LinMot®

PROFIBUS PROCESS FIELD BUS

DeviceNet™

CANopen

RS 232

RS 485

For the following LinMot controllers

B1100 series

E1100 series

LinMot Servo Controller Configuration over Fieldbus Interfaces

Document version 1v0v3/ mk/Ro, June 2010

Table of Content

1 Introduction

Users of LinMot servo controllers (series E1100 and B1100) can easily setup their drive by using the LinMot-Talk software. Beside other functionality (firmware download, monitoring, PLC emulation, etc.), the LinMot-Talk software is used for altering the firmware configuration parameters, for creating and up- and downloading of curve profiles.

Most of the LinMot servo controllers are equipped with a fieldbus interface to the superior control system (PLC, IPC). This interface is used for controlling the servo controller under normal operation conditions (read/write of control and status word, sending motion commands, etc.).

If the LinMot servo controller uses a fieldbus connection for the communication to the superior control system (PLC, IPC), then the same fieldbus interface can be used for configuration purposes as well. The following fieldbus interfaces are supported: Profibus DP, CANOpen, DeviceNet, RS232 and RS485 (using LinRS protocol).

This document describes in general the access to the configuration parameters and the curve data over fieldbus interfaces. Detailed information about how data access is implemented in the respective interfaces can be found in the corresponding interface user manuals.

NOTE: The series E1100, E1200 and B1100 differ in some functions from each other. If nothing is indicated, the behavior is the same. If E1200 is not mentioned explicitly the behavior is the same as E1100 does.

2 Firmware Parameters

2.1 Overview

The LinMot controller firmware has to be configured through its parameters in order to meet the needs of the application where the servo system has to be integrated. Typical examples of firmware parameters that must be set during the commissioning process are motor definition parameters, position control parameters, etc.

The easiest way to alter parameters is to use the LinMot-Talk software tool. The software displays the parameters in a comfortable tree structure. Most of the parameters are displayed as a scaled value with the corresponding physical unit.

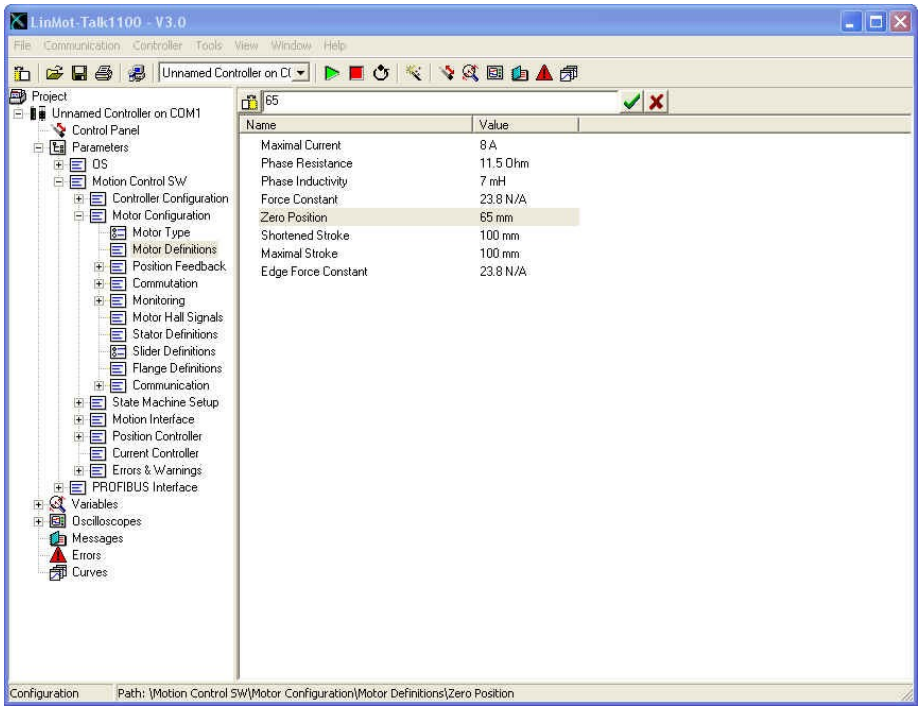

Figure 1: Firmware parameters listed in a tree structure

2.2 Unique Parameter ID and Raw Data Value

The value of any parameter is stored as an integer value (raw data) in the memory space of the controller. The parameter is identified through its Unique Parameter ID (UPID), which is a 16 Bit integer number.

Both, UPID and raw data value of any parameter can be displayed in the LinMot-Talk parameter tree structure (press Show/Hide Details button).

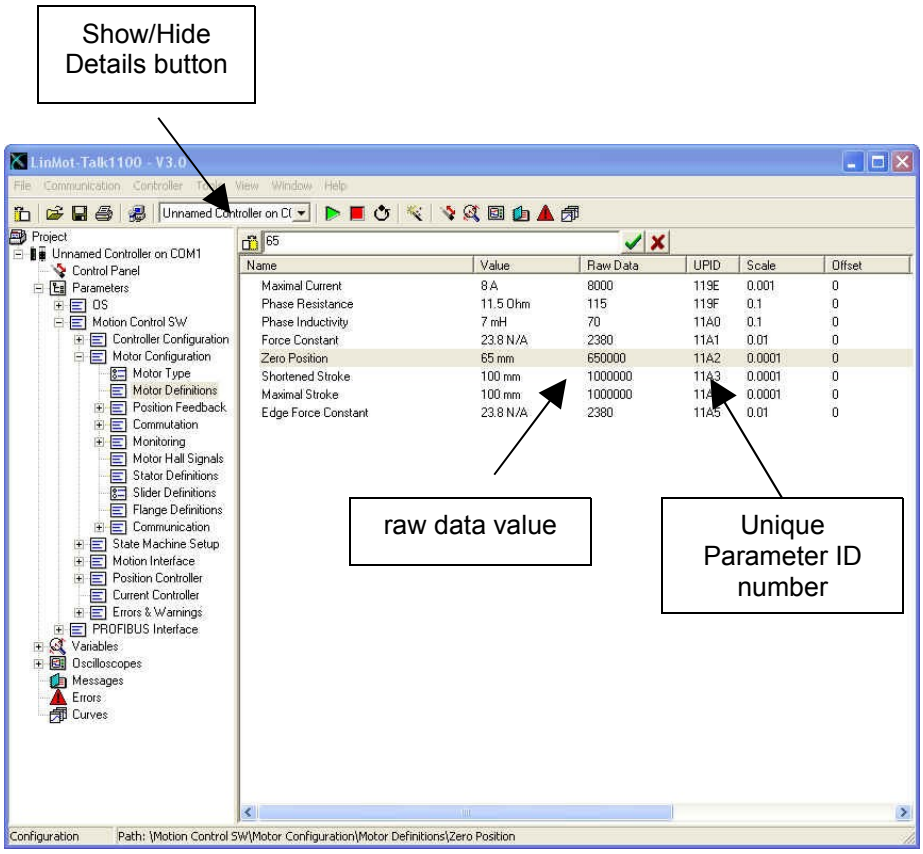

Figure 2: Detailed view for parameters

2.3 ROM and RAM Value

The value of any parameter is stored in the non-volatile memory area of the LinMot controller (ROM value). During the boot-up process the controller's operating system creates a copy of the non-volatile memory block to its RAM memory space. Thus, after the controller has powered up, each parameter has two memory locations and values: one in the ROM, the other in the RAM.

The firmware uses at run time exclusively the RAM values of the parameters in its control tasks (fast data access). The controller's

operating system and the fieldbus interfaces provide independent access to ROM and RAM values.

When parameters are altered using the LinMot-Talk software, the ROM and/or the RAM value are affected:

- When 'live' parameters are changed, then the LinMot-Talk software writes to ROM and RAM memory.

- 'Non-Live' parameters are written only to ROM (and are copied to RAM at next firmware start).

The LinMot-Talk software reads and displays the ROM values. It reads the parameter values only once (during the login process). The PC software allows altering 'non-live' parameters only if the firmware is stopped.

Over the fieldbus interface the RAM value of 'non-live' parameters cannot be changed. Changing the ROM value is possible even when the firmware is running (except for read-only parameters).

Changing the RAM value of a parameter immediately influences the system behavior when the firmware is running (e.g. control parameters of the position control loop).

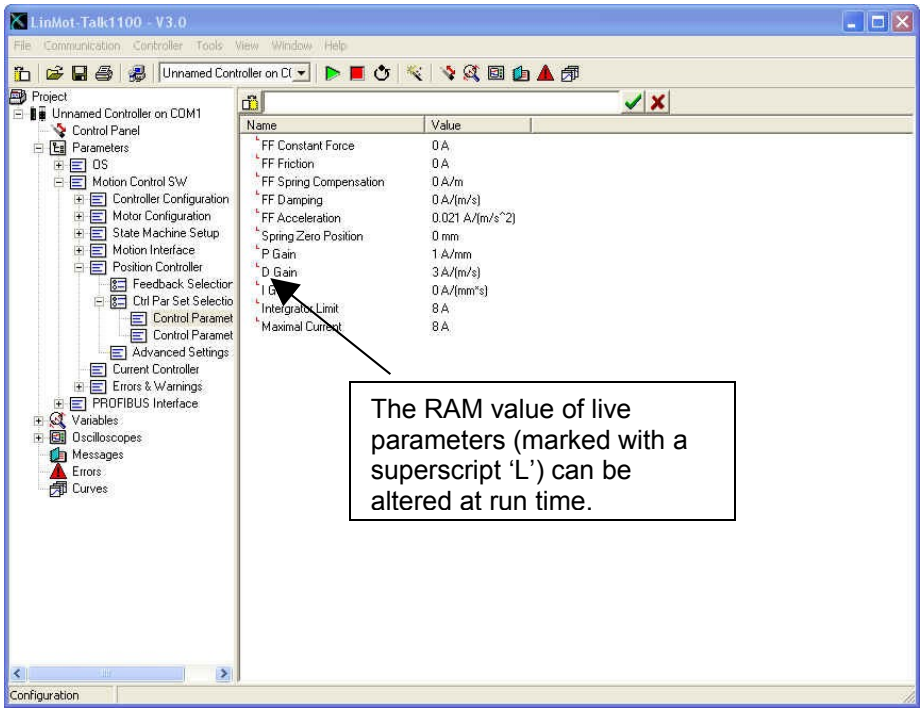

Figure 3: Live Parameters marked with a superscript 'L'

Changes to ROM values don't affect the system behavior until the next firmware startup (e.g. after a software reset initiated by the superior control system or stop/start from the LinMot-Talk software).

2.4 Default Value

Beside the actual RAM and ROM value each parameter has its default value. The default value is displayed in the detailed parameter view of the LinMot-Talk software.

During firmware installation the ROM values are set to the default values of the corresponding parameters.

E1100 controllers provide resetting the parameters to their default values over fieldbus interfaces. Resetting to default value is possible for single parameters (via UPID) or for all parameters of any firmware layer at once (see below).

NOTE: B1100 do not support this feature, because the default values are not stored on the controller.

2.5 32 Bit Access for any Parameter Type

The whole configuration consists of parameters of different types (bit, byte, 16bit integer, 32bit integer and string parameters). In order to keep the interface as simple as possible any parameter can be accessed as a 32bit integer value. The controller's operating system will filter out the relevant number of bits for each parameter.

Since string parameters can be longer than 4 characters (= 4 Bytes), a single 32bit integer value is not sufficient to define a general string. Therefore strings are handled in a special way:

- In the LinMot-Talk software to each string parameter one single UPID is shown.
- Internally the string is split into parts (so called 'stringlets') of 4 characters (= 4 bytes = 32 bits).
- Each stringlet has its own UPID.
- For **E1100 and E1200** controllers: The UPID of the first stringlet is the string UPID plus 1, the UPID of the second stringlet is the string UPID plus 2 and so on.
- For **B1100** controllers: The UPID of the first stringlet is the string UPID itself, the UPID of the second stringlet is the string UPID plus 1 and so on.

The following example shows the principle of converting strings to raw data values.

Example:

Writing the string 'X-Axis Left' to the parameter 'User Comment':

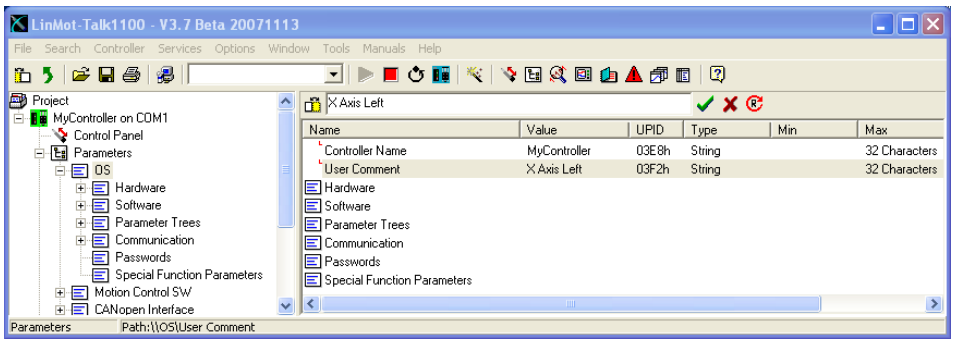

Figure 4: String parameter example for E1100 & E1200 (see UPID 03F2h)

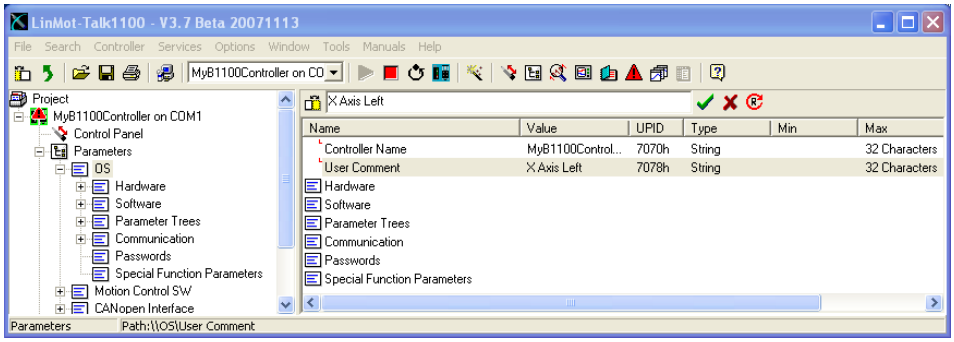

Figure 5: String parameter example for B1100 (see UPID 7078h)

The following table shows how the stringlet UPIDs and the corresponding 32bit integer values are determined:

Stringlet	"X Ax"	"is L"	"eft"
Parameter UPID **E1100 & E1200**	03F3h	03F4h	03F5h
Parameter UPID **B1100**	7078h	7079h	707Ah
Ordinal 1st Char:	Ord('X')=58h	Ord('i')=69h	Ord('e')=65h
Ordinal 2nd Char:	Ord(' ')=20h	Ord('s')=73h	Ord('f')=66h
Ordinal 3rd Char:	Ord('A')=41h	Ord(' ')=20h	Ord('t')=74h
Ordinal 4th Char:	Ord('x')=78h	Ord('L')=4Ch	00h
Parameter Value:	78412058h	4C207369h	00746665h

Table 1: The four firmware layers

3 Firmware Layer Concept

The firmware on the servo controller consists of up to four layers:

Layer	Name	Layer Functionality
1	Operating System	- Resource Management - Communication with LinMot-Talk - Start/Stop of the other SW layers - Parameter / Variable Service - Oscilloscope Service - Message / Error Service
2	Motion Control Software	- Current Control Loop - Position Control Loop - Set Value Generation - Monitoring
3	Interface Software	- Communication to superior control system (e.g. via Profibus, CANOpen, DeviceNet, LinRS, etc.)
4	Application	- Customized firmware extensions

Table 2: The four firmware layers

Each firmware part has its own parameters located in a separate branch of the parameter tree.

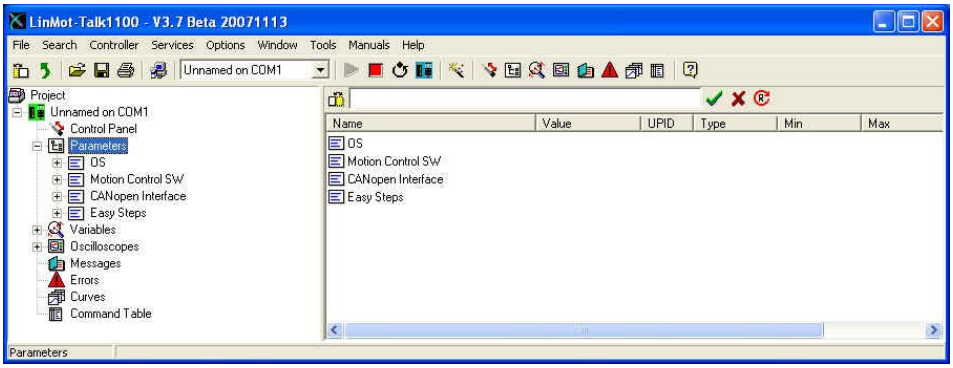

Figure 6: Parameter tree with branches for the different firmware layers

On E1100 & E1200 controllers, each firmware layer has its own range of UPIDs for its parameters and variables.

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

Table 3: UPID value ranges for E1100 controllers.

NOTE: On B1100 controllers, the UPIDs of the different software layers are mixed and cannot be grouped.

The parameter definitions are stored in the controller. On E1100 controllers the UPID, RAM/ROM locations, default values, parameter type, min/max values and access rights are stored. On B1100 controllers the UPID, RAM/ROM locations and the type are stored. The corresponding definition files have been downloaded together with the firmware when the firmware was installed on the controller. Information about the currently installed parameter tree files can be found in the operating system parameter tree branch.

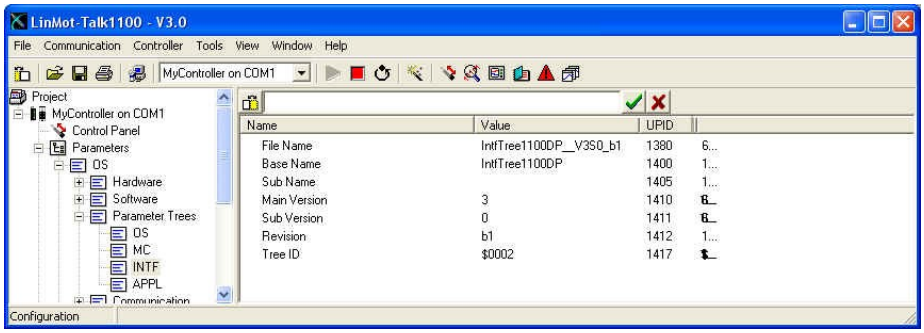

Figure 7: Parameter tree information

The parameter subtree (e.g. Profibus parameter tree V3.0 etc.) is defined through its Tree ID, Main Version and SubVersion value. This information can be captured by reading out the values of the parameters with UPID according to the following table. These parameters are used to perform compatibility tests before parameter configurations are downloaded to controllers (see below).

Layer	Tree ID UPID E1100/B1100	Main Version UPID E1100/B1100	Sub Version UPID E1100/B1100
1 OS	0539h / 2CA1h	0532h / 2C96h	0533h / 2C97h
2 MC SW	0561h / 2FA1h	055Ah / 2F96h	055Bh / 2F97h
3 Interface SW	0589h / 2EA1h	0582h / 2E96h	0583h / 2E97h
4 Application	05B1h / 2DA1	05AAh / 2D96	05ABh / 2D97h

Table 4: UPID value ranges for E1100 controllers.

4 Parameter Configuration Compatibility Rules

The current parameter configuration can be stored into a PVL file. This text file contains the parameter UPID and value in an easy to interpret list. This parameter list can easily be stored in a superior machine controller (PLC, IPC) and downloaded to any servo controller of the same type if necessary.

Besides the parameters raw data the PVL file contains additional information about the system on which the PVL file was created.

Before complete parameter configurations are downloaded from the superior machine control system to the LinMot controller, it is recommended to perform some compatibility checks. With these checks it can be ensured, that the configuration stored in the superior control system (source file) is fully compatible with the firmware installed controller (target system).

The following rules must be considered for each firmware layer:

Check Value	Rule
Tree ID	Tree ID of the source file and target system must be the same (e.g. it is not possible to download a DeviceNet setup onto a controller with Profibus interface firmware installed).
Tree Main Version	The source file parameter tree must have the same Main Version as the tree installed on the target system.
Tree Sub Version	The Sub Version of the source must be the same or smaller than the corresponding value of the target system (backward compatibility within main version).

Table 5: UPID value ranges for E1100 controllers.

5 Curves

NOTE: This chapter is only valid for E1100 controllers. B1100 controllers do NOT support curves.

The E1100 controllers can store up to 100 curves, which can be generated and downloaded by the LinMot-Talk software. The curves are identified by their ID (1..100).

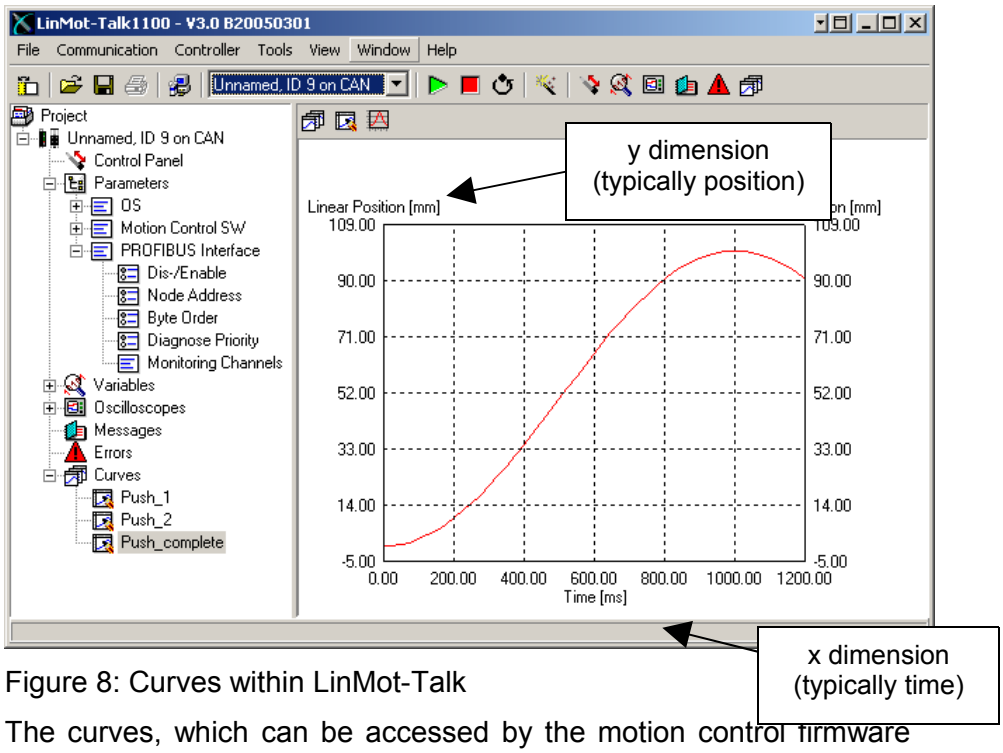

Figure 8: Curves within LinMot-Talk

The curves, which can be accessed by the motion control firmware through corresponding motion commands, are located in the RAM memory space of the controller. They can be permanently stored into a flash memory block as well. At boot time the operating system copies the curve data from the flash memory to the RAM memory block.

The curve profiles, which are generated and downloaded using the LinMot-Talk software, are always stored to RAM and flash memory (permanently saved).

It is also possible to download curves to the controller over the fieldbus. Then it is necessary to have the curve objects stored in the superior machine controller. There are three ways to bring the profiles there:

1. The curves are created and downloaded to a controller with LinMot-Talk. Then the curve objects can be uploaded from the controller over the fieldbus interface (using the curve service).

2. The curves are created and then exported to a raw data file (PVL file) using the LinMot-Talk software. Then the PVL file can be loaded to the main controller.

3. The curve object is fully generated by the main machine controller without using the LinMot-Talk curve tool.

5.1 Curve Object

Each curve object consists of a curve info block (header) and a curve data block (setpoints). The curve service provides commands for reading/writing these blocks. For further details how the curve service is implemented in the various fieldbus interfaces please consult the corresponding interface manuals.

Curve Info Block		
Byte Off	Type	Name
0..1	UInt16	Data offset
2..3	UInt16	Object type
4..5	UInt16	Number of setpoints
6..7	UInt16	Data Type size
8..29	String	Name
30..31	UInt16	Curve ID
32..35	UInt32	x-Length
36..37	UInt16	XDimUUID
38..39	UInt16	YDimUUID
40..41	UInt16	Wizard Type
42..45	UInt32	Wizard Par 1
46..49	UInt32	Wizard Par 2
50..53	UInt32	Wizard Par 3
54..57	UInt32	Wizard Par 4
58..61	UInt32	Wizard Par 5
62..65	UInt32	Wizard Par 6
66..69	UInt32	Wizard Par 7
Curve Data Block		
0..xxx	Curve Setpoints	

Table 6: Curve Object: Info Block and Data Block

5.2 Curve Info Block

Data Offset

The *Data Offset* contains the info block size information. The software expects the info block to consist of 70 bytes. So the first word of the info block must have the value 70 (= 46h).

Object Type

The *Object Type* word consists of four nibbles (1 nibble = 4 bits):

- lowest nibble: Object Version (must be 3)
- lower middle nibble: Type of Object (Curve = 0h)
- higher middle nibble: X dimension Code
- highest nibble: Y dimension Code

Curve Info Block: Object Type									
Byte	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Type & Version	Type of Object: 0h (=Curve)				Version: 3h (=SW-Version 3.x)			
1	X & Y dimension codes	**Y dimension code:** 0: Position 1: Velocity 2: Current 3: Encoder Pos (Increments) 4: Encoder Speed 5: MicroSteps				**X dimension code:** 0: Time 1: Encoder Pos (Increments) 2: Position			

Table 7 Curve Info Block: Object Type

According to the definitions above the *Object Type* has the following value:

- Position vs. Time curve: 0003h
- Cam profiles (Pos vs. Enc. Pos.): 0103h

Other object types are not supported yet.

Number of Setpoints

The number of setpoints of the profile is given as 16bit value. The minimal number of setpoints is 2.

Data Type Size

This value defines the size of one setpoint. Position values are defined in 32bit format (32bit = 4 bytes).

Name

In order to make identification easier, a descriptive name can be defined (e.g. 'Fast Move Out' or 'Retraction'). 22 bytes are reserved for the name string. The string is terminated with 00h.

If the profile is machine generated (PLC-Program, etc.), the name space will be typically filled with 00h.

Curve ID

The *Curve ID* must be unique. Allowed values are : 1..100 (0001h..0064h).

X-Length

The *X-Length* defines the base length of the curve profile.

- Position vs. Time Curve : Time [10us]
- Cam profiles (Pos vs. Enc. Pos.) : Encoder Pos [Increments]

X/Y-Dimension UUID

The following Unique Unit IDs (UUIDs) are supported:

Unit Definition		
UUID	Unit Scaling	Description
0005h	0.1 um (=1*10^{-7}m)	Standard Linear Position Unit
001Ah	0.01 ms (=1*10^{-5}s)	Standard Curve Time Unit
001Bh	1 Increment	Standard Encoder Position Unit

Table 8: Unit Definitions

According to the table above the following values are correct:

Position vs. Time Curves : XDimUUID = 001Ah
YDimUUID = 0005h

Cam Profiles : XDimUUID = 001Bh
YDimUUID = 0005h

Wizard Information

The wizard information (wizard type & wizard parameters) is used only by the LinMot-Talk software. All those bytes should be set to 00h for curves which are not generated with the LinMot-Talk curve tool.

5.3 Curve Data Block

The curve data block contains the setpoints (Y-dimension). The size of the block is: *No Of Setpoints * Data Type Size*

The setpoints are equally spaced over the x-length. The x-dimension equidistance is: *x-Length / (No Of Setpoints – 1).*

5.4 Erasing all curves in the RAM of the Controller

All curves in the RAM of the controller can be deleted by the following command:

-Curve Service: Delete all Curves (RAM)

5.5 Uploading Curves from the Controller

To read a curve from the controller using the fieldbus interface, the following commands have to be used:

-Curve Service: Get Curve

-Curve Service: Get Curve Info Block

-Curve Service: Get Curve Data

The curve service commands are described in the corresponding interface manuals.

Reading a curve is according the following scheme:

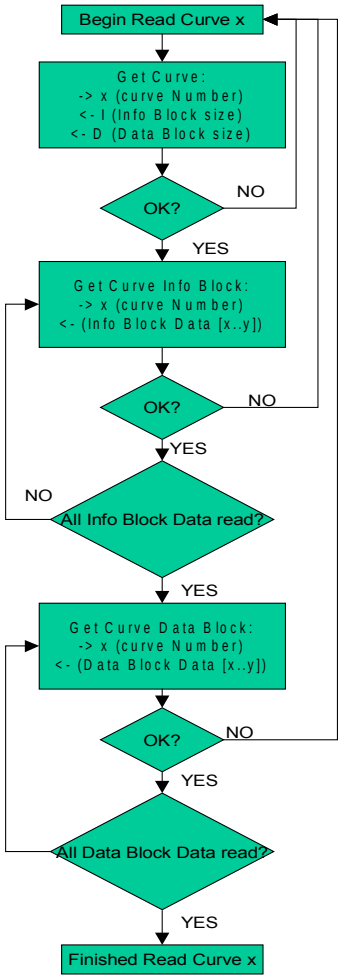

Figure 9: Flowchart Read Curve

5.6 Downloading Curves into the RAM of the Controller

To write a curve into the RAM of the controller, the following commands have to be used:

-Curve Service: Add Curve

-Curve Service: Add Curve Info Block

-Curve Service: Add Curve Data

The writing of a curve is always according to the following scheme:

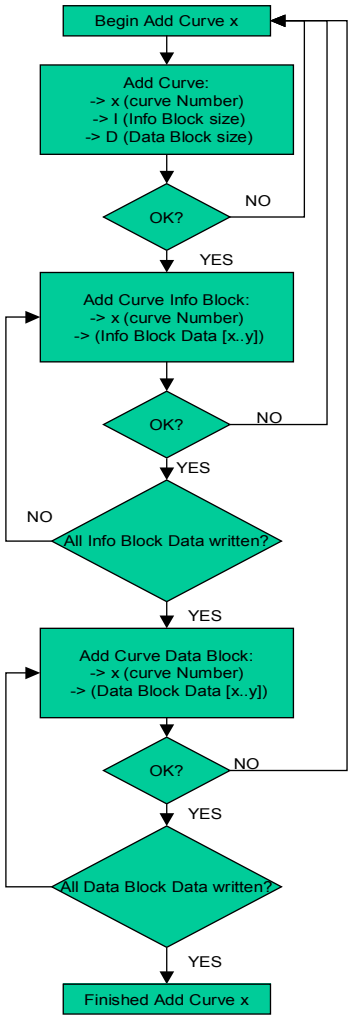

Figure 10: Flowchart Add Curve

6 Command Table

NOTE: This chapter is only valid for E1100 controllers. B1100 controllers do NOT support command tables.

For programming simple sequences with decisions the LinMot servo controller supports the command table (CT) programming utility. Up to 255 CT entries can be programmed. The entries can be arranged in sequences and some branch possibilities are supported.

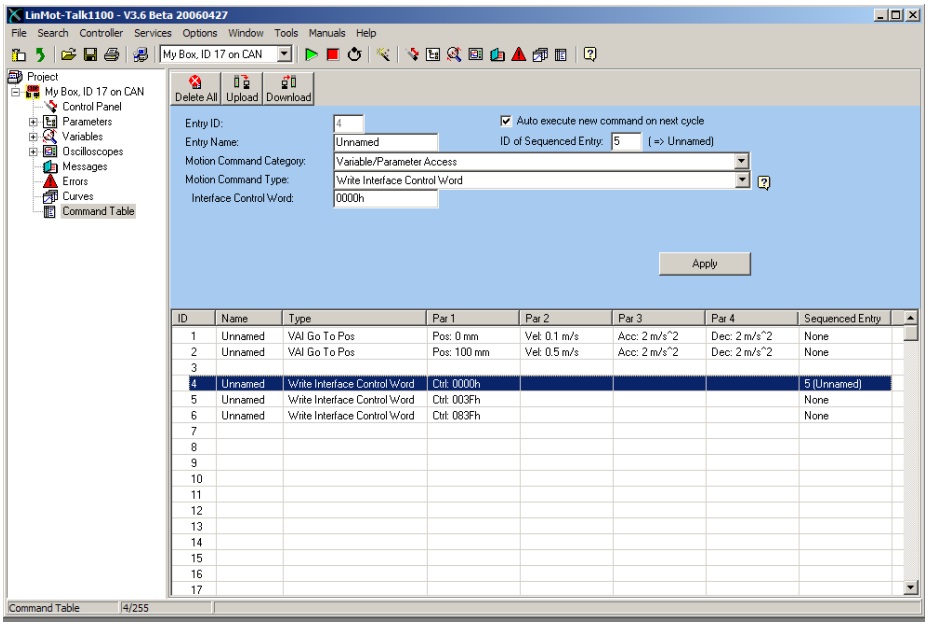

Figure 11: Command Table within LinMot-Talk

The CT entries can be accessed by the motion control firmware through corresponding motion commands or through digital IOs. The firmware uses the table data that is located in the RAM memory space of the controller. The CT can be stored permanently into the flash memory as well. At boot time the operating system copies the entire CT from the flash memory to the RAM.

If the CT entries are generated and downloaded using the LinMot-Talk 4 software, they are always stored into RAM and flash memory (permanently saved).

It is also possible to download or modify CT entries in the controller over a fieldbus. Then it is necessary to have the CT entry data stored in the superior machine controller. There are three ways to bring the entry data to the LinMot controller:

1. The CT entries are created and downloaded to a controller by using LinMot-Talk. Afterwards the CT entry data can be uploaded from the controller over the fieldbus interface (using the CT service).

2. The CT entries are created and then exported to a raw data file (PVL file) using the LinMot-Talk software. This generated file can be loaded to the main controller.

3. The CT entry data is fully generated by the main machine controller without using the LinMot-Talk command table editor.

6.1 Command Table Entry Format

Each CT entry consists of one bit in a presence list (bit = 0 entry exists) and a command table entry data block of 64 bytes size. The CT service provides commands for reading/writing those blocks. For further details how the CT services can be used for reading and writing CT entries over a fieldbus interface, please consult the corresponding interface manual.

Command Table Entry Presence List	
Byte Off	Description
00h..03h	Bit field for entries 1..31 (Bit = 0 entry exists)
04h..07h	Bit field for entries 32..63
08h..0Bh	Bit field for entries 64..95
0Ch..0Fh	Bit field for entries 96..127
10h..13h	Bit field for entries 128..159
14h..17h	Bit field for entries 160..191
18h..1Bh	Bit field for entries 192..223
1Ch..1Fh	Bit field for entries 224..255

Table 9: Command Table Entry Saving Format

Command Table Entry Data Block	
Byte Off	Description
00h..01h	Command entry Version ID fix A701h
02h..03h	Linked Command Entry ID (ID=FFFFh not linked)
04h..05h	Motion Command Header
06h..25h	Motion Command Parameters
26h..35h	Entry Name (0 terminated string with up to 16 characters)
36h..3Fh	Reserved for further use

Table 10: Command Table Entry Data Block Format

For reading a CT entry, the start reading entry with ID command returns the data block size in bytes (40h). After this command a read command can be repeated until the whole data block is read out.

For writing a CT entry, start with a command which defines the ID and the data block size, after this repeat writing the data with a command until the whole data is written. If this is done correctly the bit in the presence list will be cleared.

For modifying a single motion command parameter during runtime, a motion command exists, with ID, write offset and data as parameters.

7 PVL Data Format

The parameter settings, curve profiles and command table of a LinMot Controller can be exported together with other data (parameters, oscilloscope settings) as a configuration file (ending *.lmc) with the LinMot-Talk software. The configuration file can then be imported to other controllers again under usage of the LinMot-Talk 4 software.

Some users of the LinMot servo systems want to store the parameter setup and/or curves in their main machine controller (PLC, IPC) as a simple value list. So they don't need any PC tools for configuration of servo controllers when they produce series of the same machine type or when they have to replace a controller in the plant.

Beside the possibility of exporting configurations in LMC file format the LinMot-Talk software allows to save the setup into a simple text file. Data is stored in an easy interpretable **comma separated value format**. The file has the extension *.pvl (Parameter Value List). A simple parser can convert the text file data into the customer specific data format.

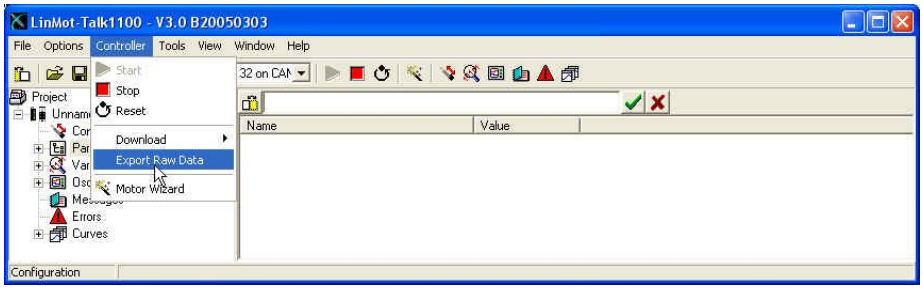

Figure 12: Export Raw Data

The LinMot-Talk software offers selective export of configuration data:

If 'All Parameters' is selected, all parameters are listed in the export file (hundreds of parameters, most of them are not used or set to their default value)

If 'Changed Parameters' is selected, the resulting file will be much smaller. Only parameters, which have been changed during the setup process (and therefore are relevant for the application), are listed in the file.

Only curves that are selected will be stored to the file.

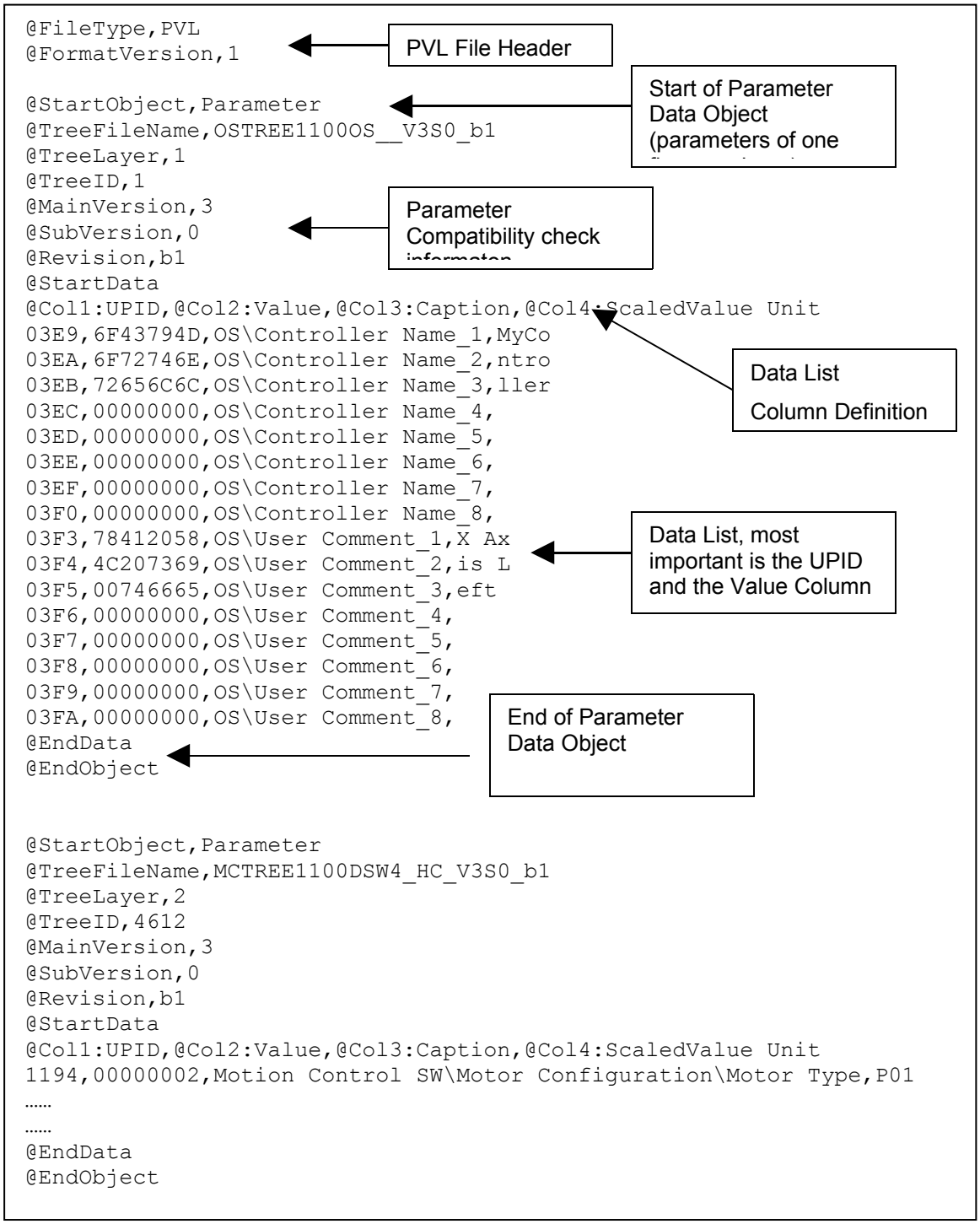

Figure 13: Example of a PVL data file with Parameter Data Objects

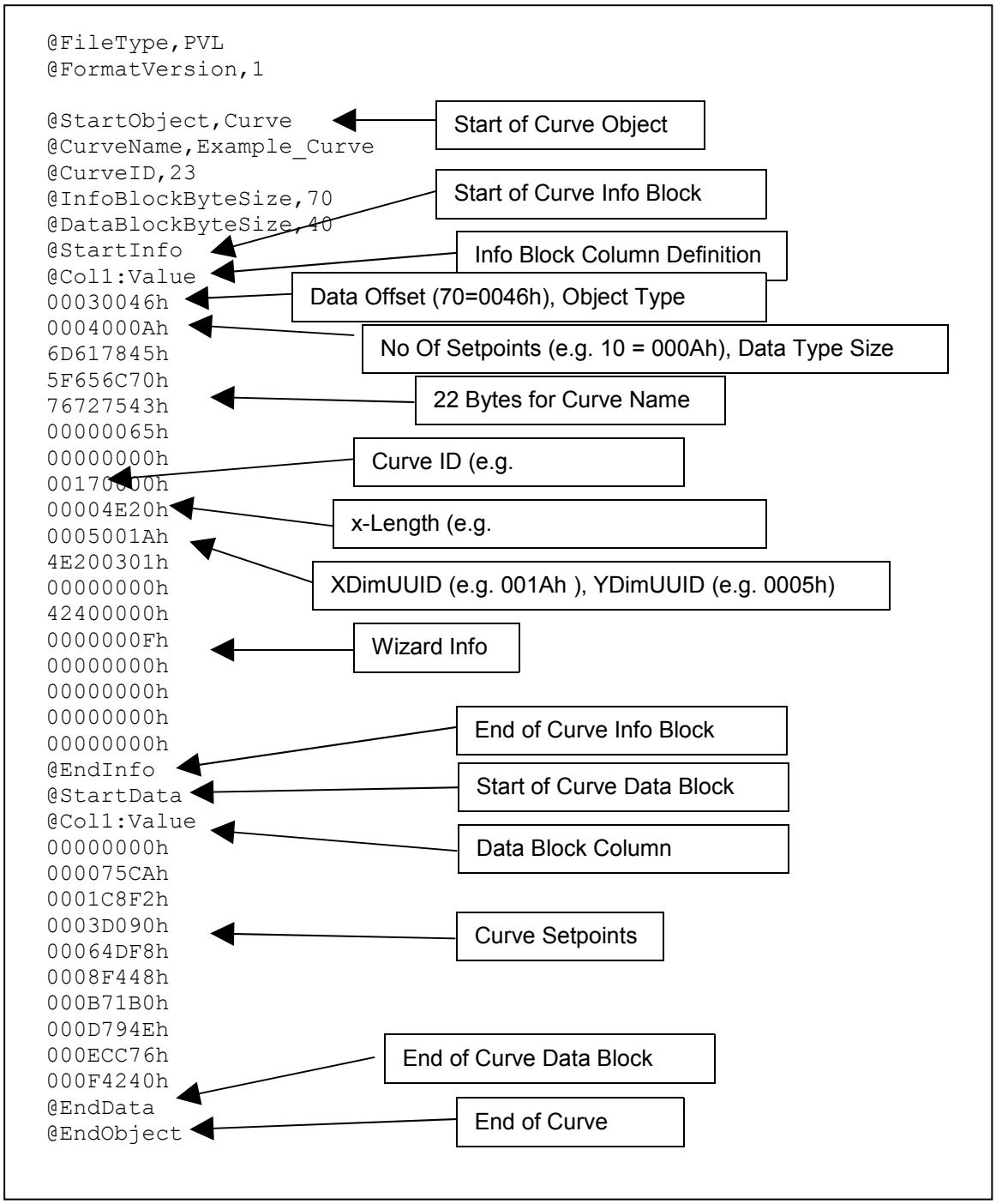

Figure 14: Example of a PVL data file with a Curve Object

8 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…