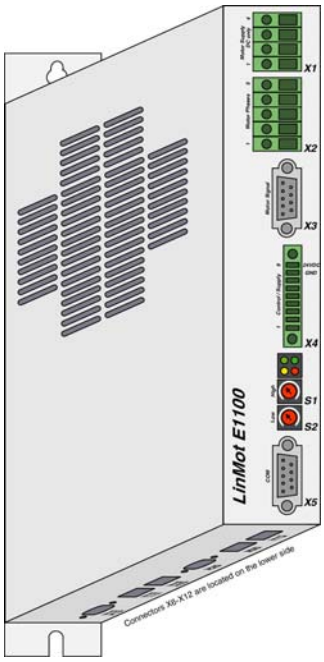
LinMot®



User's Guide

LinMot E1100 Servo Controller Configuration over Fieldbus Interfaces

Note

The information in this documentation reflects the stage of development at the time of press and is therefore without obligation.

NTI AG reserves itself the right to make changes at any time and without notice to reflect further technical advance or product

improvement. Please refer to the latest edition of our "General business terms"

Document version 0v0v4/ as, June 2006

1 Introduction

Users of LinMot E1100 Servo Controllers can easily setup their drive by using the LinMot Talk1100 software. Beside other functionality (firmware download, monitoring, PLC emulation, etc.), the LinMot Talk1100 software is used for altering the firmware configuration parameters and creating, up- and downloading curve profiles.

Most of the LinMot E1100 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 E1100 Servo Controller uses a fieldbus connection for the communication to the superior control system (PLC, IPC), then 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 the data access is implemented in the respective communication interfaces can be found in the corresponding interface user manuals.

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 Talk1100 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.

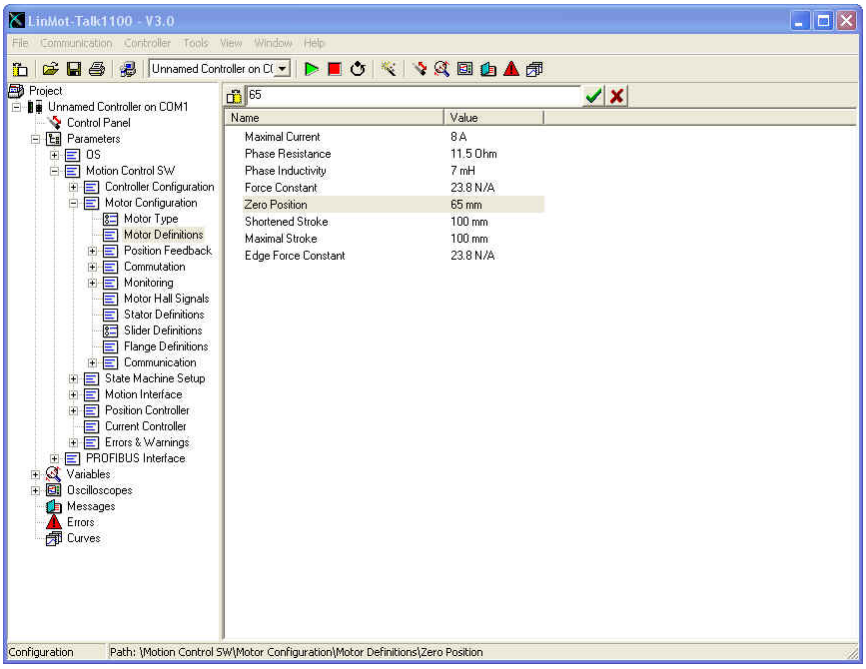


Figure 1: Firmware parameters listed in a tree structure

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. The UPID (= Unique Parameter ID) itself is a 16 Bit integer number.

Both, UPID and Raw Data value of any parameter, can be displayed in the LinMot Talk1100 parameter tree structur (press Show/Hide Details button).

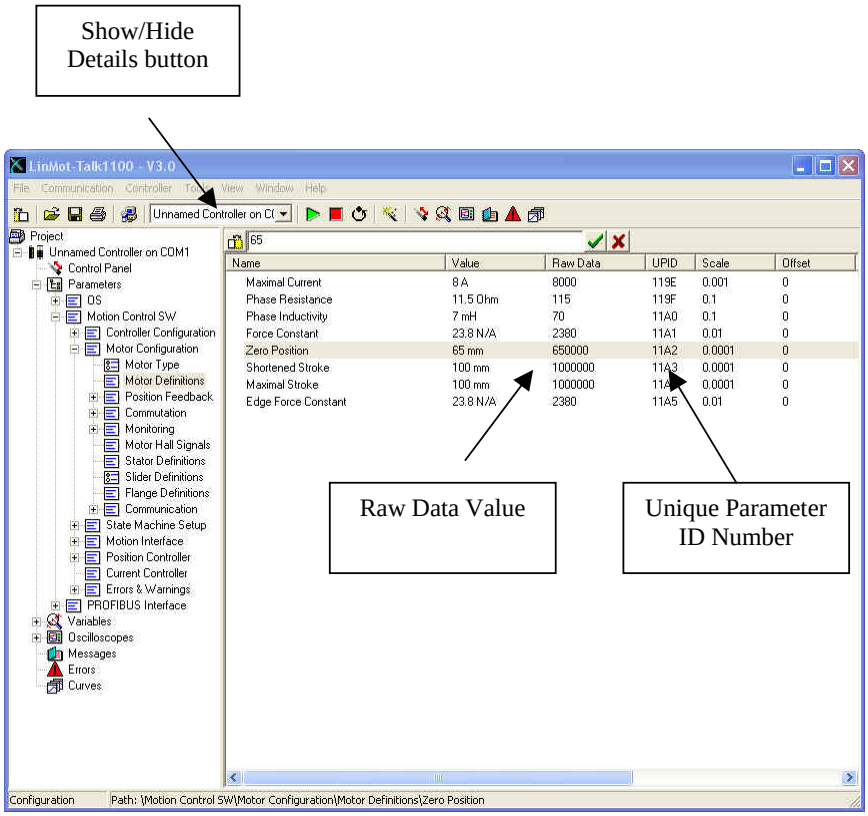


Figure 2: Detailed view for parameters

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 power-up of the controller for each parameter there are two values stored in the controller: the ROM and the RAM value.

The firmware uses exclusively the RAM values of the parameters in its control tasks at run time (fast data access). The controller's operating system and the fieldbus interfaces provide independent access to ROM and RAM value.

When parameters are altered using the LinMot Talk1100 software, the ROM value and/or the RAM values are affected:

- When 'live' parameters are changed, then the LinMot Talk1100 software writes to ROM and RAM memory.

- The changed value of 'non-Live' parameters are written to ROM exclusively (copied to RAM at next firmware start).

The LinMot Talk1100 software reads and displays the ROM value. It reads the parameter values only once (during the login process). The PC software allows altering non-live parameter only if the firmware is stopped.

Over the fieldbus interface the RAM value of 'non-live' parameters can not 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 directly influences the system behavior when the firmware is running (e.g. control parameters of the position control loop).

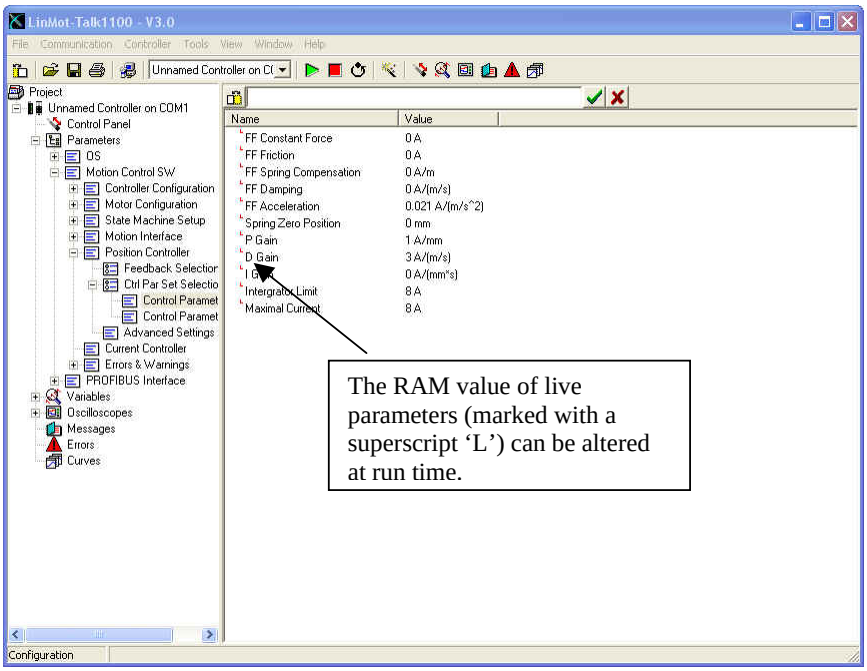


Figure 3: Live Parameters marked with a superscript 'L'

Changes on ROM values don't affect the system behavior until the next boot cycle of the firmware (e.g. after a software reset initiated by the superior control system).

2.4 Default Value

Beside the actual RAM and ROM value each parameter has its default value. The default parameter value is displayed in the detailed parameter view of the LinMot Talk1100 software.

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

The fieldbus interfaces offer the possibility to reset the parameter values to their default value. Resetting to default value is possible for single parameters (via UPID) or for all parameters of any firmware layer (see below) at once.

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 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 string in the general case. Strings are therefore handled in a special way:

- In the LinMot Talk1100 software to each string parameter one single UPID is shown.
- Internally the string is splitted into parts (so called 'stringlets') with the length of 4 characters each (= 4 bytes = 32 bits).
- Each stringlet has its own UPID.
- The UPID of the first stringlet of the whole string is the string UPID plus 1, the UPID of the second stringlet is the string UPID plus 2 and so on.

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

Example:

We want to write the string 'X-Axis Left' to the parameter 'User Comment' (UPID 1010):

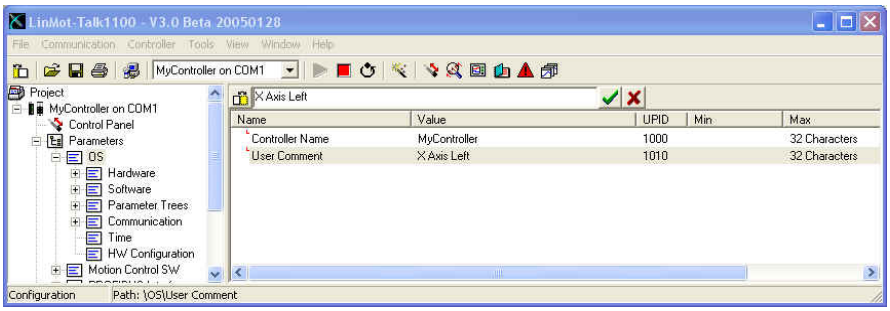


Figure 4: String parameter example

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

Stringlet	"X Ax"	"is L"	"eft"
Parameter UPID	1011	1012	1013
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: Raw data values for string parameter example

3 Firmware Layer Concept

The firmware on the E1100 servo controller consists of up to four layers:

Layer	Name	Layer Functionality
1	Operating System	- Resource Management - Communication with LinMot Talk1100 - Start/Stop of the other SW layers
2	Motion Control Software	- Current Control Loop - Position Control Loop - Set Value Generation - Monitoring
3	Interface Software	- Communication to superior control system (e.g. Profibus Interface, CANOpen Interface, DeviceNet, LinRS, etc.)
4	Application	- Custom specific firmware extensions

Table 2: The four firmware layers

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

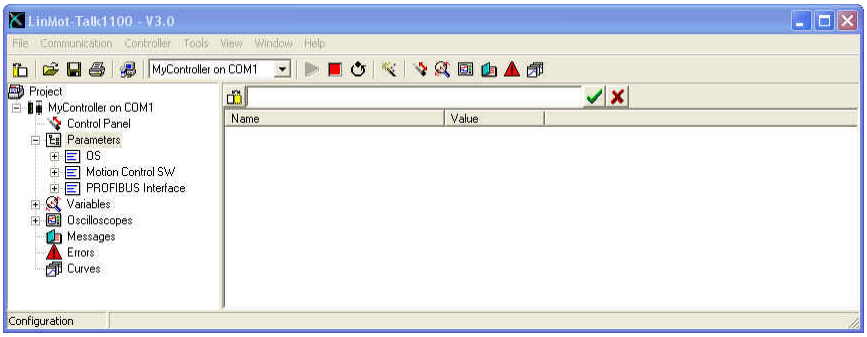


Figure 5: Parameter tree with branches for the different firmware layers

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

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

The parameter definitions (UPID, default value, min/max value, memory location and access rights) are stored in the controller. 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.

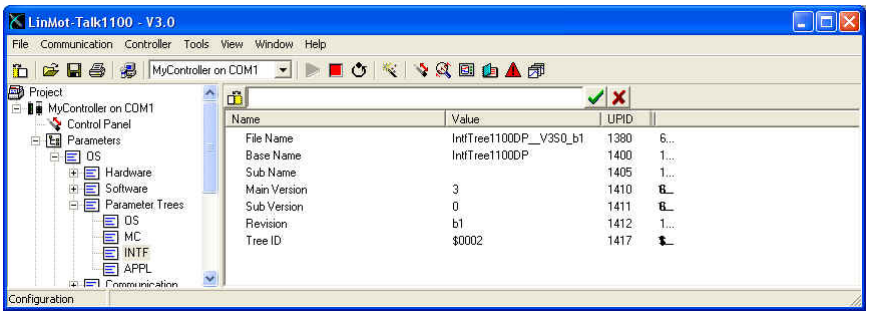


Figure 6: Parameter tree information

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	Main Version UPID	Sub Version UPID
1 (OS)	1337	1330	1331
2 (MC SW)	1377	1370	1371
3 (Intf. SW)	1417	1410	1411
4 (Appl.)	1457	1450	1451

Table 4: UPID numbers for parameter tree information

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.

Beside the parameter 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 tests. With these tests it can be ensured, that the configuration stored in the superior control system (source file) is fully compatible with the firmware installed on the connected LinMot 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 tree in the source file must be the same or smaller than the corresponding value of the target system (backward compatibility within main version).

Table 5: Parameter file compatiblity rules

5 Curves

The LinMot Controller can store up to 100 curves, which can be generated and downloaded by the LinMot-Talk1100 Software. The curves are identified by their ID (1..100).

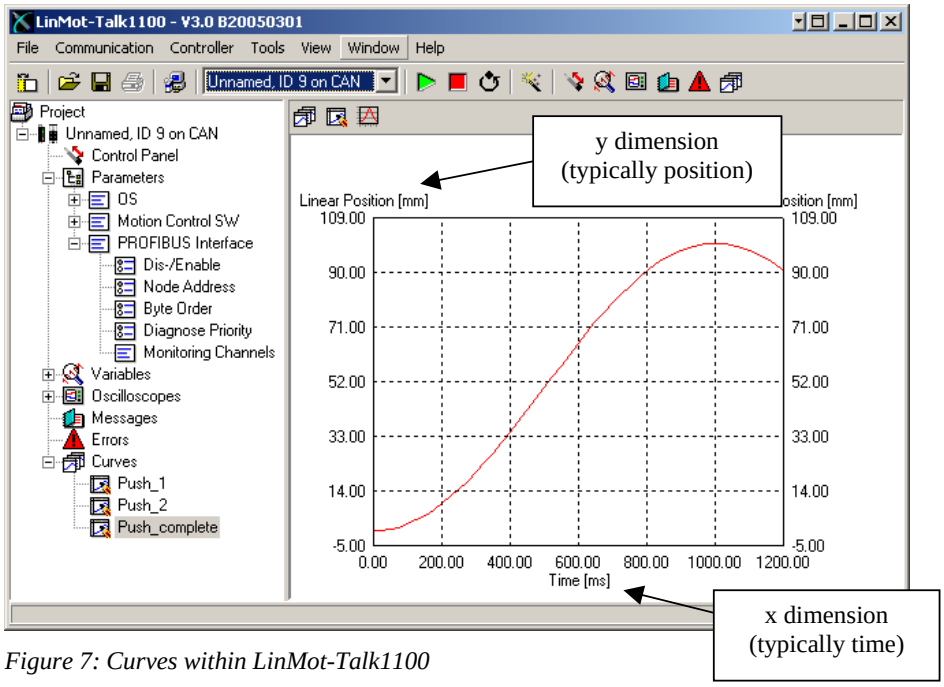


Figure 7: Curves within LinMot-Talk1100

The curves that can be accessed by the motion control firmware through corresponding motion commands are located in the RAM memory space of the controller. They can permanentely 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.

If the curve profiles are generated and downloaded using the LinMot Talk1100 software, then they are always stored to RAM and flash memory (permanetely saved).

It is also possible to download curves to the controller over the fieldbus. It is then 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 LinTalk1100. 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 Talk1100 software. The file then can be loaded to the main controller.

3. The curve object is fully generated by the main machine controller without using the LinTalk1100 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 those blocks. For further details how the Curve Service is implemented in the various fieldbus interfaces please consult the corresponding interface manual.

Curve Info Block		
Byte Off	Type	Name
0..1	UInt16	Data offset
2..3	UInt16	Object type
4..5	UInt16	No 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 comprehend 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 (=half bytes):

- 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: Ecoder Pos (Increments) 4: Ecoder Speed 5: MicroSteps				**X dimension code:** 0: Time 1: Encoder Pos (Increments) 2: Position			

Table 7 Curve Info Block: Object Type

According to the above given definition 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.

No 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 byte 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.), then the name space is typically filled with 00h.

Curve ID

The *Curve ID* must be unique. Allowed value 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 Defintions

According to the above given table 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 LinTalk1100 software. All those bytes should be 00h for curves which are not generated with the LinTalk1100 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 Uploading Curves from the Controller

To read a curve from the controller using the fieldbus inteface, 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 descriped in the corresponding interface manuals.

The reading of a curve always is according the following scheme:

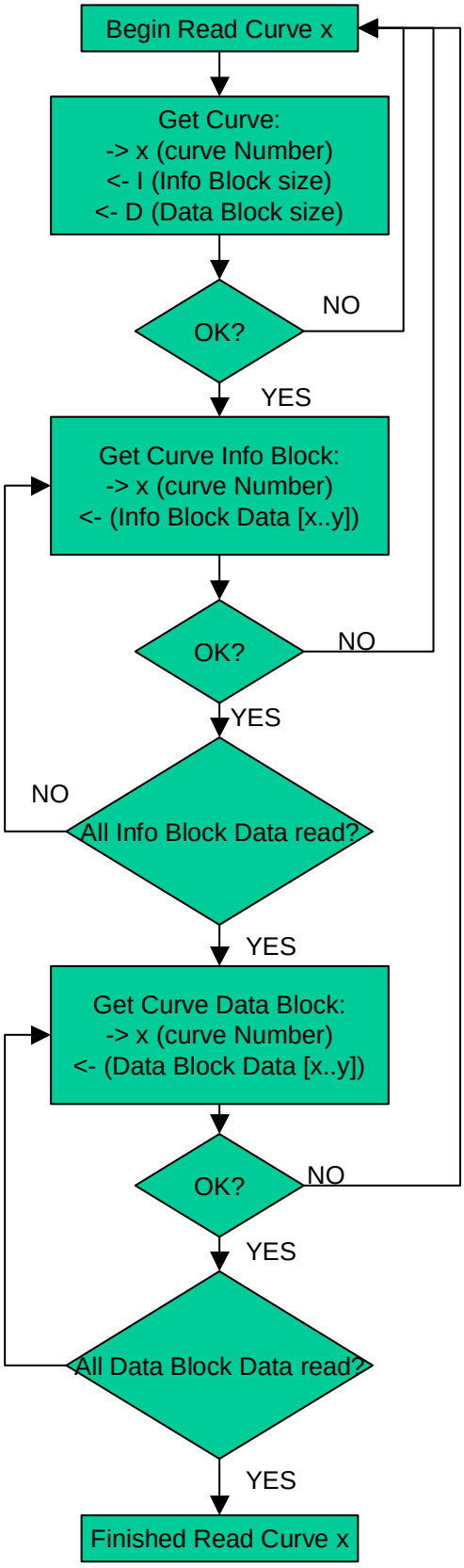


Figure 8: Flowchart Read Curve

5.5 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 the following scheme:

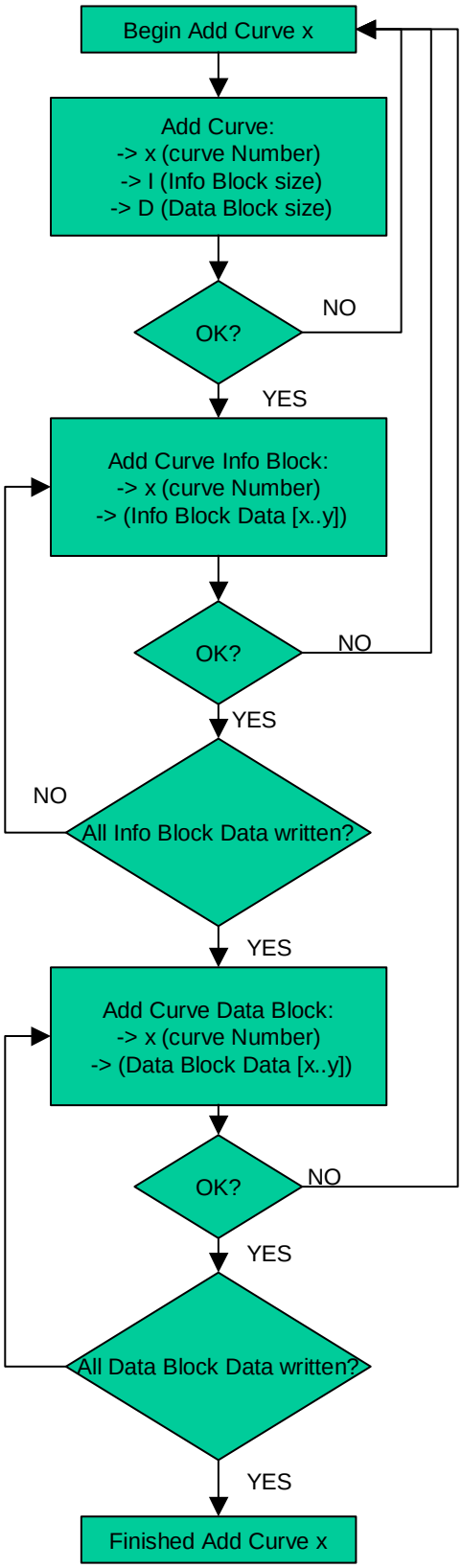


Figure 9: Flowchart Add Curve

5.6 Erasing all curves in the RAM of the Controller

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

-Curve Service: Delete all Curves (RAM)

6 Command Table

For programming simple decision sequences the LinMot Servo controller supports the Command Table programming utility. Up to 255 Command table entries can be programmed. The entries can be arranged in sequencies and some branch possibilities are offered.

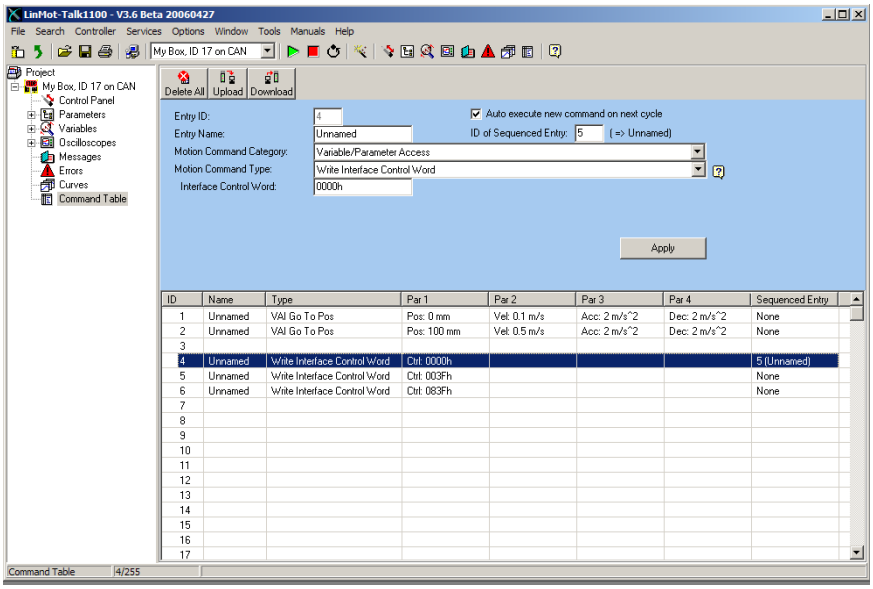


Figure 10: Command Table within LinMot-Talk1100

The Command Table 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 command table can be permanentely stored into a flash memory block as well. At boot time the operating system copies the Command Table data from the flash memory to the RAM memory block.

If the Command Table entries are generated and downloaded using the LinMot Talk1100 software, then they are always stored to RAM and flash memory (permanetely saved).

It is also possible to download or modify Command Table entries in the controller over the fieldbus. It is then necessary to have the Command Table entry data stored in the superior machine controller. There are two ways to bring the entry data there:

1. The Command Table entries are created and downloaded to a controller by using LinMot-Talk1100. Afterwards the Command Table entry data can be uploaded from the controller over the fieldbus interface (using the Command Table service).

2. The Command Table entries are created and then exported to a raw data file (PVL file) using the LinMot Talk1100 software. The file then can be loaded to the main controller.

3. The Command Table entry data is fully generated by the main machine controller without using the LinTalk1100 command table editor.

6.1 Command Table Entry Format

Each Command Table Entry consists of a Bit in a Presence List (bit = 0 entry exists) and a Command Table Entry Data Block. The Command Table Services provides commands for reading/writing those blocks. For further details how the Curve Services could be used for reading and writing Command Table entries with the different fieldbus interfaces 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 Command Table Entry, a start reading entry with ID command exists which returns the data block size in bytes (40h). After this command a read command can be repeated until the whole data is read out.

For writing a Command Table Entry Start with a command which defines the ID and the size data block, 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 is cleared.

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

7 PVL Data Format

The parameter setting and the curve profiles of a LinMot Controller can be exported together with other data (variables, oscilloscope settings) as a configuration file (ending *.lmc) with the LinMot Talk1100 software. The configuration file can then be imported to other controllers again under usage of the LinMot Talk1100 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 to export configurations in LMC file format the LinMot Talk1100 software allows to save the setup into a simple text file. The data are stored in an easy to interpret **comma separated value format**. The file has the ending *.pvl (Parameter Value List). A simple parser tool can convert the text file data into the customer specific data format.

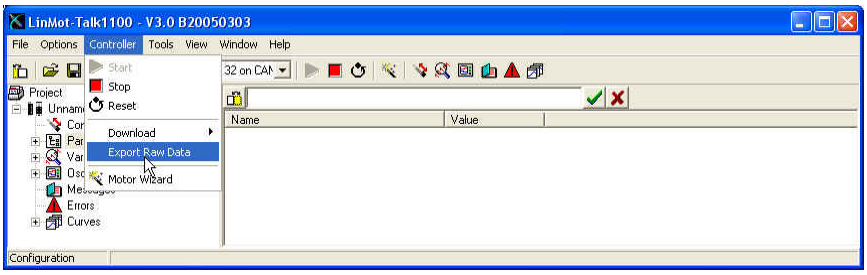


Figure 11Export Raw Data

The LinMot Talk1100 software offers selective export of configuration data:

- if 'All Parameters' is selected, then all parameters are listed in the export file (hundreds of parameters, most of them are not used)
- if 'Changed Parameters' is selected, then the resulting file will be much smaller. Only parameters that were changed during the commissioning process (and therefore are relevant for the application) are listed in the file.
- Only curves that are selected will be stored to the file.

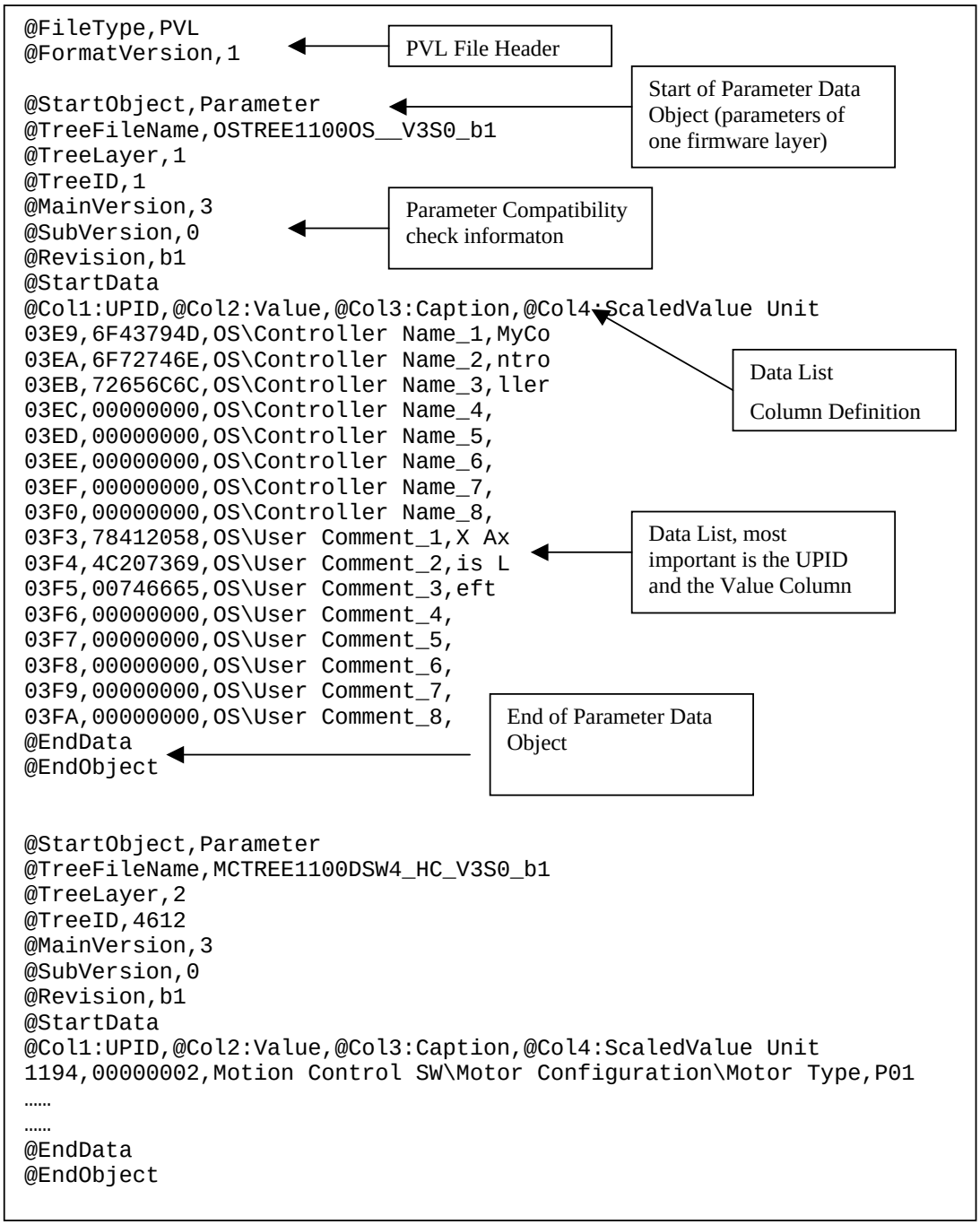


Figure 12: Example of a PVL data file with Parameter Data Objects

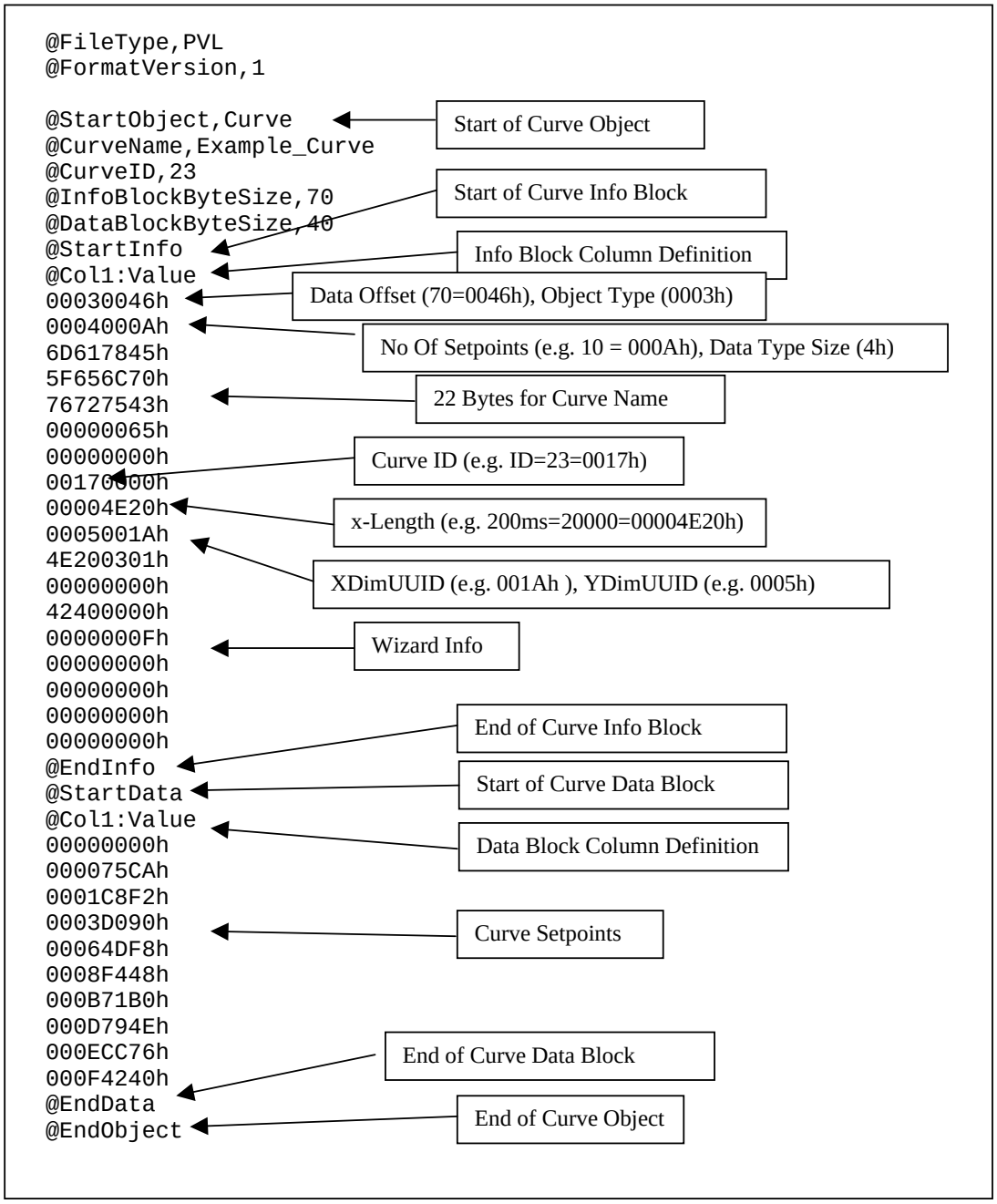


Figure 13: Example of a PVL data file with a Curve Object

Smart solutions are…

NTI AG
LinMot
Haerdlistr. 15
CH-8957 Spreitenbach
Schweiz

Tel.: +41 (0)56 419 91 91
Fax: +41 (0)56 419 91 92
office@LinMot.com
www.LinMot.com

LinMot Inc.
N2444 Broad Street
Delavan
WI 53115
USA

Phone: +1-877-546-3270
+1-262-728-22699
Fax: +1-800-463-8708
officeUS@LinMot.com
www.LinMot.com

***LinMot*®** products are available from more than 80 Distributors worldwide. For the distribution nearest you, visit http://www.LinMot.com