol

Directory for configuring the nodeguarding.

Nodeguarding Enable

Enable/Disable the node guarding feature.

CANopen Interface\ NMT Error Control\ Node Guarding Protocol\ Node Guarding Enable	
Disable	The Node Guarding Protocol is deactivated.
Enable	The Node Guarding Protocol is activated.
Guard Time	The Guard time, when Node Guarding is activated.

Guard Time

The Guard time, when Node Guarding is activated.

Heartbeat Protocol

These parameters configure the Heartbeat Protocol.

CANopen Interface\ NMT Error Control\ Heartbeat Protocol	
Produce	Cyclic Heartbeat is produced.
Consume	Cyclic Heartbeat is consumed
Producer Time	Cycle Time for producing Heartbeat
Consumer Time	Guarding Time for consumed Heartbeat
Consumed Node ID (Master)	Node ID of the Master

Legacy Sync Watchdog

These parameters configure the legacy watchdog of the Sync Telegram. This can be used together with Heartbeat or Node Guarding (CO firmware Version ≥3.8).

Watchdog Enable

Enabling/Disabling the legacy sync watchdog feature.

CANopen Interface\ NMT Error Control\ Legacy Sync Watchdog\ Watchdog Enable	
Disable	The Sync Watchdog is deactivated.
Enable	The Sync Watchdog is activated.
Sync Cycle Time	The expected Sync Cycle Time.

Sync Cycle Time

The Sync cycle is monitored with 1.5* Sync Cycle Time. This means that the real expected Sync Cycle Time can be configured here.

⚠ Only one NMT Error Control Protocol should be activated.

5 Mapping of the PDOs

5.1 Mapping Table

The PDOs are mapped by default according to the following scheme:

5.1.1 Receive PDOs

RxPDO 1	RxPDO 2	RxPDO 3	
Control Word	CMD Header	CMD Slave Header	
	Par 1	Par 4	
	Par 2	Direct Par Channel 1	
	Par 3		

Because the CMD interface of the LinMot controller consists of more than 8 Bytes, it's necessary to couple two PDOs together to ensure data consistency. This is done by the "CMD Slave Header". In order to execute a command both headers have to be toggled. On the slave Header only the last 4 bit are evaluated, so it's possible to simply copy the "CMD Header" from RxPDO 2 to the "CMD Slave Header" of RxPDO 3.

5.1.2 Transmit PDOs

TxPDO 1	TxPDO 2	TxPDO 3	
Status Word	CMD Status		
Run State	Actual Position		
Error Code	Actual Current		
Warn Word	Actual SetPosition		

If the application requires it, the mapping can be completely changed by the PDO Mapping parameter settings. Many applications do not require to use all resources.

5.1.3 Default Identifier

The default identifiers (11 Bit identifier) are allocated by the following scheme:

10	9	8	7	6	5	4	3	2	1	0
Function Code				Node ID						

This results in the following table:

Object	Function Code (binary)	COB ID (hex)	Object for Comm. Parameter / Mapping
NMT	0000	00h	- / -
SYNC	0001	80h	1005h / 1006h
Emergency	0001	81h – FFh	- / -
TxPDO 1	0011	181h – 1FFh	1800h
TxPDO 2	0101	281h – 2FFh	1801h
TxPDO 3	0111	381h – 3FFh	1802h
RxPDO 1	0100	201h – 27Fh	1400h
RxPDO 2	0110	301h – 37Fh	1401h
RxPDO 3	1000	401h – 47Fh	1402h
TxSDO	1011	581h – 5FFh	- / -
RxSDO	1100	601h – 67Fh	- / -

In the Pre-Operational state, this can be changed with SDO downloads by the master.

6 Motor Commands

Please refer to “Usermanual Motion Control Software”

7 State Machine

Please refer to “Usermanual Motion Control Software”

8 Interface Error Codes

Please refer to “Usermanual Motion Control Software” for the Error Codes of the MC Software. The CANopen Interface has the following additional Error Codes:

Error Code Hexadecimal	**Error Description**
C1h	The Controller is not compatible with CANopen
C2h	The configured ID is not valid (switches or parameter)
C3h	CANopen Error: Data out of Range
C4h	CANopen Error: Invalid Command
C5h	CANopen Error: Bus error
C6h	CANopen Error: general Bus error
C7h	CANopen Error: Bus error, stuff error
C8h	CANopen Error: Bus error, form error
C9h	CANopen Error: Bus error, ack error
CAh	CANopen Error: Bus error, bit 1 error
CBh	CANopen Error: Bus error, bit 0 error
CCh	CANopen Error: Bus error, CRC error
CDh	CANopen Error: Bus error, guard timeout
CEh	CANopen Error: Invalid UPID configured on Direct Par 1
CFh	CANopen Error: Invalid UPID configured on Direct Par 2
D0h	CANopen Error: Error: Invalid ID by Hex Switch S1
D1h	CANopen Error: Invalid Mapping in TxPDO 1
D2h	CANopen Error: Invalid Mapping in TxPDO 2
D3h	CANopen Error: Invalid Mapping in TxPDO 3
D4h	CANopen Error: Invalid Mapping in RxPDO 1
D5h	CANopen Error: Invalid Mapping in RxPDO 2
D6h	CANopen Error: Invalid Mapping in RxPDO 3
D7h	CANopen Error: Invalid UPID in TxPDO 1 Mapping
D8h	CANopen Error: Invalid UPID in TxPDO 2 Mapping
D9h	CANopen Error: Invalid UPID in TxPDO 3 Mapping
DAh	CANopen Error: Invalid UPID in RxPDO 1 Mapping
DBh	CANopen Error: Invalid UPID in RxPDO 2 Mapping
DCh	CANopen Error: Invalid UPID in RxPDO 3 Mapping

9 WarnWord

Please refer to “Usermanual Motion Control Software”

10 Object Dictionary

10.1 E1100

Index	Sub-Index	Description	Data Type	Value
0001h–001Fh		Data Types	DEFTYPE	
0020h		Communication Parameter	DEFSTRUCT	
	0h	Number of entries	UI8	
	1h	COB-ID	UI32	
	2h	Transmission type	UI8	
	3h	Inhibit time	UI16	
	4h	Reserved	UI8	
	5h	Event timer	UI16	
1000h		Device Type	UI32	0
1001h		Error register	UI8	
1008h		Manufacturer Device Name	Visible String	4 ASCII characters, which contain the last 4 characters of the article number.
1018h		Idendity Object	Record	
	0h	Number of Entries	UI8	4
	1h	Vendor ID	UI32	0000 0156h
	2h	Product Code	UI32	4 ASCII characters, which contain the last 4 characters of the article number.
	3h	Revision Number	UI32	
	4h	Serial Number	UI32	Serial Number UI32 encoded
2000h - 5FFFh		LinMot Parameters Index = 2000h + UPID	UI32	
	00h	Number of Entries		
	01h	RAM Value	SI32	RAM Value (rw)
	02h	ROM Value	SI32	ROM Value (rw)
	03h	Min Value	SI32	Minimal Value (ro)
	04h	Max Value	SI32	Maximal Value (ro)
	05h	Default Value	SI32	Default Value (ro)
	06h	RAM/ROM Write	SI32	RAM and ROM value can be written with the same value (wo)
	07h	Set ROM to default (OS)		Write anything to 2000h sub 7 to set all parameters of the OS to default values (wo). This command needs about 0.5s to finish.
	08h	Set ROM to default (MC)		Write anything to 2000h sub 8 to set all parameters of the MC Sw to default values (wo). This command needs about 2s to finish.
	09h	Set ROM to default (Interface)		Write anything to 2000h sub 9 to set all parameters of the CANopen Interface to default values (wo). This command needs about 0.5 s to finish.

	0Ah	Set ROM to default (Application)		Write anything to 2000h sub Ah to set all parameters of the Application to default values (wo)
	0Bh	Reset Controller		Write anything to 2000h sub Bh to reset the Controller (wo)
	20h	Start Getting UPID List		See chapter 11.3
	21h	Get Next UPID List item		See chapter 11.3
	22h	Start Getting Modified UPID List		See chapter 11.3
	23h	Get Next Modified UPID List item		See chapter 11.3
	35h	Stop MC and Application Software (for Flash access)		Write anything to 2000h sub 35h to stop the MC and Application SW (wo)
	36h	Start MC and Application Software		Write anything to 2000h sub 36h to start the MC and Application SW (wo)
	40h	Curve Service: Save to Flash		Write anything to 2000h sub 40h to save the curves from the RAM into the Flash ROM (wo)
	41h	Curve Service: Delete all Curves (RAM)		Write anything to 2000h sub 41h to Delete all Curves in the RAM (wo)
	42h	Curve Service: Poll Flash		Read anything from 2000h sub 42h to get the Flash state (r)
	50h	Curve Service: Add Curve		See chapter 11
	51h	Curve Service: Add Curve Info Block		See chapter 11
	52h	Curve Service: Add Curve Data		See chapter 11
	53h	Curve Service: Add Curve Data (32 Bit)		See chapter 11
	54h	Curve Service: Add Curve Info Block (32 Bit)		See chapter 11
	60h	Curve Service: Get Curve		See chapter 11
	61h	Curve Service: Get Curve Info Block		See chapter 11
	62h	Curve Service: Get Curve Data		See chapter 11
	70h	Get Error Log Entry Counter		See chapter 11.4
	71h	Get Error Log Entry Error Code		See chapter 11.4
	72h	Get Error Log Entry Time low		See chapter 11.4
	73h	Get Error Log Entry Time high		See chapter 11.4
	74h	Get Error Code Text Stringlet		See chapter 11.4
	80h	CT: Save to Flash		Write anything to 2000h sub 80h to save the Command Table from the RAM into the Flash ROM (w)
	80h	CT: Poll Flash		Read anything from 2000h sub 80h to get the Flash state (r)
	81h	CT: Delete all Entries (RAM)		Write anything to 2000h sub 81h to delete the complete Command Table in the RAM (wo)
	82h	CT: Delete Entry (Entry Nr.)		Write anything to 2000h + Entry Nr. Sub 82h to delete entry in the RAM

	83h	CT: Write Entry (Entry Nr.)		Write block size to 2000h + Entry Nr. Sub 83h to prepare entry in the RAM
	84h	CT: Write Entry Data		Write 2 Byte Data to 2000h + Entry Nr. Sub 84h, until block size has reached (the entry will be activated at this time)
	85h	CT: Get Entry (Entry Nr.)		Read the block size of 2000h + Entry Nr. Sub 85h.
	86h	CT: Get Entry Data		Read 2 byte data
	87h	CT: Get Entry List (Entry 0..31)		Read Bitfield (0=present)
	88h	CT: Get Entry List (Entry 32..63)		Read Bitfield (0=present)
	89h	CT: Get Entry List (Entry 64..95)		Read Bitfield (0=present)
	8Ah	CT: Get Entry List (Entry 96..127)		Read Bitfield (0=present)
	8Bh	CT: Get Entry List (Entry 128..159)		Read Bitfield (0=present)
	8Ch	CT: Get Entry List (Entry 160..191)		Read Bitfield (0=present)
	8Dh	CT: Get Entry List (Entry 192..223)		Read Bitfield (0=present)
	8Eh	CT: Get Entry List (Entry 224..255)		Read Bitfield (0=present)

10.2 B1100

Index	Sub-Index	Description	Data Type	Value
0001h–001Fh		Data Types	DEFTYPE	
0020h		Communication Parameter	DEFSTRUCT	
	0h	Number of entries	UI8	
	1h	COB-ID	UI32	
	2h	Transmission type	UI8	
	3h	Inhibit time	UI16	
	4h	Reserved	UI8	
	5h	Event timer	UI16	
1000h		Device Type	UI32	0
1001h		Error register	UI8	
1008h		Manufacturer Device Name	Visible String	4 ASCII Characters, which consist of the last 4 characters of the article number
1018h		Idendity Object	Record	
	0h	Number of Entries	UI8	1
	1h	Vendor ID	UI32	0000 0156h for LinMot
2000h		LinMot Parameters	UI32	
	00h	Number of Entries		
	01h	RAM Value of current UPID	SI32	RAM Value (rw)
	02h	ROM Value of current UPID	SI32	ROM Value (rw)
	0Bh	Reset Controller		Write anything to 2000h sub Bh to reset the Controller (wo)
	C0h	Update Current UPID		Write the current UPID the controller uses to get RAM and ROM values to the controller (wo)

11 Examples for controllers of the E1100 series

11.1 Write curve into the controller via CANopen

Add curve

A curve with the ID “CurveID” will be created. If a curve with the same ID already exists, an error will be generated.

Index	Sub-Index	Data	Result
2000h + CurveID	50h	InfoBlockSize (2 bytes) + DataBlockSize (2 bytes)	00h: No error D4h: Curve already exist

Example
LinMot MACID = 1
CuveID = 1
InfoBlockSize = 70 (0046h)
DataBlockSize = 164 (00A4h)

Index = 2001h
Sub-Index = 50h

CAN Telegram (8 Byte Data), COB-ID 601, PLC -> LinMot Controller:

		Index		Sub-Index	Info Block Size		Data Block Size	
Data	**23h**	**01h**	**20h**	**50h**	**46h**	**00h**	**A4h**	**00h**

Response: (8 Byte Data), COB-ID 581, LinMot Controller -> PLC:

		Index		Sub-Index	Unused Data		Result	
Data	**60h**	**01h**	**20h**	**50h**	**00h**	**00h**	**00h**	**00h**

Add Curve Info Block

Index	Sub-Index	Data	Result
2000h + CurveID	51h	Unused data (2 bytes) Info Block data (2 bytes)	04h: Info Block is not finished 00h: Info Block is finished D0h: Error: Info Block was already finished

Example
Index = 2001h
Sub-Index = 51h
Data = 0046h

CAN Telegram (8 Byte Data), COB-ID 601, PLC -> LinMot Controller:

		Index		Sub-Index	Unused Data		Info Block Data	
Data	**23h**	**01h**	**20h**	**51h**	**00h**	**00h**	**46h**	**00h**

Response: (8 Byte Data), COB-ID 581, LinMot Controller -> PLC:

		Index		Sub-Index	Unused Data		Result	
Data	**60h**	**01h**	**20h**	**51h**	**00h**	**00h**	**04h**	**00h**

Add Curve Info Block 32Bit

Index	Sub-Index	Data	Result
2000h + CurveID	54h	Info Block data (4 bytes)	04h: Info Block is not finished 00h: Info Block is finished D0h: Error: Info Block was already finished

Example
Index = 2001h
Sub-Index = 51h
Data = 0046h

CAN Telegram (8 Byte Data), COB-ID 601, PLC -> LinMot Controller:

		Index		Sub-Index	Info Block Data			
Data	**23h**	**01h**	**20h**	**51h**	**00h**	**00h**	**00h**	**00h**

Response: (8 Byte Data), COB-ID 581, LinMot Controller -> PLC:

		Index		Sub-Index	Unused Data		Result	
Data	**60h**	**01h**	**20h**	**51h**	**00h**	**00h**	**04h**	**00h**

Add Curve Data

Index	Sub-Index	Data	Result
2000h + CurveID	52h	Unused data (2 bytes) Data Block data (2 bytes)	04h: Data Block is not finished 00h: Data Block is finished D0h: Error: Data Block was already finished

Example
Index = 2001h
Sub-Index = 52h
Data = 2710h

CAN Telegram (8 Byte Data), COB-ID 601, PLC -> LinMot Controller:

		Index		Sub-Index	Unused Data		Data Block Data	
Data	**23h**	**01h**	**20h**	**52h**	**00h**	**00h**	**10h**	**27h**

Response: (8 Byte Data), COB-ID 581, LinMot Controller -> PLC:

		Index		Sub-Index	Unused Data		Result	
Data	**60h**	**01h**	**20h**	**52h**	**00h**	**00h**	**04h**	**00h**

Add Curve Data 32 Bit

Index	Sub-Index	Data	Result
2000h + CurveID	53h	Data Block data (4 bytes)	04h: Data Block is not finished 00h: Data Block is finished D0h: Error: Data Block was already finished

Example
Index = 2001h
Sub-Index = 53h
Data = 01312D00h

CAN Telegram (8 Byte Data), COB-ID 601, PLC -> LinMot Controller:

		Index		**Sub-Index**	**Data Block Data**			
Data	**23h**	**01h**	**20h**	**53h**	**00h**	**2Dh**	**31h**	**01h**

Response: (8 Byte Data), COB-ID 581, LinMot Controller -> PLC:

		Index		**Sub-Index**	**Unused Data**		**Result**	
Data	**60h**	**01h**	**20h**	**53h**	**00h**	**00h**	**04h**	**00h**

11.2 Read curve from controller via CANopen

Get curve

Index	Sub-Index	Data	Result (4 bytes)
2000h + CurveID	60h	-	00h: Curve exists D4h: Curve does not exist

Example
CuveID = 1

Result = 00 46 1401 ->	00: Curve exists 46: InfoBlock Size bytes 0114: DataBlock Size bytes
Result = D4 xx xxxx ->	D4: Curve does not exist

Get Curve Info Block

Index	Sub-Index	Data	Result (4 bytes)
2000h + CurveID	61h	-	04h: Info Block is not finished 00h: Info Block is finished D0h: Error: Info Block was already finished

Get Curve Data

Index	Sub-Index	Data	Result (4 bytes)
2000h + CurveID	62h	-	04h: Data Block is not finished 00h: Data Block is finished D0h: Error: Data Block was already finished

11.3 Get UPID List from Controller via CANopen

Start getting UPID List

Index	Sub-Index	Data	Result
2000h	20h	Start UPID (2 bytes)	00h: OK

Example
Index = 2000h
Sub-Index = 20h
Start UPID 1000h

CAN Telegram (8 Byte Data), COB-ID 601, PLC -> LinMot Controller:

		Index		Sub-Index	Unused Data		Start UPID	
Data	**23h**	**00h**	**20h**	**20h**	**00h**	**00h**	**00h**	**10h**

Response: (8 Byte Data), COB-ID 581, LinMot Controller -> PLC:

		Index		Sub-Index	Unused Data		Result	
Data	**60h**	**00h**	**20h**	**20h**	**00h**	**00h**	**04h**	**00h**

Get Next UPID List Item

Index	Sub-Index	Data	Result
2000h	21h	Address Usage	UPID found

When the end of the list is reached the UPID FFFFh is sent.

Example
Index = 2000h
Sub-Index = 21h
UPID found = 1004h
Address Usage = 000Dh

CAN Telegram (8 Byte Data), COB-ID 601, PLC -> LinMot Controller:

		Index		Sub-Index	Unused Data			
Data	**40h**	**00h**	**20h**	**21h**	**00h**	**00h**	**00h**	**00h**

Response: (8 Byte Data), COB-ID 581, LinMot Controller -> PLC:

		Index		Sub-Index	Address Usage		UPID found	
Data	**43h**	**00h**	**20h**	**21h**	**0Dh**	**00h**	**04h**	**10h**

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

The commands for getting the modified UPID List are used the same way.

11.4 Read the Error Log from the Controller

Get Error Log Entry Counter

Index	Sub-Index	Data	Result
2000h	70h	-	Number of Logged Errors Number of Occurred Errors

Example
Index = 2000h
Sub-Index = 70h
Number of Logged Errors = 0015h
Number of Occurred Errors = 0034h

CAN Telegram (8 Byte Data), COB-ID 601, PLC -> LinMot Controller:

		Index		Sub-Index	Unused Data			
Data	40h	00h	20h	70h	00h	00h	00h	00h

Response: (8 Byte Data), COB-ID 581, LinMot Controller -> PLC:

		Index		Sub-Index	Nr. of Logged Err		Nr. Of Occurred Err	
Data	43h	00h	20h	70h	15h	00h	34h	00h

Get Error Log Entry Error Code

Index	Sub-Index	Data	Result
2000h + Entry Nr.	71h	-	Error Code

Example
Index = 2005h
Sub-Index = 71h
Error Code of entry 5 = 64h (Cfg. Err: No Motor defined)

CAN Telegram (8 Byte Data), COB-ID 601, PLC -> LinMot Controller:

		Index		Sub-Index	Unused Data			
Data	40h	05h	20h	71h	00h	00h	00h	00h

Response: (8 Byte Data), COB-ID 581, LinMot Controller -> PLC:

		Index		Sub-Index	Unused Data		Error Code	
Data	43h	05h	20h	71h	00h	00h	64h	00h

Get Error Log Entry Time Low

Index	Sub-Index	Data	Result
2000h + Entry Nr.	72h	-	Time Low (milliseconds)

Example
Index = 2005h
Sub-Index = 72h
Time Low of entry 5 = 28C1h (=10433ms=10.433s)

CAN Telegram (8 Byte Data), COB-ID 601, PLC -> LinMot Controller:

		Index		Sub-Index	Unused Data			
Data	40h	05h	20h	72h	00h	00h	00h	00h

Response: (8 Byte Data), COB-ID 581, LinMot Controller -> PLC:

		Index		Sub-Index	Time Low		Time Mid Low	
Data	**43h**	**05h**	**20h**	**72h**	**C1h**	**28h**	**00h**	**00h**

Get Error Log Entry Time High

Index	Sub-Index	Data	Result
2000h + Entry Nr.	73h	-	Time High (hours)

Example
Index = 2005h
Sub-Index = 73h
Time High of entry 5 = 0398h (=920 hours)

CAN Telegram (8 Byte Data), COB-ID 601, PLC -> LinMot Controller:

		Index		Sub-Index	Unused Data			
Data	**40h**	**05h**	**20h**	**73h**	**00h**	**00h**	**00h**	**00h**

Response: (8 Byte Data), COB-ID 581, LinMot Controller -> PLC:

		Index		Sub-Index	Time Mid High		Time High	
Data	**43h**	**05h**	**20h**	**73h**	**98h**	**03h**	**00h**	**00h**

The Time of an entry consists of 32Bit hours and 32Bit milliseconds.

Get Error Code Text Stringlet

Index	Sub-Index	Data	Result
2000h + Error Code.	74h + (Stringlet No. 0..7)	-	4 Bytes of Error Code Text

Example
Index = 2064h (Error Code 64h = „Cfg Err: No Motor Defined“)
Sub-Index = 74h
Character 0..3 = 43 66 67 20 = “Cfg “

CAN Telegram (8 Byte Data), COB-ID 601, PLC -> LinMot Controller:

		Index		Sub-Index	Unused Data			
Data	**40h**	**64h**	**20h**	**74h**	**00h**	**00h**	**00h**	**00h**

Response: (8 Byte Data), COB-ID 581, LinMot Controller -> PLC:

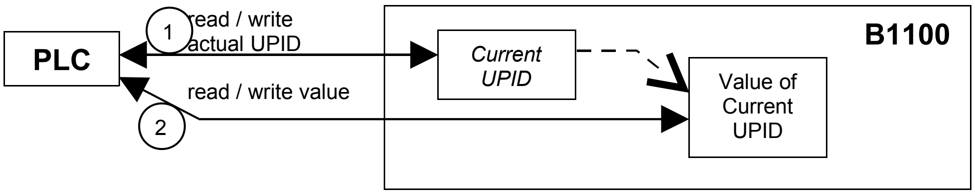
		Index		Sub-Index	Char 0	Char 1	Char 2	Char 3
Data	**43h**	**64h**	**20h**	**74h**	**43h**	**66h**	**67h**	**20h**

The Time of an entry consists of 32Bit hours and 32Bit milliseconds.

12 Examples for controllers of the B1100 series

Read or Write the value of a UPID of the controller via CANopen

Reading or writing the value of a UPID from/to the controller, has to be performed in two separate steps. First the UPID on which to operate (i.e. read or write) has to be sent to the controller via an SDO-command (index 2000h and sub index C0h). This UPID will be referred to as the actual UPID from here on. After this is done the value of the actual UPID can be read or written by other SDO-Commands (index 2000h and sub indices 01h and 02h).



12.1 Read the value of a UPID from a B1100 controller

1. Write the actual UPID which the controller uses

Index	Sub-Index	Data	Unused
2000h	C0h	UPID	-

Example
Index = 2000h
Sub-Index = C0h
UPID = E9E7h (UPID of the Demand Current)

CAN Telegram (8 Byte Data), COB-ID 601h, PLC -> LinMot Controller:

		Index		Sub-Index	UPID		Unused Data	
Data	**23h**	**00h**	**20h**	**C0h**	**E7h**	**E9h**	**00h**	**00h**

Response: (8 Byte Data), COB-ID 581h, LinMot Controller -> PLC:

		Index		Sub-Index	Unused Data			
Data	**60h**	**00h**	**20h**	**C0h**	**00h**	**00h**	**00h**	**00h**

2. Read the value of the actual UPID from the controller

Index	Sub-Index	Result
2000h	01h	Value of the actual UPID

Example
Index = 2000h
Sub-Index = 01h (Value is read from the RAM)

CAN Telegram (8 Byte Data), COB-ID 601h, PLC -> LinMot Controller:

		Index		**Sub-Index**	**Unused Data**			
Data	**40h**	**00h**	**20h**	**01h**	**00h**	**00h**	**00h**	**00h**

Response: (8 Byte Data), COB-ID 581h, LinMot Controller -> PLC:

		Index		**Sub-Index**	**UPID Parameter Value**			
Data	**43h**	**00h**	**20h**	**01h**	**DCh**	**FEh**	**FFh**	**FFh**

Returned value of the Demand Current	= FFFFFEDCh
	= -292 Dec. (Scale 0.001 A)
	= -0.292 A

12.2 Write the value of a UPID to a B1100 controller

1. Write the actual UPID which the controller uses

Index	Sub-Index	Data	Unused
2000h	C0h	UPID	-

Example
Index = 2000h
Sub-Index = C0h
UPID = E19Ch (UPID of the Maximal Current)

CAN Telegram (8 Byte Data), COB-ID 601h, PLC -> LinMot Controller:

		Index		Sub-Index	UPID		Unused Data	
Data	**23h**	**00h**	**20h**	**C0h**	**9Ch**	**E1h**	**00h**	**00h**

Response: (8 Byte Data), COB-ID 581h, LinMot Controller -> PLC:

		Index		Sub-Index	Unused Data			
Data	**60h**	**00h**	**20h**	**C0h**	**00h**	**00h**	**00h**	**00h**

2. Write the desired value of the actual UPID to the controller

Index	Sub-Index	Data
2000h	01h	Desired value of the actual UPID

Example
Index = 2000h
Sub-Index = 01h (The value is written to the RAM)

Desired value of the Maximal Current	= 0BB8h
	= 3000 Dec. (Scale 0.001 A)
	= 3 A

CAN Telegram (8 Byte Data), COB-ID 601h, PLC -> LinMot Controller:

		Index		Sub-Index	Desired UPID Parameter Value			
Data	**23h**	**00h**	**20h**	**01h**	**B8h**	**0Bh**	**00h**	**00h**

Response: (8 Byte Data), COB-ID 581h, LinMot Controller -> PLC:

		Index		Sub-Index	Unused Data			
Data	**60h**	**00h**	**20h**	**01h**	**00h**	**00h**	**00h**	**00h**

13 Reset Parameters to default values on E1100

There are three options to reset the parameters of a LinMot E1100 controller to default values:

1) By manipulating the two rotary hex switches:
   - Power Off the controller
   - Set the switches to FF
   - Power On the controller
   - Set the switches to 00
   - Wait for 10 s
   - Power Off the controller

2) By writing Index 2000h sub-index 7h, 8h, 9h, Ah of the Object dictionary.
   After changing the ROM values, a Reset should be performed either by a NMT Reset command or by Power OFF and ON the controller.

3) Reinstall the firmware will always reset the parameters to default values

14 Example for setting up a motion command

The following example shows the homing procedure and execution of a motion command via CANopen:

The PDO mapping is default:

RxPDO 1	RxPDO 2	RxPDO 3
Control Word	CMD Header	CMD Slave Header
	Par Byte 0…1	Par Byte 6…7
	Par Byte 2…3	
	Par Byte 4…5	

1) Homing (Control Word = 083Fh)

RxPDO 1
CAN Telegram (2 Byte Data), COB-ID 201h, PLC -> LinMot Controller:

Byte Nr.	0	1	2	3	4	5	6	7
Data	3Fh	08h	xx	xx	xx	xx	xx	xx

2) Enter Operational State (Control Word = 003Fh)

RxPDO 1
CAN Telegram (2 Byte Data), COB-ID 201h, PLC -> LinMot Controller:

Byte Nr.	0	1	2	3	4	5	6	7
Data	3Fh	00h	xx	xx	xx	xx	xx	xx

3) Execute Motion Command : VAI 16Bit Go To Pos (090xh)

CMD Header	→			0901h
CMD Slave Header	→			0901h
Par Byte 0…1	→	Target Position :	50mm	01F4h
Par Byte 2…3	→	Maximal Velocity :	1m/s	03E8h
Par Byte 4…5	→	Acceleration :	10m/s^2	0064h
Par Byte 6…7	→	Deceleration :	10m/s^2	0064h

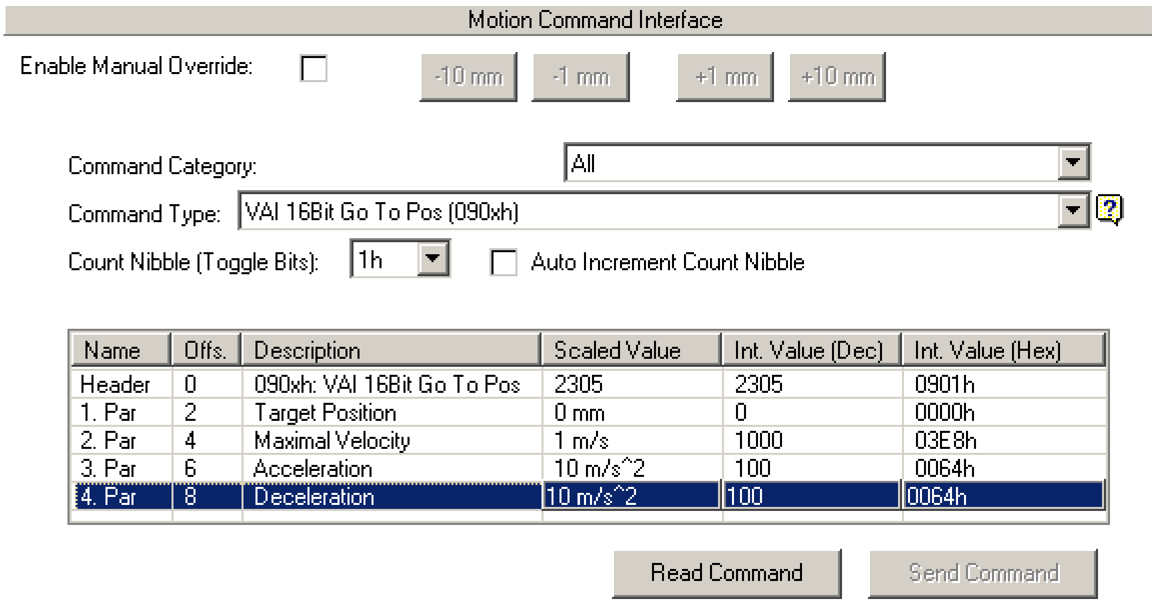
RxPDO 2
CAN Telegram (8 Byte Data), COB-ID 301h, PLC -> LinMot Controller:

Byte Nr.	0	1	2	3	4	5	6	7
Data	01h	09h	F4h	01h	E8h	03h	64h	00h

RxPDO 3
CAN Telegram (4 Byte Data), COB-ID 401h, PLC -> LinMot Controller:

Byte Nr.	0	1	2	3	4	5	6	7
Data	01h	09h	64h	00h	xx	xx	xx	xx

As it appears with LinMot-Talk after «Read Command» in the Control Panel :



Enable Manual Override: ☐

Command Category: All

Command Type: VAI 16Bit Go To Pos (090xh)

Count Nibble (Toggle Bits): 1h

☐ Auto Increment Count Nibble

Name	Offs.	Description	Scaled Value	Int. Value (Dec)	Int. Value (Hex)
Header	0	090xh: VAI 16Bit Go To Pos	2305	2305	0901h
1. Par	2	Target Position	0 mm	0	0000h
2. Par	4	Maximal Velocity	1 m/s	1000	03E8h
3. Par	6	Acceleration	10 m/s^2	100	0064h
4. Par	8	Deceleration	10 m/s^2	100	0064h

15 Quick Start Guide for advanced users

The aim of this chapter is to help users who are already familiar with the LinMot-Controllers and the LinMot-Talk software with the setup which is needed to get the controller up and running in CANopen-Network.

15.1 Hardware-Setup

Set up the hardware as described in chapter 3 of this manual.

15.2 Configuration of the controller

The default value for the MACID (Node ID) is « 1 » for E1100 controllers and « 63 » for B1100 controllers. The default baud rate is «500 kBaud».
If different settings are to be used, those parameters have to be properly configured first. This has to be done with the LinMot-Talk software.

15.3 Starting the device

The CANopen Network Management (NMT) protocol allows starting of devices with a single NMT-Telegram :

CAN Telegram (2 Byte Data), COB-ID 000, PLC -> LinMot Controller:

Byte Nr.	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Data	**01h**	**MACID**	**xx**	**xx**	**xx**	**xx**	**xx**	**xx**

The first byte identifies the Start Remote Node command, the second byte is the MACID of the controller. If 00h is used as the MACID, all Nodes in the network are started. This command puts the controller in operational mode.

After this all SDOs and PDOs can be used.
The default configuration for the transmission type of the PDOs is 254 (Asynchronous transmission with inhibit Time).

16 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:	+41-(0)56-419 91 92	
	Web:	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:	us-sales@linmot.com
	Web:	http://www.linmot-usa.com/

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

Smart solutions are…

driven by
LinMot®
www.LinMot.com