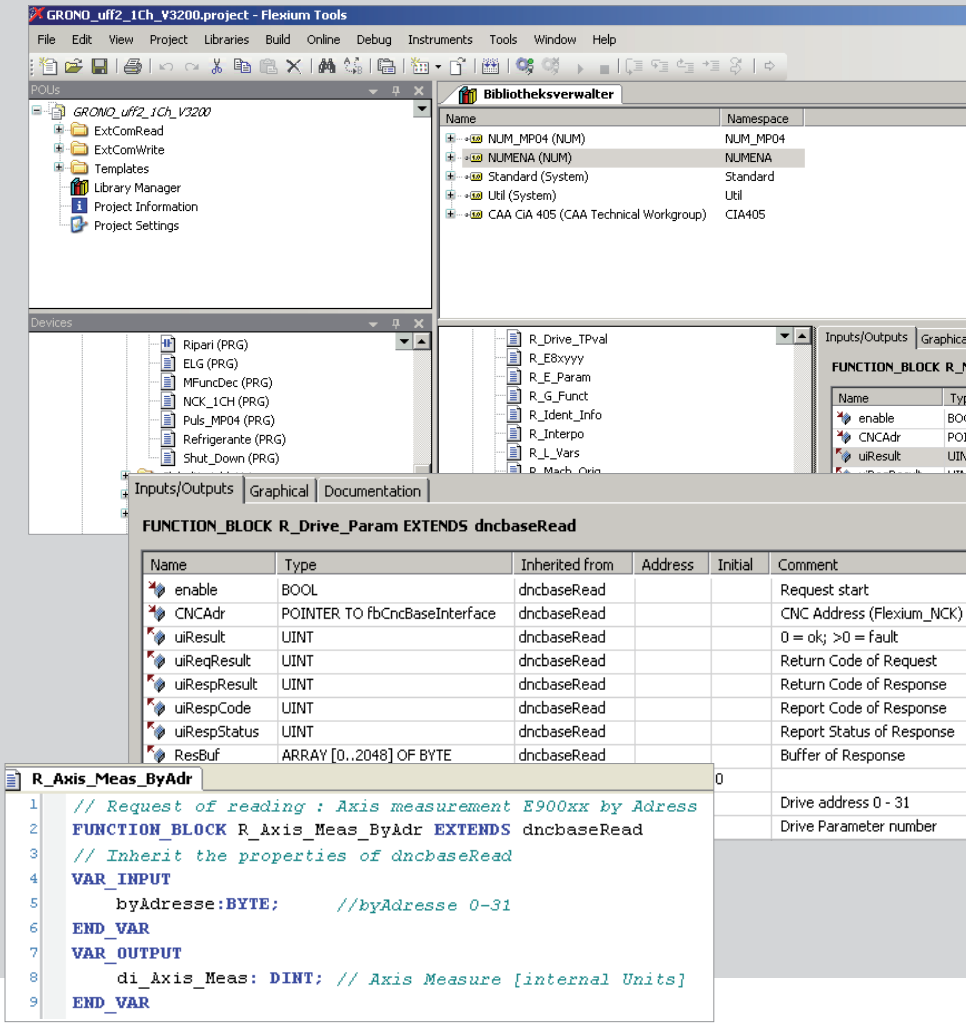
Flexium and Flexium+

Extended NCK Access

Programming manual

M00026EN-02

Edition 11.2016



www.num.com

CNC HighEnd Applications

Despite the care taken in the preparation of this document, NUM cannot guarantee the accuracy of the information it contains and cannot be held responsible for any errors therein, nor for any damage which might result from the use or application of the document.

The physical, technical and functional characteristics of the hardware and software products and the services described in this document are subject to modification and cannot under any circumstances be regarded as contractual.

The programming examples described in this manual are intended for guidance only. They must be specially adapted before they can be used in programs with an industrial application, according to the automated system used and the safety levels required.

© Copyright NUM 2011
All rights reserved. No part of this manual may be copied or reproduced in any form or by any means whatsoever, including photographic or magnetic processes. The transcription on an electronic machine of all or part of the contents is forbidden.

© Copyright NUM 2011 software
This software is the property of NUM. Each memorized copy of this software sold confers upon the purchaser a non-exclusive licence strictly limited to the use of the said copy. No copy or other form of duplication of this product is authorized.

Summary

Page deliberately empty.

Summary

Page deliberately empty.

1 Foreword

1.1 Terminology

ENA stands for Extended NCK Access.
It extends the previous DNC1000 functionality allowing exchanges between PLC and NCK of numerous data which are not part of the standard exchange area.

It is provided as a library integrated into Flexium Tools.

1.2 Purpose of the ENA library

The aim of this built-in library is to provide an easy access to all functions without having to deal with request formatting.

This library is dedicated to the Flexium PLC and is managed via Flexium Tools.

This library is an option of the Flexium system:

- Option code: *FXSW282124 Extended NCK access*

- *# PLC licences*
- *# FXSW282124 Extended NCK access*

1.3 Use of the manual

This manual is intended to provide the user with all the information required to use the ENA library in a Flexium application.

Previous knowledge of PLC programming with Flexium Tools is required, in particular integration and use of a library.

1.4 Document revisions

Date	Index	Reason for revision
05 - 2011	00	Document creation conform with Flexium Tools version 3.3.0.0
12 - 2012	01	- Chapter 6.3 Read NUMDrive test points: new procedure and user example. - Deleted section Appendix 4.
11 - 2016	02	- Introduced separated Read requests / Write requests informations for Flexium and Flexium+. - Conform with Flexium Flexium system 4.0.30.

1.5 Use of symbology in the manual

Symbol	Meaning of the symbol
	Important notes on product use.
	You may find this symbol whenever a wrong operation may damage the product.

1.6 NUM Agencies

The list of NUM agencies is given on http://www.num.com

Page deliberately empty.

Summary

Page deliberately empty.

2 General

2.1 Basic functions

All ENA functions are defined in a PLC library. This library is not installed as standard and must therefore be included in your project should you need any of these functions.

An ENA action is done in three steps:
- Posting a request to the CNC
- Reading the answer when it becomes available
- Clearing the request.

Those steps are handled automatically by each function; there is no need to program them.
As a general rule, an ENA function should be programmed in a specific task (50 to 100 ms).

An instance of the function is invoked and enabled during a particular scan and the answer checked at the next scan. If the answer is positive and without error, the function has been executed and the result is available; as soon as the function is posted the answer code is 0x99 (wait) meaning that the request is correctly taken into account.

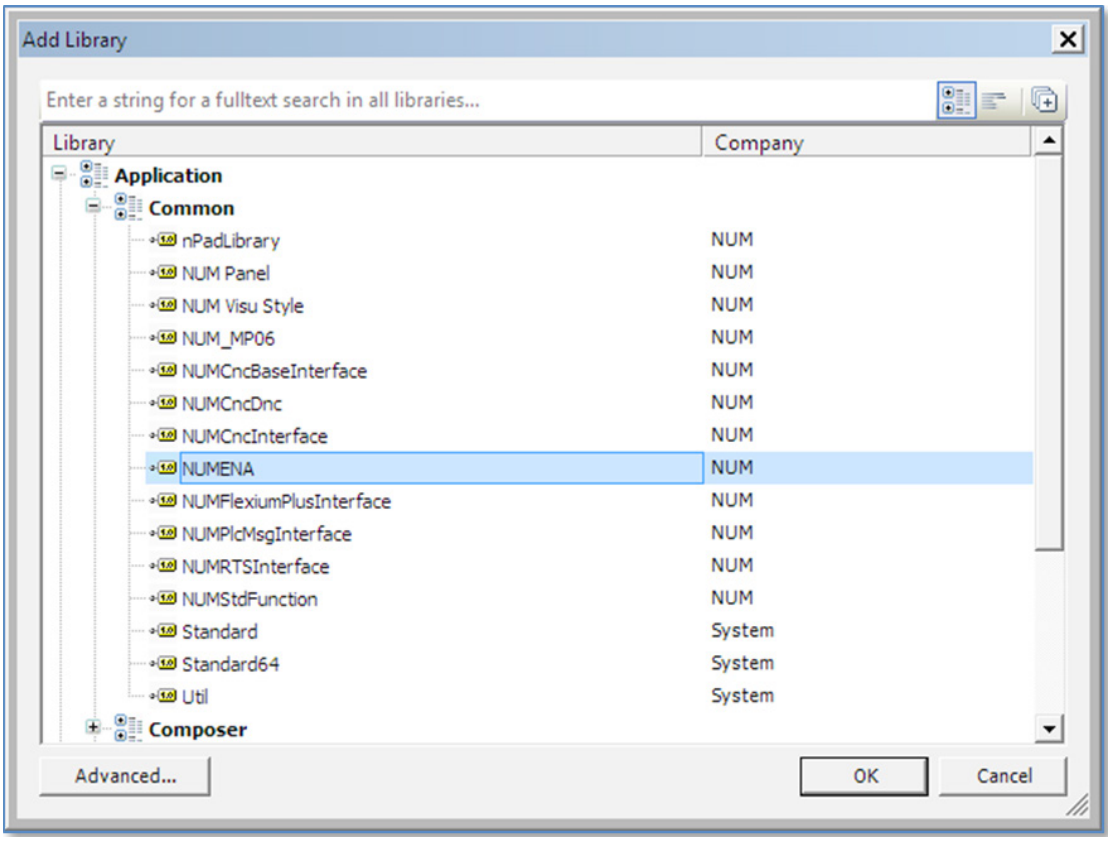


Figure 1 : ENA Library

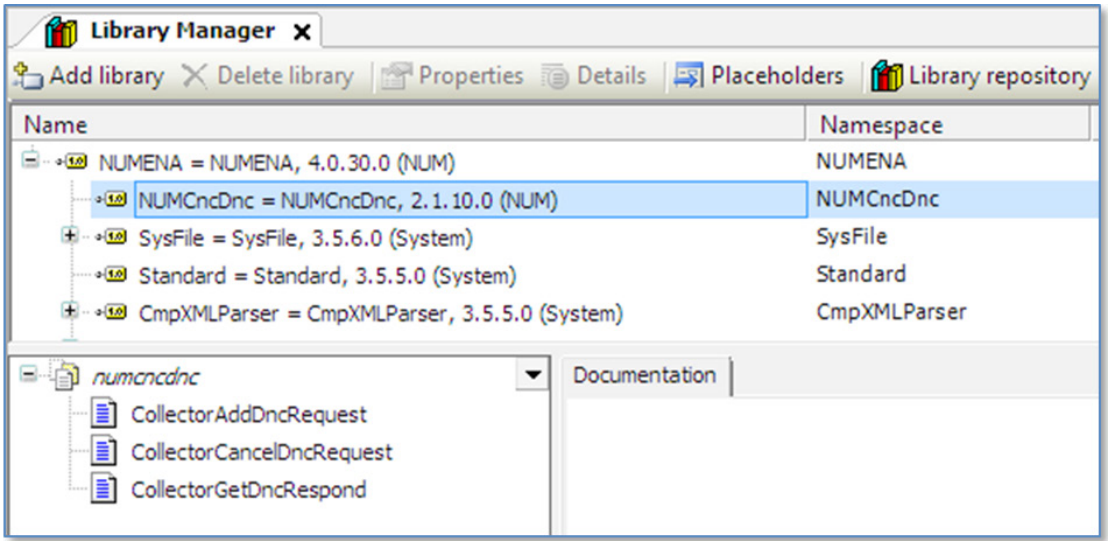


Figure 2 : The three steps of a function (for information)

2.2 Format of the data returned

Type	Lower limit	Upper limit	Size	Prefix	Remark
BOOL	FALSE	TRUE	1 (8) bits	bo	
BYTE	0	255	8 bits	by	
WORD	0	65535	16 bits	wo	
DWORD	0	4294967295	32 bits	dw	
LWORD	0	2^{64}	64 bits	lw	
SINT	-128	127	8 bits	si	
USINT	0	255	8 bits	us	
INT	-32768	32767	16 bits	in	
UINT	0	65535	16 bits	ui	
DINT	-2147483648	2147483647	32 bits	di	
LINT	-2^{63}	$2^{63}-1$	64 bits	li	
UDINT	0	4294967295	32 bits	ud	
ULINT	0	2^{64}	64 bits	ul	
REAL	$\pm10^{-37}$	$\pm10^{+38}$	32 bits	re	
LREAL	$\pm10^{-307}$	$\pm10^{+308}$	64 bits	lr	
STRING	1	255	bytes	st	String
TIME	T#0	T#12h34m15s10ms	32 bits	ti	
TOD	TOD#0	TOD#23:59:59	32 bits	to	Time_of_Day
DATE	DATE#1970-01-01	DATE#1996-05-06	32 bits	da	
DT	DT#1970-01-01-00:00:00	DT#1996-05-06-15:36:30	32 bits	dt	Date and Time

2.3 Error Codes

These codes are returned when invoking the function. Any value different of 0 (answer ready) or 0x99 (answer pending) must be managed.

The error code `uiResult` allows for checking if any error is present,

```
IF uiResult = 0 THEN…
```

uiResult is made up of four return values:

```
uiResult:= iReqResult + (irespResult * 10) + (Resbuf [2] * 100) + (Resbuf [4] * 1000);
```

If uiResult is not zero, the following variables can be used to find the type of error.

uiReqResult		Response on request sending	CollectorAddDncRequest
uiRespResult		Response on data return	CollectorGetDncRespond
uiRespCode	Resbuf [2]	Response Code	Global answer
uiRespStatus	Resbuf [4]	Status code	Set of answer

iReqResult	Error name	Error	Remark
0	ERROK	Transmission OK	
1	BUFFERFULL	Buffer full	Resend the request
3	BUFFERERR	Buffer error	Message + fix application
4	NCKNOTREADY	CNC not ready	Message + resend request
6	UNKNOWNCKADRESS		Error + fix application
9		No answer	Wait
50		No license	Activate ENA option

iRespResult	Error name	Error	Remark
0	ERROK	Read OK	
2	RESPONDNOTREADY	Answer not yet ready	Read again the answer
4	NCKNOTREADY	CNC not ready	Message + resend request
5	REQUESTNUMBERNOTREADY	Request not ready	Cancel and resend the request
6		No message at this port number	
7		Buffer too small for the answer	
8		Invalid Line Number	
9		No answer	Wait

iRespCode	Error name	Error	Remark
0	ERROK	Answer OK	
1		Answer OK but partial	
2		Variable access conflict	
3		Bad request parameter	
9		No answer	Wait

iRespStatus	Error name	Error	Remark
0	ERROK	Transmission OK	
1		Protected access	
2		Non existing element	
3		Limit error	
4		Currently meaningless	
9		No answer	Wait

Page deliberately empty.

Page deliberately empty.

Summary

Page deliberately empty.

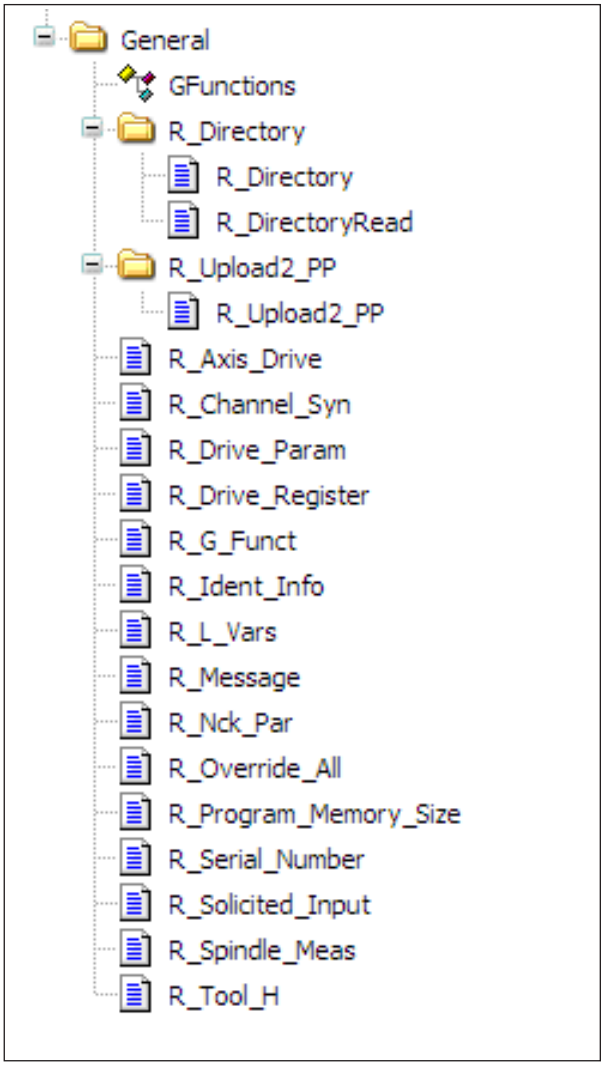
3 List of requests

The library contains Read as well as Write requests grouped in three categories:
- Common requests
- Flexium specific requests
- Flexium+ specific requests.

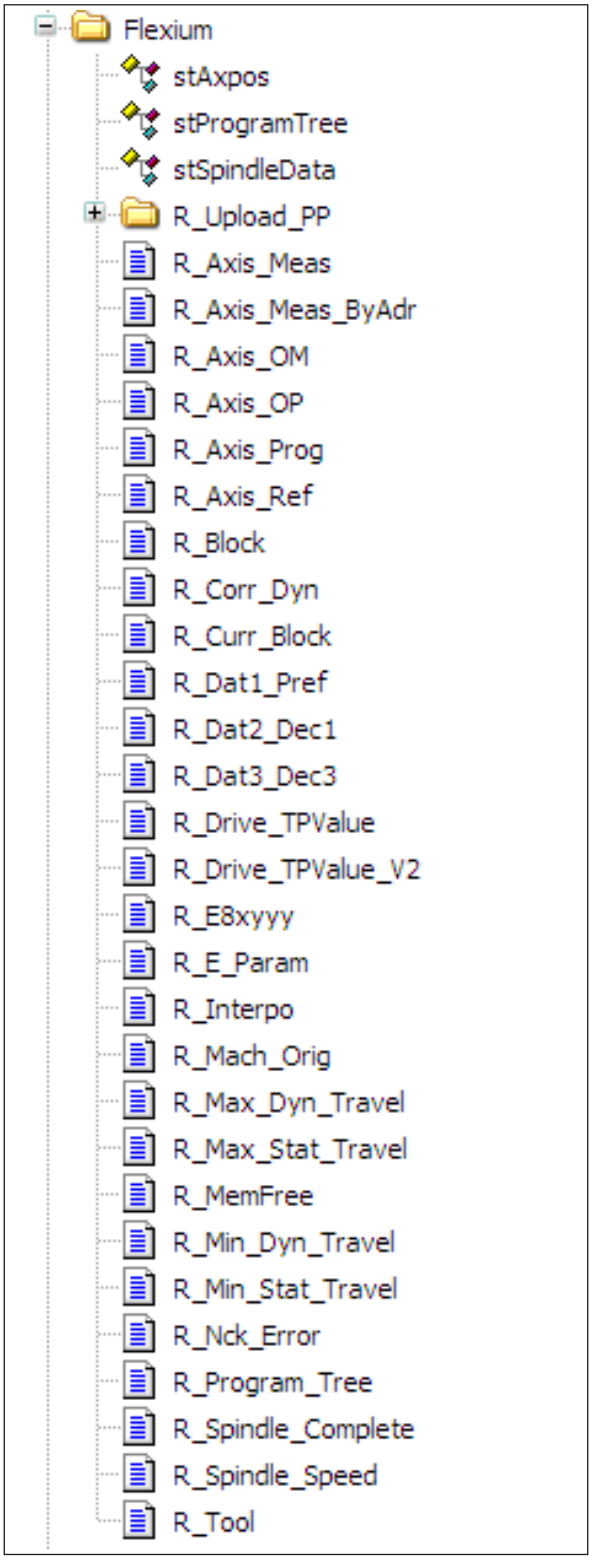
In addition several data structure are defined.

3.1 Read requests

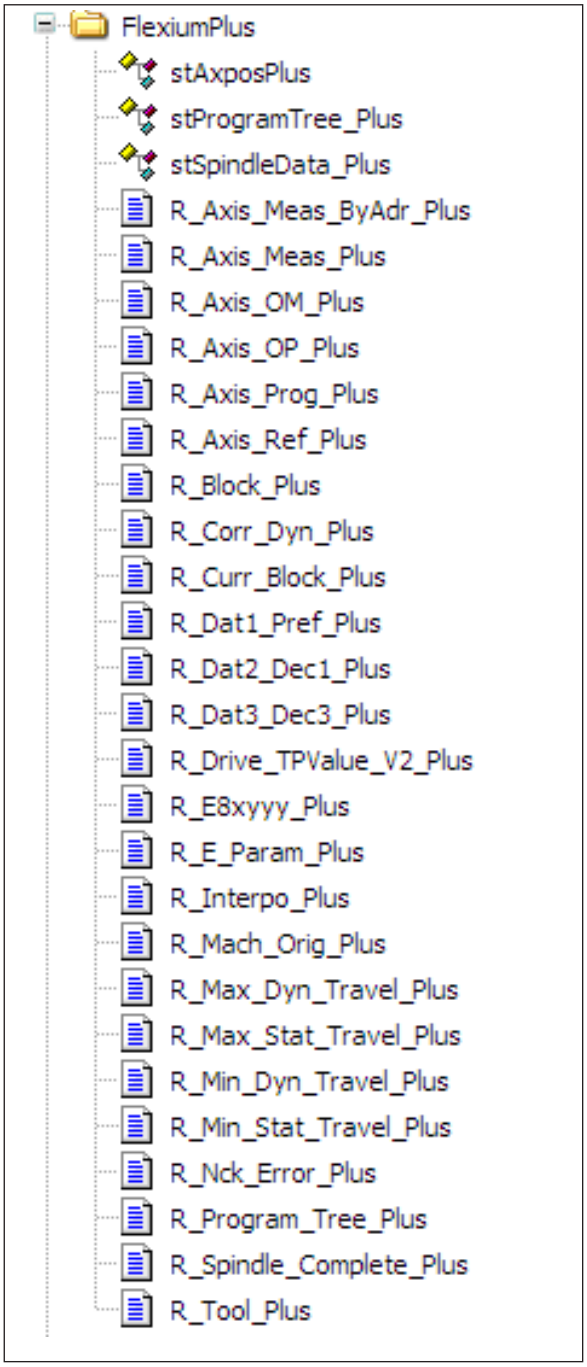
3.1.1 Common read requests



3.1.2 Flexium specific read requests

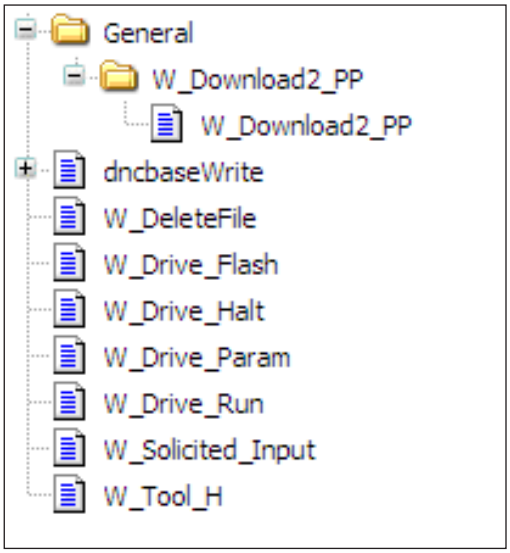


3.1.3 Flexium+ specific read requests

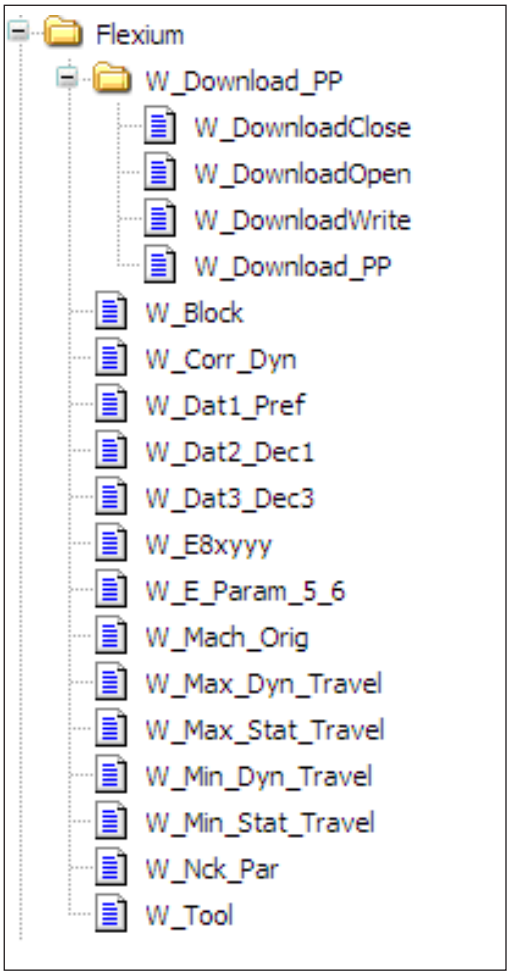


3.2 Write requests

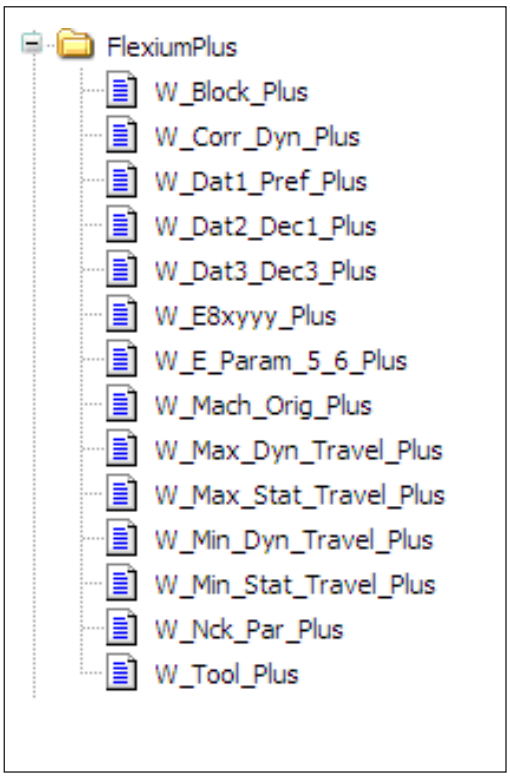
3.2.1 Common write requests



3.2.2 Flexium specific write requests



3.2.3 Flexium+ specific write requests



Page deliberately empty.

Summary

Page deliberately empty.

4 Programming

The use of ENA is quite simple as all request are build on the same model.

This is generally done in the following steps:

- A specific task should be created for all ENA requests:
  It is in most cases not necessary to have a period lower than 50ms and this task priority should be lower than the PLC_PRG main PLC tasks.
- In a POU of this task, declare an instance of the request to execute.
- Fill in the required parameters. Pay particular attention to the CNCAddress to avoid any exception.
- Use the enable flag to launch the request either one shot or permanently according to the context.
- Check the return code; handle the errors if they are some or manage the answer when this code is 0. Once the return code is 0 the request is terminated and the enable can be cleared.

P.S. Some request (super requests) require several cycles to be executed.
These requests are relative to the part program memory management. The number of cycles can vary according to the size of data transmitted. Explanations will be given when studying those super requests.

The following examples are given only to demonstrate the function. They are simplified on purpose and in particular the errors are not handled. In a real application it will be necessary to manage the possible errors in order to prevent a malfunction.

Return codes

The different possibility of return codes are given at chapter 2.3.
In the following example the codes handled will be:

- 0x99 : Answer pending
- 0x00 : Answer available or no request.

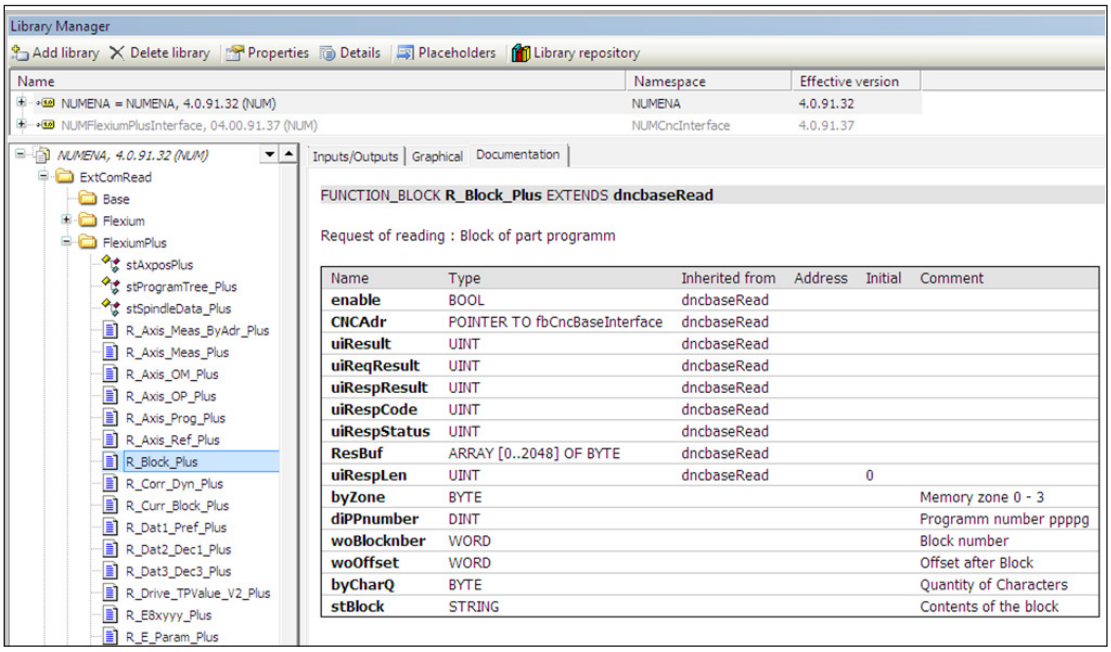
4.1 Program structure

The following requests were all launched one by one thanks to a specific task of 60ms period and a priority of 10. This task contains nothing else than the request to check.

In those examples, the flag *xEnable* is a Boolean which is SET by an action. The clearing of this flag is done in the example itself. As already stated the return code is not checked against an error.

This check must be made for real cases to prevent any lock-up or unwanted machine reaction.
In general the *VAR* definition part of these examples only declares the mandatory variables in order to be able to run the said example.
More variables may exist.

For the full list refer to the documentation tab of the request in the *Library Manager editor*.



Page deliberately empty.

Page deliberately empty.

Summary

5 Read requests

5.1 General requests

Those requests are available for Flexium as well as Flexium+

5.1.1 List of interpolated axes ► R_Axis_Drive.

This request returns a DINT value.
Each bit corresponds to an axis declared as servo controlled and interpolated.
(P120 N2 for Flexium+ or P3 for Flexium)

R_Gen_001

```
PROGRAM R_Gen_001
VAR

    oR_Axis_Drive               : R_Axis_Drive;
    answer                      : DINT;
```

```


oR_Axis_Drive.CNCAdr            := ADR(NCK);
oR_Axis_Drive(enable := xEnable);

IF oR_Axis_Drive.uiResult   = 0 THEN
    xEnable     := FALSE;
    answer      := oR_Axis_Drive.diDrive;
END_IF
```

5.1.2 Synchronization status in multi channels ► R_Channel_Syn

This request invoked for a particular channel returns the status of the part program synchronization in a multi channel system (P and Q parameters of the G78 function).

Please note that like for any channel relative request, the channels are numbered from 0 to 7 contrary to the part program index which is from 1 to 8.

R_Gen_002

```
PROGRAM R_Gen_002
VAR
    oR_Channel_Syn                  : R_Channel_Syn;
    P1                              : INT;
    Q2                              : INT;
END_VAR
```

```

oR_Channel_Syn.CNCAdr           := ADR(NCK);
oR_Channel_Syn.byChannel        := 0;
                            // Channels numbered 0 to 7 for the requests

oR_Channel_Syn(enable := xEnable);

IF oR_Channel_Syn.uiResult  = 0 THEN
    xEnable         := FALSE;
    P1              := oR_Channel_Syn.byPointEmit;
                            // Synchro point reached
    Q2              := oR_Channel_Syn.byPointCh2;
                            // synchro point expected from Channel 2
END_IF
```

5.1.3 Reading of the value of a drive parameter ► R_Drive_Param

This request returns a particular drive parameter value in internal units (*diPdriveUI*) and in one of the other types according to the parameter structure. (see Annex A2)

R_GEN_003

```
PROGRAM R_GEN_003
VAR
    oR_Drive_Param                  :R_Drive_Param;
    re_Answer                       : REAL;
    di_Answer                       : DINT;
    dw_Answer                       : DWORD;
    di_UI_Answer                    : DINT;
END_VAR
```

```

oR_Drive_Param.CNCAdr           := ADR(NCK);
oR_Drive_Param.byDriveAdd       := 1;           // Reading drive 1
oR_Drive_Param.woPnumber        := 246;         // Reading V246

oR_Drive_Param(enable := xEnable);              // Trigger the request

IF oR_Drive_Param.uiResult  = 0 THEN
    xEnable         := FALSE;
    re_Answer       := oR_Drive_Param.rePDrive;
    di_Answer       := oR_Drive_Param.diPDrive;
    dw_Answer       := oR_Drive_Param.dwPDrive;
    di_UI_Answer    := oR_Drive_Param.diPDriveUI;
END_IF
```

5.1.4 Reading of the content of a drive register ► R_Drive_Register

This request is generally reserved for drive advanced diagnostic.

For Drive Registers definition and structure please refer to the drive parameters manuals (M00031).

R_GEN_004

```
PROGRAM R_GEN_004

VAR
    oR_Drive_Register               : R_Drive_Register;
    wo_Answer                       : WORD;
END_VAR
```

```

oR_Drive_register.CNCAdr            := ADR(NCK);
oR_Drive_Register.byDriveAdd        := 1;
                                // Reading drive 1
oR_Drive_Register.woRegister        := 16#1AF1;
                                // Register address

oR_Drive_Register(enable := xEnable);
                                // Trigger the request

IF oR_Drive_Register.uiResult   = 0 THEN
    xEnable         := FALSE;
    wo_Answer       := oR_Drive_Register.woRegValue;
END_IF
```

5.1.5 List of G codes active in the channel ► R_G_Funct

This channel specific request returns the list of all currently active G codes as two lists of bits.

The structure of this answer is given by the table below.

R_GEN_005

```
PROGRAM R_GEN_005

VAR
    oR_G_Function                   : R_G_Funct;
    wo_Answer1                      : DINT;
    wo_Answer2                      : DINT;
END_VAR
```

```

oR_G_Function.CNCAdr            := ADR(NCK);
oR_G_Function.byChannel         := 0;
                            // Channels numbered 0 to 7

oR_G_Function(enable := xEnable);
                            // Trigger the request

IF oR_G_Function.uiResult   = 0 THEN
    xEnable         := FALSE;
    wo_Answer1      := oR_G_Function.diG_function0;
    wo_Answer2      := oR_G_Function.diG_function1;
END_IF
```

List of G codes per set

Index Set 1	Fn	Description
.0	**G94**	Feed rate expressed in millimetres, inches or degrees per minute
.1	**G95**	Feed rate expressed in millimetres or inches per revolution
.2	**G38**	Threading
.3	**G96**	Activate CSS
.4	**G97**	Cancel CSS
.5	**G92**	Tangential feed rate
.6	**G20**	Polar programming (X-C) lathe
.7	**G21**	Cartesian programming (X-Y) lathe
.8	**G40**	Cancel radius correction
.9	**G41**	Radius correction left
.10	**G42**	Radius correction right
.11	**G53**	Inhibit DAT1
.12	**G54**	Activate DAT1
.13	**G29**	3D tool correction
.14	*NUM Reserved*	
.15	**G93**	Feed in 1/T
.16	**G17**	Interpolation XY
.17	**G18**	Interpolation YZ
.18	**G19**	Interpolation XZ
.19	**G90**	Absolute programming
.20	**G91**	Incremental programming
.21	**G70**	Inches programming
.22	**G52**	Machine coordinates programming
.23	**G22**	Interpolation CZ
.24	**G0**	Rapid positioning
.25	**G01**	Linear interpolation
.26	**G02**	Interpolation CW
.27	**G03**	Interpolation CCW
.28	**G04**	Dwell
.29	*NUM Reserved*	
.30		
.31	**G09**	Precise stop

Index Set 2	Fn	Description
.0	**G10**	Activate interrupt stop
.1	**G16**	Tool axis selection
.2	**G51**	Mirror function
.3	**G45**	Pocket cycle
.4	**G12**	Overspeed by handwheel
.5	*NUM Reserved*	
.6		
.7	**G46**	Complex pocket milling
.8	**G26**	Activate RTCP
.9	**G73**	Cancel scaling
.10	**G74**	Activate scaling
.11	**G999**	Block concatenation
.12	*NUM Reserved*	
.13		
.14		
.15		
.16		
.17		
.18		
.19		
.20		
.21		
.22		
.23	**G24**	Activate inclined plane
.24	**G80**	Cancel machining cycles
.25	*NUM Reserved*	
.26		
.27		
.28	**G64**	Roughing cycles
.29	**G65**	Cycle de gorge
.30	*NUM Reserved*	
.31	**G07**	Inclined axes programming (grinders)

List of G codes per set

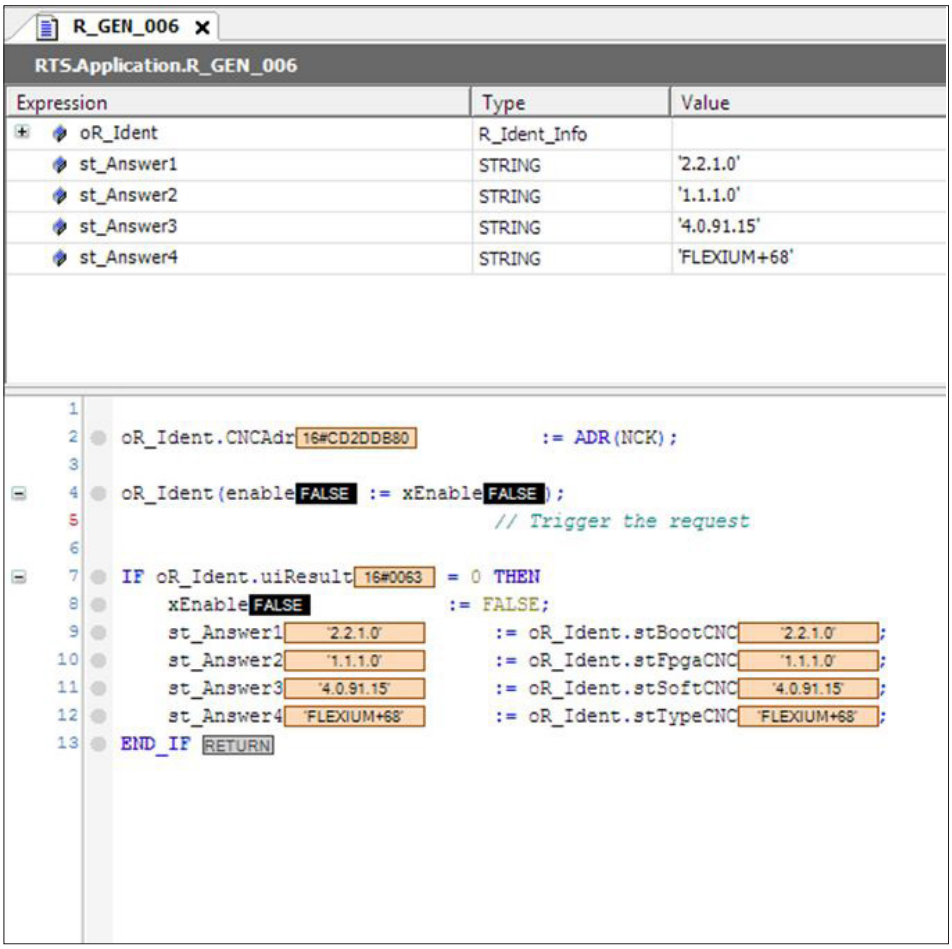
Fn	Description	Index	Set
G00	Rapid positioning	.24	1
G01	Linear interpolation	.25	1
G02	Interpolation CW	.26	1
G03	Interpolation CCW	.27	1
G04	Dwell	.28	1
G07	Inclined axes programming (grinders)	.31	2
G09	Precise stop	.31	1
G10	Activate interrupt stop	.0	2
G12	Overspeed by handwheel	.4	2
G16	Tool axis selection	.1	2
G17	Interpolation XY	.16	1
G18	Interpolation YZ	.17	1
G19	Interpolation XZ	.18	1
G20	Polar programming (X-C) lathe	.6	1
G21	Cartesian programming (X-Y) lathe	.7	1
G22	Interpolation CZ	.23	1
G24	Activate inclined plane	.23	2
G26	Activate RTCP	.8	2
G29	3D tool correction	.13	1
G38	Threading	.2	1
G40	Cancel radius correction	.8	1
G41	Radius correction left	.9	1
G42	Radius correction right	.10	1
G45	Pocket cycle	.3	2
G46	Complex pocket milling	.7	2
G51	Mirror function	.2	2
G52	Machine coordinates programming	.22	1
G53	Inhibit DAT1	.11	1
G54	Activate DAT1	.12	1
G64	Roughing cycles	.28	2
G65	Cycle de gorge	.29	2
G70	Inches programming	.21	1
G73	Cancel scaling	.9	2
G74	Activate scaling	.10	2

G80	Cancel machining cycles	**.24**	**2**
G90	Absolute programming	**.19**	**1**
G91	Incremental programming	**.20**	**1**
G92	Tangential feed rate	**.5**	**1**
G93	Feed in 1/T	**.15**	**1**
G94	Feed per mn	**.0**	**1**
G95	Feed per rev.	**.1**	**1**
G96	Activate CSS	**.3**	**1**
G97	Cancel CSS	**.4**	**1**
G999	Block concatenation	**.11**	**2**

5.1.6 System identification ► R_Ident_Info

This request returns different information about the system configuration as shown in the online visualization below.

The serial number of the system is returned by another function.



5.1.7 List of L Parameters ► R_L_Vars

This channel specific request will return the value of 10 consecutive L parameters.

The request will not work if the list passes the limit of L parameters:
e.g. Request the reading starting at L102.

R_GEN_007

```
PROGRAM R_GEN_007

VAR
    oR_L_Param                  : R_L_Vars;
    re_Answer1                  : REAL;
    re_Answer2                  : REAL;
    re_Answer3                  : REAL;
    re_Answer4                  : REAL;
    re_Answer5                  : REAL;
    re_Answer6                  : REAL;
    re_Answer7                  : REAL;
    re_Answer8                  : REAL;
    re_Answer9                  : REAL;
    re_Answer10                 : REAL;
```

```

oR_L_Param.CNCAdr               := ADR(NCK);
oR_L_Param.byChannel            := 0;
                    // Channels numbered 0 to 7
oR_L_Param.uiLnumber            := 102;
                    // Read from L102

oR_L_Param(enable := xEnable);
                    // Triggers the request

IF oR_L_Param.uiResult  = 0 THEN
    xEnable         := FALSE;
    re_Answer1      := oR_L_Param.reLVars_0;
    re_Answer2      := oR_L_Param.reLVars_1;
    re_Answer3      := oR_L_Param.reLVars_2;
    re_Answer4      := oR_L_Param.reLVars_3;
    re_Answer5      := oR_L_Param.reLVars_4;
    re_Answer6      := oR_L_Param.reLVars_5;
    re_Answer7      := oR_L_Param.reLVars_6;
    re_Answer8      := oR_L_Param.reLVars_7;
    re_Answer9      := oR_L_Param.reLVars_8;
    re_Answer10     := oR_L_Param.reLVars_9;
END_IF
```

5.1.8 Reading of a dialog ($) message ► R_Message

This channel specific request will return a string corresponding to a $ message posted.

In addition to the string itself, the number of characters in the string is also reported.

R_GEN_008

```
PROGRAM R_GEN_008

VAR
    oR_Message                  : R_Message;
    by_Answer1                  : BYTE;
    st_Answer2                  : STRING;
END_VAR
```

```
oR_Message.CNCAdr           := ADR(NCK);
oR_Message.woChannel        := -1;
                            // Channels numbered 0 to 7 or -1

oR_Message(enable := xEnable);
                            // Triggers the request

IF oR_Message.uiResult  = 0 THEN
    xEnable         := FALSE;
    by_Answer1      := oR_Message.byCharQ;
    st_Answer2      := oR_Message.stMessage;
END_IF
```

5.1.9 Reading of an NCK parameter ► R_Nck_Par

This request will return the values of the different words of a parameter according to the parameter format (refer to the Flexium or Flexium+ Parameters manual M00021 or M00041).

Due to the high possible number of words it may be necessary to invoke the request several times by increasing the Offset attribute.
The flag *byByte_4* indicates if the whole parameter has been read or not.

The offset value is given in Bytes, regardless of the type of parameters.

R_GEN_009

```
VAR
    oR_NCK_param                : R_Nck_Par;
    woAnswer0                   : WORD;
    woAnswer1                   : WORD;
    woAnswer2                   : WORD;
    woAnswer3                   : WORD;
    byAnswer4                   : BYTE;
    byAnswer5                   : BYTE;
    inAnswer6                   : INT;
    diAnswer7                   : DINT;
    siAnswer8                   : SINT;
    udAnswer9                   : UDINT;
    uiAnswer10                  : UINT;
    diAnswer11                  : DINT;
    stAnswer12                  : STRING;
    lrAnswer13                  : LREAL;
END_VAR
```

```
oR_NCK_param.CNCAdr             := ADR(NCK);
oR_NCK_param.uiPnumber          := 49;
                                // Parameter to read
oR_NCK_param.uiPoffset          := 2;
                                // Offset given in Bytes

oR_NCK_param(enable := xEnable);
                                // Triggers the request

IF oR_NCK_param.uiResult    = 0 THEN
    xEnable         := FALSE;
    woAnswer0       := oR_NCK_param.woWord_0;       // Parameter number
    woAnswer1       := oR_NCK_param.woWord_1;       // Parameters type
    woAnswer2       := oR_NCK_param.woWord_2;       // Parameter point
    woAnswer3       := oR_NCK_param.woWord_3;       // Bytes read
    byAnswer4       := oR_NCK_param.byByte_4;       // read complete or not
    byAnswer5       := oR_NCK_param.byValue;        // for parameters type 0
    inAnswer6       := oR_NCK_param.inValue;        // for parameters type 1
    diAnswer7       := oR_NCK_param.diValue;        // for parameters type 2 and 6
    siAnswer8       := oR_NCK_param.siValue;        // for parameters type 3
    udAnswer9       := oR_NCK_param.udValue;        // for parameters type 4
    uiAnswer10      := oR_NCK_param.uiValue;        // for parameters type 5 and 7
    diAnswer11      := oR_NCK_param.diValue;        // for parameters type 2 and 6
    stAnswer12      := oR_NCK_param.stValue;        // for parameters type 8
    lrAnswer13      := oR_NCK_param.lrValue;        // for parameters type 9 and 10
END_IF
```

5.1.10 Reading of the overrides values ► R_Override_All

This request will return the value of the 8 feed overrides, regardless of the number of channels declared.

R_GEN_010

```
PROGRAM R_GEN_010

VAR
    oR_Override                 : R_Override_All;
    di_Answer1                  : DINT;
    di_Answer2                  : DINT;
    di_Answer3                  : DINT;
    di_Answer4                  : DINT;
    di_Answer5                  : DINT;
    di_Answer6                  : DINT;
    di_Answer7                  : DINT;
    di_Answer8                  : DINT;
    diOverrideCh8               : DINT;
END_VAR
```

```
oR_Override.CNCAdr          := ADR(NCK);

oR_Override(enable := xEnable);
                                // Trigger the request

IF oR_Override.uiResult = 0 THEN
    xEnable         := FALSE;
    di_Answer1      := oR_Override.byOverrideCh1;
    di_Answer2      := oR_Override.byOverrideCh2;
    di_Answer3      := oR_Override.byOverrideCh3;
    di_Answer4      := oR_Override.byOverrideCh4;
    di_Answer5      := oR_Override.byOverrideCh5;
    di_Answer6      := oR_Override.byOverrideCh6;
    di_Answer7      := oR_Override.byOverrideCh7;
    di_Answer8      := oR_Override.byOverrideCh8;
END_IF
```

5.1.11 Reading the part program memory size ► R_Program_Memory_Size

This request returns the total as well as the available size of each segment of part program memory. In addition it also returns the stack size (for symbolic variables).

R_GEN_011

```
PROGRAM R_GEN_011

VAR
    oR_Memory_Size                  : R_Program_Memory_Size;
    udAnswer1                       : UDINT;
    udAnswer2                       : UDINT;
    udAnswer3                       : UDINT;
    udAnswer4                       : UDINT;
    udAnswer5                       : UDINT;
    udAnswer6                       : UDINT;
    udAnswer7                       : UDINT;
    udAnswer8                       : UDINT;
    udAnswer9                       : UDINT;
END_VAR
```

```

oR_Memory_Size.CNCAdr           := ADR(NCK);

oR_Memory_Size(enable := xEnable);
                                    // Trigger the request

IF oR_Memory_Size.uiResult  = 0 THEN
    xEnable          := FALSE;
    udAnswer1            := oR_Memory_Size.udStackSIze;
    udAnswer2            := oR_Memory_Size.udTotalSize;
    udAnswer3            := oR_Memory_Size.udZone0Free;
    udAnswer4            := oR_Memory_Size.udZone1Free;
    udAnswer5            := oR_Memory_Size.udZone1Size;
    udAnswer6            := oR_Memory_Size.udZone2Free;
    udAnswer7            := oR_Memory_Size.udZone2Size;
    udAnswer8            := oR_Memory_Size.udZone3Free;
    udAnswer9            := oR_Memory_Size.udZone3Size;
END_IF
```

5.1.12 Reading the system serial number ► R_Serial_Number

As indicated before, the system serial number is only returned by this request and not by System Identification.

R_GEN_012

```
PROGRAM R_GEN_012

VAR
    oR_Serial                   : R_Serial_Number;
    stAnswer2                   : STRING(8);
END_VAR
```

```

oR_Serial.CNCAdr            := ADR(NCK);

oR_Serial(enable := xEnable);
                                    // Trigger the request

IF oR_Serial.uiResult   = 0 THEN
    xEnable             := FALSE;
    stAnswer2           := oR_Serial.stSerialNumber;
END_IF
```

5.1.13 Reading a solicited input ► R_Solicited_Input

This request indicates if an operator input is requested by a $ message.
In addition it also returns the current CNC mode.

R_GEN_013

```
PROGRAM R_GEN_013

VAR
    oR_Solicited_Input                  : R_Solicited_Input;
    xAnswer1                            : BOOL;
    enumAnswer2                         : EnumMode;
END_VAR
```

```

oR_Solicited_Input.CNCAdr           := ADR(NCK);
oR_Solicited_Input.byChannel        := 0;
                                        // Channels numbered 0 to 7
oR_Solicited_Input.byAxisNo         := 0;

oR_Solicited_Input(enable := xEnable);
                                        // Trigger the request

IF oR_Solicited_Input.uiResult  = 0 THEN
    xEnable         := FALSE;
    xAnswer1        := oR_Solicited_Input.boSolInput;
    enumAnswer2     := oR_Solicited_Input.EnumMode;
END_IF
```

5.1.14 Reading the measure of a spindle ► R_Spindle_Meas

This request returns the current measure of a spindle according to the spindle physical address.

R_GEN_014

```
PROGRAM R_GEN_014

VAR
    oR_Spindle_Meas                 : R_Spindle_Meas;
    xAnswer1                        : BOOL;
    ui_Answer2                      : UDINT;
    xAnswer0                        : BOOL;
END_VAR
```

```
oR_Spindle_Meas.CNCAdr          := ADR(NCK);
oR_Spindle_Meas.bySpindleNo     := 31;

oR_Spindle_Meas(enable := xEnable);
                                // Trigger the request

IF oR_Spindle_Meas.uiResult = 0 THEN
    xEnable         := FALSE;
    xAnswer1        := oR_Spindle_Meas.ok;
    ui_Answer2      := oR_Spindle_Meas.uiMeas;
    xAnswer0        := oR_Spindle_Meas.ok;
END_IF
```

5.1.15 Reading the tool offset H dimension ► R_Tool_H

This request will return the H value of 10 consecutive tool offsets.

R_GEN_015

```
PROGRAM R_GEN_015

VAR
    oR_Tool_H                   : R_Tool_H;
    wo_Answer1                  : DINT;
    wo_Answer2                  : DINT;
    wo_Answer3                  : DINT;
    wo_Answer4                  : DINT;
    wo_Answer5                  : DINT;
    wo_Answer6                  : DINT;
    wo_Answer7                  : DINT;
    wo_Answer8                  : DINT;
    wo_Answer9                  : DINT;
    wo_Answer10                 : DINT;
END_VAR
```

```

oR_Tool_H.CNCAdr            := ADR(NCK);
oR_Tool_H.byH_No            := 1;           // Tool offset n° 1


oR_Tool_H(enable := xEnable);               // Trigger the request

IF oR_Tool_H.uiResult   = 0 THEN
    xEnable         := FALSE;
    wo_Answer1      := oR_Tool_H.diE56_n0;
    wo_Answer2      := oR_Tool_H.diE56_n1;
    wo_Answer3      := oR_Tool_H.diE56_n2;
    wo_Answer4      := oR_Tool_H.diE56_n3;
    wo_Answer5      := oR_Tool_H.diE56_n4;
    wo_Answer6      := oR_Tool_H.diE56_n5;
    wo_Answer7      := oR_Tool_H.diE56_n6;
    wo_Answer8      := oR_Tool_H.diE56_n7;
    wo_Answer9      := oR_Tool_H.diE56_n8;
    wo_Answer10     := oR_Tool_H.diE56_n9;
END_IF
```

5.1.16 Reading a memory zone directory ► R_Directory

This request will return the list of programs of a particular memory zone. The data are stored in a file on the system mass memory.

As this list can be long the request can take a certain time to get the complete list. The reading is complete when the *iuResult* flag is zero.

R_GEN_016

```
PROGRAM R_GEN_016

VAR
    oR_Directory                    : R_Directory;
END_VAR


oR_Directory.CNCAddress             := ADR(NCK);
oR_Directory.byRamZone0123          := 0;
                                        // Memory zone 0
oR_Directory.stFileName             := 'C:\Temp\Directory.txt';

IF xEnable THEN
    oR_Directory();                 // Trigger the request
END_IF

IF oR_Directory.uiResult    = 0 THEN
    xEnable         := FALSE;
END_IF
```

5.1.17 Read a part program from NC memory ► R_Upload2_PP

This request replaces the previous *R_Upload_PP*.
It will read a part program in NC memory and store it as a file in the system mass memory.

The request requires the number of the part program in NC memory (zone 0) as well as the path name used to store it in mass memory.

The part program number passed to the request is separated in two elements. The last digit represents the index and the other digits the part program number itself.

The part program is always read from memory zone 0.

R_GEN_017

```
1  PROGRAM R_GEN_017
2
3  VAR
4      oR_Upload_Program                    : R_Upload2_PP;
5  END_VAR
```

```
1  oR_Upload_Program.CNCAddress             := ADR(NCK);
2  oR_Upload_Program.stFileName             := 'C:\Temp\PartProgram.xpi';
3  oR_Upload_Program.udPPnumber             := 10001;
4
5  // File with same name is automatically deleted
6
7  IF xEnable THEN
8      oR_Upload_Program();                 // Execute request
9  END_IF
10
11 IF (oR_Upload_Program.uiResult  = 0) THEN
12     xEnable          := FALSE;
13 END_IF
14
```

5.2 Flexium requests

5.2.1 Reading an axis measure by address ► R_Axis_Meas_ByAdr

This request will return an axis measure according to the axis physical address. It returns the data for one axis at a time.

To get the data for all axes of a channel, use *Read_Axis_Meas_ByAdr*.

R_Std_001

```
PROGRAM R_Std_001
VAR
    oR_byAdr                    : R_Axis_Meas_ByAdr;
    di_AxisPstn                 : DINT;
END_VAR
```

```
oR_byAdr.CNCAdr                 := ADR(NCK);
oR_byAdr.byAdresse              := 0;
                                // Reading for axis @0

oR_byAdr(enable := xEnable);;

IF oR_byAdr.uiResult = 0 THEN
    xEnable     := FALSE;
    di_AxisPstn := oR_byAdr.di_Axis_Meas;
END_IF
```

5.2.2 Reading the measure of all axes in a channel ► R_Axis_Meas

This request will return the measure of all axes in a particular channel.

Contrary to *R_Axis_Meas_ByAdr* the data are sorted according to the axis name and not the physical address.

R_Std_002

```
PROGRAM R_Std_002
VAR
    oR_Meas                     : R_Axis_Meas;
    diMeas_X                    : DINT;
END_VAR
```

```
oR_Meas.CNCAdr                  := ADR(NCK);
oR_Meas.byChannel               := 0;
                                // Channels numbered 0 to 7

oR_Meas(enable := xEnable);;

IF oR_Meas.uiResult = 0 THEN
    xEnable     := FALSE;
    diMeas_X    := oR_Meas.diX_Axis;
END_IF
```

5.2.3 Reading machine data about axes in motion ► R_Axis_OM

This request returns Position in the machine referential as well as distance to go, lag and resolution about the axes of a particular channel.

The data are returned in a structure *Axis[i]* and are individually validated by a flag *xxxValid* e.g. *LagValid*.

R_Std_003

```
PROGRAM R_Std_003
VAR
    oR_Meas_OM                  : R_Axis_OM;
    Pos_0                       : DINT;
    ToGo_0                      : DINT;
    Lag_0                       : DINT;
    Res_0                       : BYTE;
END_VAR
```

```
oR_Meas_OM.CNCAdr               := ADR(NCK);
oR_Meas_OM.byChannel            := 0;
                                // Channels numbered 0 to 7

oR_Meas_OM(enable := xEnable);;

IF oR_Meas_OM.uiResult = 0 THEN
    xEnable             := FALSE;
    IF oR_Meas_OM.Axis[0].PosValid THEN
        Pos_0           := oR_Meas_OM.Axis[0].Position;
    END_IF
    IF oR_Meas_OM.Axis[0].ToGoValid THEN
        ToGo_0          := oR_Meas_OM.Axis[0].ToGo;
    END_IF
    IF oR_Meas_OM.Axis[0].LagValid THEN
        Lag_0           := oR_Meas_OM.Axis[0].Lag;
    END_IF
    Res_0               := oR_Meas_OM.Axis[0].Resolution;

END_IF
```

5.2.4 Reading part origin data about axes in motion ► R_Axis_OP

This request returns the axes position in the part referential as well as distance to go, lag and resolution for a particular channel.

The data are returned in a structure *Axis[i]* and are individually validated by a flag *xxxValid* e.g. *LagValid*.

R_Std_004

```
PROGRAM R_Std_004
VAR
    oR_Meas_OP                  : R_Axis_OP;
    Pos_0                       : DINT;
    ToGo_0                      : DINT;
    Lag_0                       : DINT;
    Res_0                       : BYTE;
END_VAR
```

```
oR_Meas_OP.CNCAdr               := ADR(NCK);
oR_Meas_OP.byChannel            := 0;
                                // Channels numbered 0 to 7

oR_Meas_OP(enable := xEnable);;

IF oR_Meas_OP.uiResult = 0 THEN
    xEnable     := FALSE;
    IF oR_Meas_OP.Axis[0].PosValid THEN
        Pos_0           := oR_Meas_OP.Axis[0].Position;
    END_IF
    IF oR_Meas_OP.Axis[0].ToGoValid THEN
        ToGo_0          := oR_Meas_OP.Axis[0].ToGo;
    END_IF
    IF oR_Meas_OP.Axis[0].LagValid THEN
        Lag_0           := oR_Meas_OP.Axis[0].Lag;
    END_IF
    Res_0               := oR_Meas_OP.Axis[0].Resolution;
END_IF
```

5.2.5 Reading axes programmed position ► R_Axis_Prog

This request returns the programmed position of all axes in a particular channel.
The data are returned with respect to the axis name in the channel.

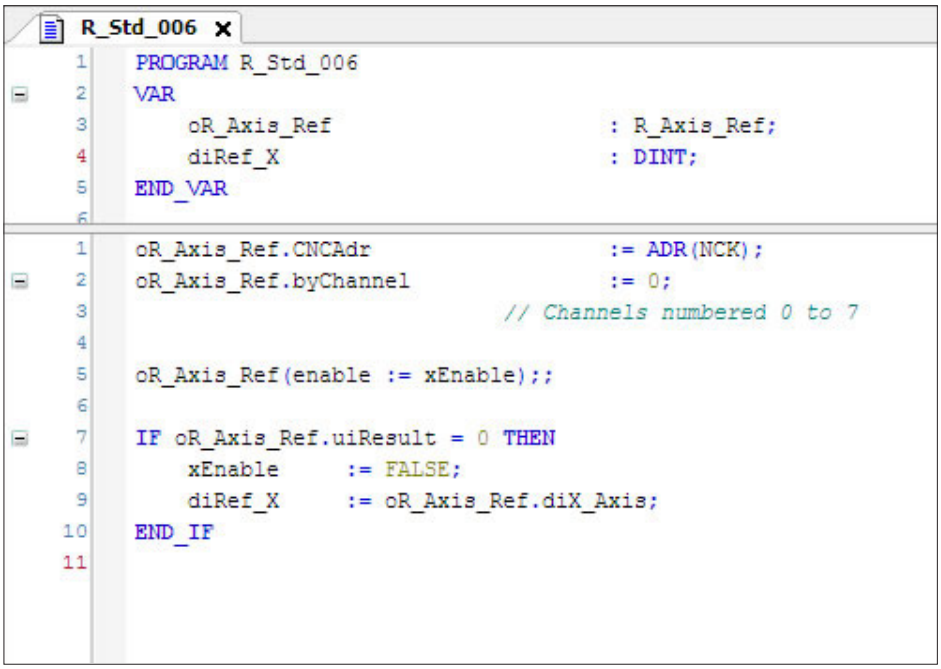
R_Std_005

```
PROGRAM R_Std_005
VAR
    oR_Dim_Prog                 : R_Axis_Prog;
    diProg_X                    : DINT;
END_VAR
```

```
oR_Dim_Prog.CNCAdr                  := ADR(NCK);
oR_Dim_Prog.byChannel               := 0;
                                // Channels numbered 0 to 7

oR_Dim_Prog(enable := xEnable);;

IF oR_Dim_Prog.uiResult = 0 THEN
    xEnable     := FALSE;
    diProg_X        := oR_Dim_Prog.diX_Axis;
END_IF
```

5.2.6 Reading axes reference ► R_Axis_Ref

This request returns the reference value (in the machine referential) of the axes in a particular channel. The data are returned with respect to the axis name.



5.2.7 Reading a part program block ► R_Block

This request returns a string corresponding to a particular part program block.

Input data will be the memory zone where the part program is searched for, the part program number (times 10 and including the index), the block number and the offset (in number of blocks) relative to this number. The string returned ends with $N (meaning end of block).

In case of non numbered part program, the number 0 corresponds to the first line of program.

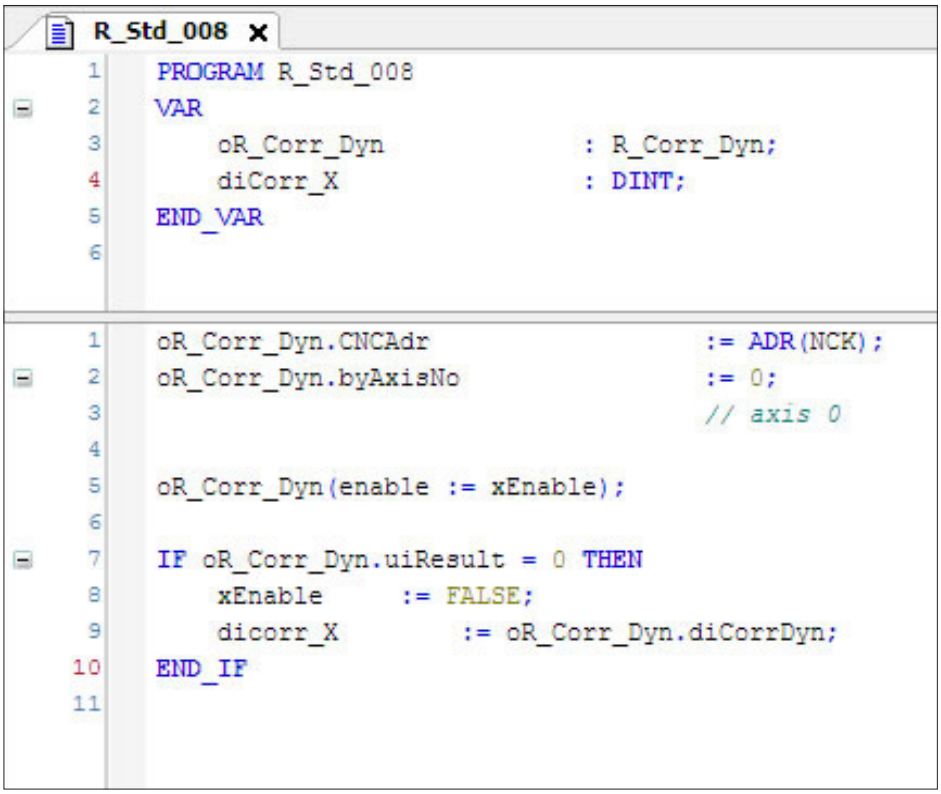
R_Std_007

```
PROGRAM R_Std_007
VAR
    oR_Block                    : R_Block;
    Block                       : STRING;
END_VAR
```

```
oR_Block.CNCAdr                 := ADR(NCK);
oR_Block.byZone                 := 0;
                                // memory zone 0
oR_Block.diPPnumber             := 1560;
oR_Block.woBlocknber            := 1080;
oR_Block.woOffset               := 0;

oR_Block(enable := xEnable);

IF oR_Block.uiResult = 0 THEN
    xEnable     := FALSE;
    Block       := oR_Block.stBlock;
END_IF
```

5.2.8 Reading an axis measure correction (E952xx) ► R_Corr_Dyn

This request returns the value of parameter E952xx (measure correction) of a particular axis.



5.2.9 Reading the part program block in execution ► R_Curr_Block

This request returns a string corresponding to the block in execution. As the answer can take several steps a Read_Complete status flag is part of the answer.

Return code of byReadStatus:
- 0x06 : Block reading incomplete : Increment *woSequence* to get the rest of the block·
- 0x0F : Block read finished *woSequence* = 0 to read new block·
- 0x19 : Sequencing error.

The first sequence number should be 0.
If the block is not read entirely then read the next part after incrementing the sequence number : *woSequence* = *woSequence* +1 and so on until *byReadStatus* = 0x0F.

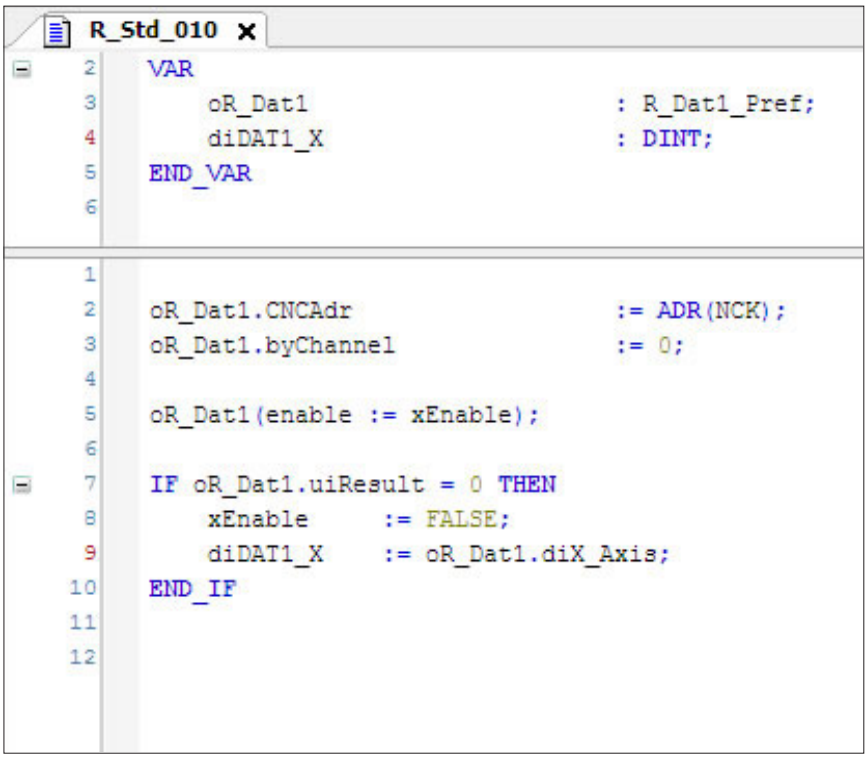
R_Std_009

```
PROGRAM R_Std_009
VAR
    oR_Curr_block               : R_Curr_Block;
    Block                       : STRING;
    Read_Complete               : BOOL;
END_VAR
```

```
oR_Curr_block.CNCAdr                := ADR(NCK);
oR_Curr_block.woChannel             := 0;
oR_Curr_block.woSequence            := 0;
oR_Curr_block.

oR_Curr_block(enable := xEnable);

IF oR_Curr_block.uiResult = 0 THEN
    xEnable     := FALSE;
    Block       := oR_Curr_block.stCurrBlock;
END_IF

IF oR_Curr_block.byReadStatus   = 16#0F THEN
    Read_Complete   := TRUE;
    // If NOT incrememnt woSequence and read again
END_IF
```

5.2.10 Reading current Dat1 value ► R_Dat1_Pref

This request returns the current DAT1 set of values for a particular channel.
The data are returned with respect to the axes names.



5.2.11 Reading current Dat2 value ► R_Dat2_Dec1

This request returns the current DAT2 set of values for a particular channel.
The data are returned with respect to the axes names.

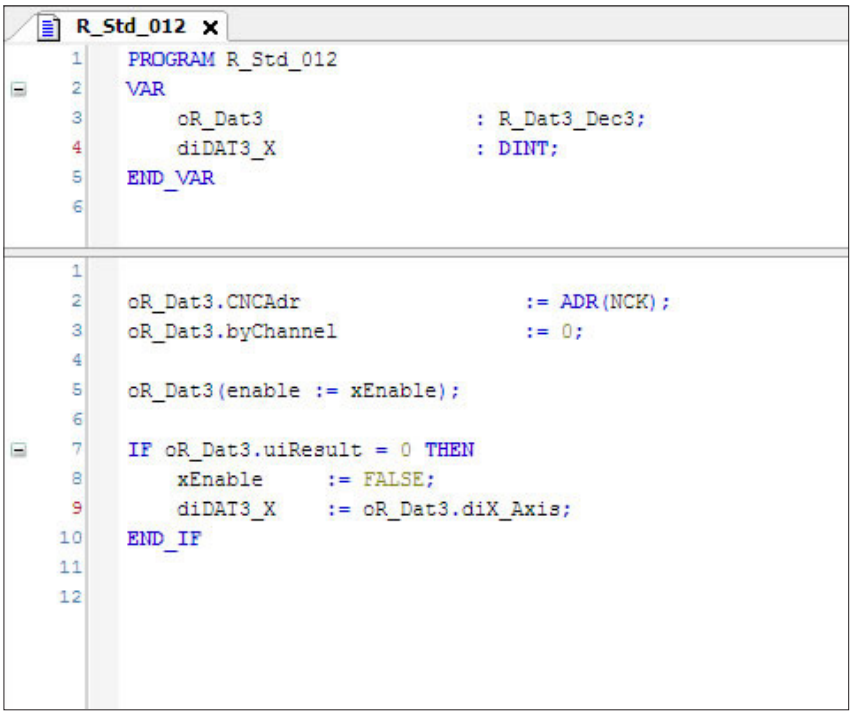
R_Std_011

```
1  PROGRAM R_Std_011
2  VAR
3      oR_Dat2                    : R_Dat2_Dec1;
4      diDAT2_X                   : DINT;
5  END_VAR
6
```

```
1
2  oR_Dat2.CNCAdr                 := ADR(NCK);
3  oR_Dat2.byChannel              := 0;
4
5  oR_Dat2(enable := xEnable);
6
7  IF oR_Dat2.uiResult = 0 THEN
8      xEnable      := FALSE;
9      diDAT2_X     := oR_Dat2.diX_Axis;
10 END_IF
11
12
```

5.2.12 Reading current Dat3 value ► R_Dat3_Dec3

This request returns the current DAT3 set of values for a particular channel. The data are returned with respect to the axes names.



5.2.13 Reading Drives Test points values ► R_Drive_TPValue_V2

This request returns up to eight test point values of one or several drives. The values are returned in Drive Internal units.

The test points configuration is defined in the drive parameters V130 to V133 and the values read are expressed in physical units.

R_Std_020

```
PROGRAM R_Std_020
VAR
    oR_TP_Value                 : R_Drive_TPValue_V2;
    Drives                      : ARRAY[1..8] OF BYTE
                                    := [0,0,0,0,1,1,1,1];
    Points                      : ARRAY[1..8] OF BYTE
                                    := [0,1,2,3,0,1,2,3];
    Values                      : ARRAY[1..8] OF LREAL;
END_VAR
```

```

oR_TP_Value.CNCAdr              := ADR(NCK);
oR_TP_Value.DriveAddresse       := ADR(Drives);
oR_TP_Value.DriveTP             := ADR(Points);
oR_TP_Value.Number              := 4;

oR_TP_Value(enable := xEnable);

IF oR_TP_Value.uiResult = 0 THEN
    xEnable     := FALSE;
    Values[1]   := oR_TP_Value.Values[0];
    Values[2]   := oR_TP_Value.Values[1];
    Values[3]   := oR_TP_Value.Values[2];
    Values[4]   := oR_TP_Value.Values[3];
END_IF

```

5.2.14 Reading a set of E8xyyy parameters ► R_E8xyyy

This request returns a set of 10 E8…. parameters from the series E80, E81, E82 E83 or E84.

R_Std_030

```
PROGRAM R_Std_030
VAR
    oR_E8xyyy                       : R_E8xyyy;
    Para                            : UDINT     := 80000;
    Values                          : ARRAY[1..10] OF DINT;
END_VAR
```

```
oR_E8xyyy.CNCAdr                    := ADR(NCK);
oR_E8xyyy.udE8xyyyNo                := Para;

oR_E8xyyy(enable := xEnable);

IF oR_E8xyyy.uiResult = 0 THEN
    xEnable     := FALSE;
    Values[1]   := oR_E8xyyy.diE8_n0;
    Values[2]   := oR_E8xyyy.diE8_n1;
    Values[3]   := oR_E8xyyy.diE8_n2;
    Values[4]   := oR_E8xyyy.diE8_n3;
    Values[5]   := oR_E8xyyy.diE8_n4;
    Values[6]   := oR_E8xyyy.diE8_n5;
    Values[7]   := oR_E8xyyy.diE8_n6;
    Values[8]   := oR_E8xyyy.diE8_n7;
    Values[9]   := oR_E8xyyy.diE8_n8;
    Values[10]  := oR_E8xyyy.diE8_n9;

END_IF
```

5.2.15 Reading a program parameter ► R_E_Param

This request returns the value of a particular E parameter.

Other requests on parameters are:

- *R_E_8xyyy()* returns a set of 10 parameters
- *R_Tool()* returns all data about a tool offset
- *R_Tool_H()* return H values by set of 10.

R_Std_031

```
1  PROGRAM R_Std_031
2  VAR
3      oR_E_Param                    : R_E_Param;
4      Parameter                     : UDINT     := 50000;
5      Value                         : DINT;
6  END_VAR
7
```

```
1
2  oR_E_Param.CNCAdr                 := ADR(NCK);
3  oR_E_Param.byChannel              := 0;
4  oR_E_Param.udE_number             := Parameter;
5
6  oR_E_Param(enable := xEnable);
7
8  IF oR_E_Param.uiResult = 0 THEN
9      xEnable      := FALSE;
10     Value        := oR_E_Param.diValue_E;
11 END_IF
12
```

5.2.16 Reading the Interpolation data ► R_Interpo

This request returns current interpolation data of a particular channel.
The data are curvilinear distance to go, feedrate as well as the override value.

R_Std_040

```
PROGRAM R_Std_040
VAR
    oR_Interpo                  : R_Interpo;
    di_Distance                 : DINT;
    di_Feed                     : DINT;
    di_Override                 : DINT;
END_VAR
```

```

oR_Interpo.CNCAdr               := ADR(NCK);
oR_Interpo.byChannel            := 0;

oR_Interpo(enable := xEnable);

IF oR_Interpo.uiResult = 0 THEN
    xEnable         := FALSE;
    di_Distance     := oR_Interpo.diDistance;
    di_Feed         := oR_Interpo.diFeedProg;
    di_Override     := oR_Interpo.diOverride;
END_IF

```

5.2.17 Reading machine origin values ► R_Mach_Orig

This request returns the machine origin offset values (equivalent to parameter P16) according to the axis physical address. The values are returned for three consecutive addresses.

R_Std_041

```
PROGRAM R_Std_041
VAR
    oR_Mach_Origin                  : R_Mach_Orig;
    di_Axe1                         : DINT;
    di_Axe2                         : DINT;
    di_axe3                         : DINT;
END_VAR
```

```

oR_Mach_Origin.CNCAdr               := ADR(NCK);
oR_Mach_Origin.byAxisNo             := 1;

oR_Mach_Origin(enable := xEnable);

IF oR_Mach_Origin.uiResult = 0 THEN
    xEnable         := FALSE;
    di_Axe1         := oR_Mach_Origin.diAxe_n1;
    di_Axe2         := oR_Mach_Origin.diAxe_n2;
    di_Axe3         := oR_Mach_Origin.diAxe_n3;
END_IF

```

5.2.18 Reading Maximum Dynamic Travel values ► R_Max_Dyn_Travel

This request returns the maximum dynamic travels of the axes in a particular channel. The values are returned in reference to the axes names.

R_Std_042

```
PROGRAM R_Std_042
VAR
    oR_Max_Dyn_Travel               : R_Max_Dyn_Travel;
    di_MaxDyn_A                     : DINT;
    di_MaxDyn_B                     : DINT;
    di_MaxDyn_C                     : DINT;
    di_MaxDyn_U                     : DINT;
    di_MaxDyn_V                     : DINT;
    di_MaxDyn_W                     : DINT;
    di_MaxDyn_X                     : DINT;
    di_MaxDyn_Y                     : DINT;
    di_MaxDyn_Z                     : DINT;
END_VAR
```

```

oR_Max_Dyn_Travel.CNCAdr                    := ADR(NCK);
oR_Max_Dyn_Travel.byChannel                 := 0;

oR_Max_Dyn_Travel(enable := xEnable);

IF oR_Max_Dyn_Travel.uiResult = 0 THEN
    xEnable         := FALSE;
    di_MaxDyn_A                     := oR_Max_Dyn_Travel.diA_Axis;
    di_MaxDyn_B                     := oR_Max_Dyn_Travel.diB_Axis;
    di_MaxDyn_C                     := oR_Max_Dyn_Travel.diC_Axis;;
    di_MaxDyn_U                     := oR_Max_Dyn_Travel.diU_Axis;
    di_MaxDyn_V                     := oR_Max_Dyn_Travel.diV_Axis;;
    di_MaxDyn_W                     := oR_Max_Dyn_Travel.diW_Axis;;
    di_MaxDyn_X                     := oR_Max_Dyn_Travel.diX_Axis;;
    di_MaxDyn_Y                     := oR_Max_Dyn_Travel.diY_Axis;;
    di_MaxDyn_Z                     := oR_Max_Dyn_Travel.diZ_Axis;;
END_IF

```

5.2.19 Reading Minimum Dynamic Travel values ► R_Min_Dyn_Travel

This request returns the minimum dynamic travels of the axes in a particular channel. The values are returned in reference to the axes names.

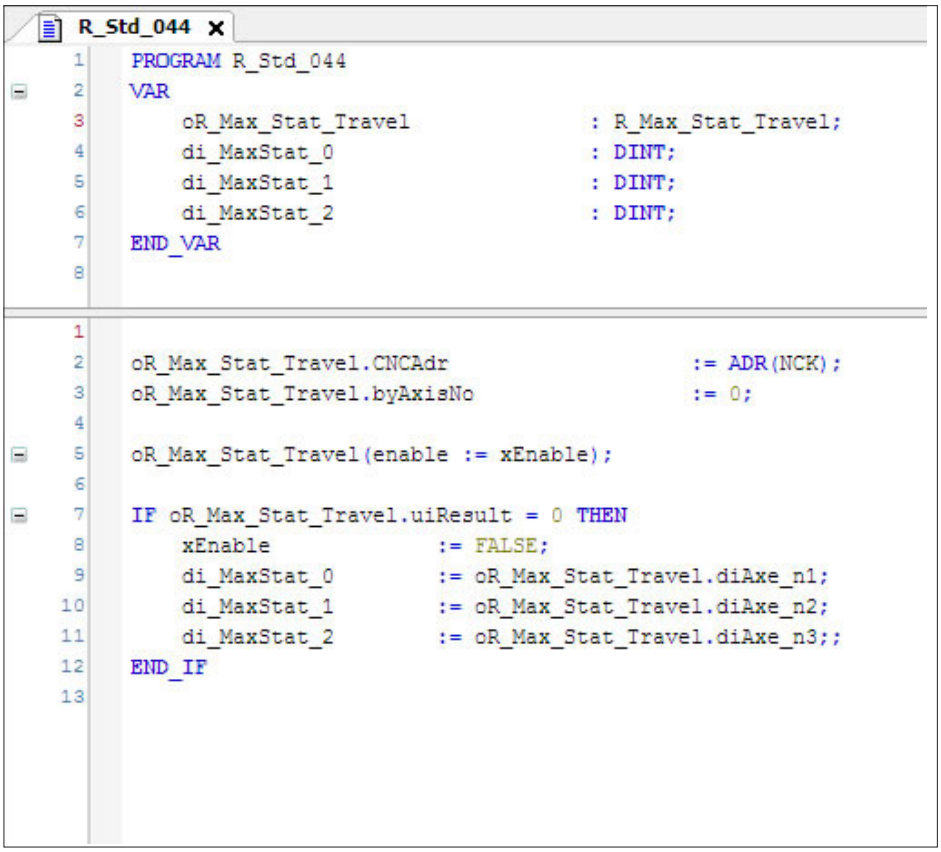
R_Std_043

```
PROGRAM R_Std_043
VAR
    oR_Min_Dyn_Travel           : R_Min_Dyn_Travel;
    di_MinDyn_A                 : DINT;
    di_MinDyn_B                 : DINT;
    di_MinDyn_C                 : DINT;
    di_MinDyn_U                 : DINT;
    di_MinDyn_V                 : DINT;
    di_MinDyn_W                 : DINT;
    di_MinDyn_X                 : DINT;
    di_MinDyn_Y                 : DINT;
    di_MinDyn_Z                 : DINT;
END_VAR
```

```
oR_Min_Dyn_Travel.CNCAdr                    := ADR(NCK);
oR_Min_Dyn_Travel.byChannel                 := 0;

oR_Min_Dyn_Travel(enable := xEnable);

IF oR_Min_Dyn_Travel.uiResult = 0 THEN
    xEnable         := FALSE;
    di_MinDyn_A                 := oR_Min_Dyn_Travel.diA_Axis;
    di_MinDyn_B                 := oR_Min_Dyn_Travel.diB_Axis;
    di_MinDyn_C                 := oR_Min_Dyn_Travel.diC_Axis;
    di_MinDyn_U                 := oR_Min_Dyn_Travel.diU_Axis;
    di_MinDyn_V                 := oR_Min_Dyn_Travel.diV_Axis;
    di_MinDyn_W                 := oR_Min_Dyn_Travel.diW_Axis;
    di_MinDyn_X                 := oR_Min_Dyn_Travel.diX_Axis;
    di_MinDyn_Y                 := oR_Min_Dyn_Travel.diY_Axis;
    di_MinDyn_Z                 := oR_Min_Dyn_Travel.diZ_Axis;
END_IF
```

5.2.20 Reading Maximum Static Travel values ► R_Max_Stat_Travel

This request returns the maximum static travels of axes.
The values are given for three consecutives physical addresses.



5.2.21 Reading Minimum Static Travel values ► R_Min_Stat_Travel

This request returns the maximum static travels of axes.
The values are given for three consecutives physical addresses.

R_Std_045

```
PROGRAM R_Std_045
VAR
    oR_Min_Stat_Travel              : R_Min_Stat_Travel;
    di_MinStat_0                    : DINT;
    di_MinStat_1                    : DINT;
    di_MinStat_2                    : DINT;
END_VAR
```

```

oR_Min_Stat_Travel.CNCAdr                   := ADR(NCK);
oR_Min_Stat_Travel.byAxisNo                 := 0;

oR_Min_Stat_Travel(enable := xEnable);

IF oR_Min_Stat_Travel.uiResult = 0 THEN
    xEnable             := FALSE;
    di_MinStat_0        := oR_Min_Stat_Travel.diAxe_n1;
    di_MinStat_1        := oR_Min_Stat_Travel.diAxe_n2;
    di_MinStat_2        := oR_Min_Stat_Travel.diAxe_n3;;
END_IF
```

5.2.22 Reading NCK detected error ► R_Nck_Error

This request returns the information about an NCK detected error. The data are the block number, line number where the error has occurred as well as the error number.

R_Std_046

```
PROGRAM R_Std_046
VAR
    oR_NckError                 : R_Nck_Error;
    wo_Block                    : WORD;
    wo_Line                     : WORD;
    wo_Error                    : WORD;
END_VAR
```

```
oR_NckError.CNCAdr              := ADR(NCK);
oR_NckError.byChannel           := 0;

oR_NckError(enable := xEnable);

IF oR_NckError.uiResult = 0 THEN
    xEnable             := FALSE;
    wo_Block            := oR_NckError.woBlockNumber;
    wo_Line             := oR_NckError.woLineNumber;
    wo_Error            := oR_NckError.woErrorNumber;
END_IF
```

5.2.23 Reading Part program execution tree ► R_Program_Tree

This request returns information about the part program in execution.

The information include block number, the current subroutine number with number of times it has been called as well as all the chain of calling programs (up to eight nesting level).

R_Std_047

```
PROGRAM R_Std_047
VAR
    oR_Prog_Tree                : R_Program_Tree;
    Main                        : UDINT;
    SubPrgs                     : ARRAY [0..7] OF UDINT;
    Calls                       : BYTE;
    Block                       : WORD;
    Line                        : UDINT;
END_VAR
```

```
oR_Prog_Tree.CNCAdr             := ADR(NCK);
oR_Prog_Tree.byChannel          := 0;

oR_Prog_Tree(enable := xEnable);

IF oR_Prog_Tree.uiResult = 0 THEN
    xEnable             := FALSE;
    Main                := oR_Prog_Tree.ProgramData.MainProgram;
    SubPrgs             := oR_Prog_Tree.ProgramData.Subprograms;
    Calls               := oR_Prog_Tree.ProgramData.NumberOfSubprogramCalls;
    Block               := oR_Prog_Tree.ProgramData.BlockNumber;
END_IF
```

5.2.24 Reading Spindle information ► R_Spindle_Complete

This request returns all relevant spindle information excluding the range currently active. The full list of data can be seen in the example below.

R_Std_048

```
PROGRAM R_Std_048
VAR
    oR_Spindle                  : R_Spindle_Complete;
    bChannel                    : BYTE;
    bFormat                     : BYTE;
    xM19                        : BOOL;
    xM3                         : BOOL;
    xM4                         : BOOL;
    xM5                         : BOOL;
    uiOverride                  : UINT;
    lrProgSpeed                 : LREAL;
    lrRealSpeed                 : LREAL;
END_VAR
```

```

oR_Spindle.CNCAdr               := ADR(NCK);
oR_Spindle(enable := xEnable);

IF oR_Spindle.uiResult = 0 THEN
    xEnable                     := FALSE;
    bChannel                    := oR_Spindle.SpindleData[0].Channel;
    bFormat                     := oR_Spindle.SpindleData[0].Format;
    xM19                        := oR_Spindle.SpindleData[0].M19;
    xM3                         := oR_Spindle.SpindleData[0].M3;
    xM4                         := oR_Spindle.SpindleData[0].M4;
    xM5                         := oR_Spindle.SpindleData[0].M5;
    uiOverride                  := oR_Spindle.SpindleData[0].Override;
    lrProgSpeed                 := oR_Spindle.SpindleData[0].SpeedProgrammed;
    lrRealSpeed                 := oR_Spindle.SpindleData[0].SpeedReal;
END_IF

```

5.2.25 Reading Tool offsets data ► R_Tool

This request returns all data about a particular tool offset, excluding the H value that can be obtained via the *Read_E_Param* request.

R_Std_049

```
PROGRAM R_Std_049
VAR
    oR_Tool                     : R_Tool;
    diToolPar1                  : DINT;
    diToolPar2                  : DINT;
    diToolPar3                  : DINT;
    diToolPar4                  : DINT;
    diToolPar5                  : DINT;
    diToolPar6                  : DINT;
    diToolPar7                  : DINT;
END_VAR
```

```
oR_Tool.CNCAdr                  := ADR(NCK);
oR_Tool.byCorrNo                := 1;       // Tool offset number

oR_Tool(enable := xEnable);

IF oR_Tool.uiResult = 0 THEN
    xEnable                     := FALSE;

    diToolPar1                  := oR_Tool.diE50xxx;
    diToolPar2                  := oR_Tool.diE51xxx;
    diToolPar3                  := oR_Tool.diE52xxx;
    diToolPar4                  := oR_Tool.diE53xxx;
    diToolPar5                  := oR_Tool.diE54xxx;
    diToolPar6                  := oR_Tool.diE55xxx;
    diToolPar7                  := oR_Tool.diE57xxx;
END_IF
```

5.3 Flexium+ requests

5.3.1 Reading an axis measure by address ► R_Axis_Meas_ByAdr_Plus

This request will return an axis measure according to the axis physical address. It returns the data for one axis at a time.
To get the data for all axes of a channel, use *Read_Axis_Meas_ByAdr*.

R_Plus_001

```
PROGRAM R_Plus_001
VAR
    oR_byAdr_Plus                   : R_Axis_Meas_ByAdr_Plus;
    lr_AxisPstn                     : LREAL;
END_VAR
```

```
oR_byAdr_Plus.CNCAdr                := ADR(NCK);
oR_byAdr_Plus.byAdresse             := 0;
                                // Reading for axis @0

oR_byAdr_Plus(enable := xEnable);;

IF oR_byAdr_Plus.uiResult = 0 THEN
    xEnable     := FALSE;
    lr_AxisPstn := oR_byAdr_Plus.lr_Axis_Meas;
END_IF
```

5.3.2 Reading the measure of all axes in a channel ► R_Axis_Meas_Plus

This request will return the measure of all axes in a particular channel.
Contrary to *R_Axis_Meas_ByAdr* the data are sorted according to the axis name and not the physical address.

R_Plus_002

```
1  PROGRAM R_Plus_002
2  VAR
3      oR_Meas_Plus                    : R_Axis_Meas_Plus;
4      lrMeas_X                        : LREAL;
5  END_VAR
```

```
1  oR_Meas_Plus.CNCAdr                 := ADR(NCK);
2  oR_Meas_Plus.byChannel              := 0;
3                                  // Channels numbered 0 to 7
4
5  oR_Meas_Plus(enable := xEnable);;
6
7  IF oR_Meas_Plus.uiResult = 0 THEN
8      xEnable      := FALSE;
9      lrMeas_X     := oR_Meas_Plus.lrX_Axis;
10 END_IF
11
```

5.3.3 Reading machine data about axes in motion ► R_Axis_OM_Plus

This request returns Position in the machine referential as well as distance to go, lag and resolution about the axes of a particular channel.

The data are returned in a structure *Axis[i]* and are individually validated by a flag *xxxValid* e.g. *LagValid*.

R_Plus_003

```
PROGRAM R_Plus_003
VAR
    oR_Meas_OM_Plus                 : R_Axis_OM_Plus;
    Pos_0                           : LREAL;
    ToGo_0                          : LREAL;
    Lag_0                           : LREAL;
    Res_0                           : LREAL;
END_VAR
```

```
oR_Meas_OM_Plus.CNCAdr              := ADR(NCK);
oR_Meas_OM_Plus.byChannel           := 0;
                                 // Channels numbered 0 to 7

oR_Meas_OM_Plus(enable := xEnable);;

IF oR_Meas_OM_Plus.uiResult = 0 THEN
    xEnable     := FALSE;
    IF oR_Meas_OM_Plus.Axis[0].PosValid THEN
        Pos_0       := oR_Meas_OM_Plus.Axis[0].Position;
    END_IF
    IF oR_Meas_OM_Plus.Axis[0].ToGoValid THEN
        ToGo_0      := oR_Meas_OM_Plus.Axis[0].ToGo;
    END_IF
    IF oR_Meas_OM_Plus.Axis[0].LagValid THEN
        Lag_0       := oR_Meas_OM_Plus.Axis[0].Lag;
    END_IF
    Res_0           := oR_Meas_OM_Plus.Axis[0].Resolution;

END_IF
```

5.3.4 Reading part origin data about axes in motion ► R_Axis_OP_Plus

This request returns Position in the part programming referential as well as distance to go, lag and resolution about the axes of a particular channel.

The data are returned in a structure *Axis[i]* and are individually validated by a flag *xxxValid* e.g. *LagValid*.

R_Plus_004

```
PROGRAM R_Plus_004
VAR
    oR_Meas_OP_Plus                 : R_Axis_OP_Plus;
    Pos_0                           : LREAL;
    ToGo_0                          : LREAL;
    Lag_0                           : LREAL;
    Res_0                           : LREAL;
END_VAR
```

```
oR_Meas_OP_Plus.CNCAdr              := ADR(NCK);
oR_Meas_OP_Plus.byChannel           := 0;
                            // Channels numbered 0 to 7

oR_Meas_OP_Plus(enable := xEnable);;

IF oR_Meas_OP_Plus.uiResult = 0 THEN
    xEnable     := FALSE;
    IF oR_Meas_OP_Plus.Axis[0].PosValid THEN
        Pos_0       := oR_Meas_OP_Plus.Axis[0].Position;
    END_IF
    IF oR_Meas_OP_Plus.Axis[0].ToGoValid THEN
        ToGo_0      := oR_Meas_OP_Plus.Axis[0].ToGo;
    END_IF
    IF oR_Meas_OP_Plus.Axis[0].LagValid THEN
        Lag_0       := oR_Meas_OP_Plus.Axis[0].Lag;
    END_IF
    Res_0           := oR_Meas_OP_Plus.Axis[0].Resolution;
END_IF
```

5.3.5 Reading axes programmed position ► R_Axis_Prog_Plus

This request returns the programmed position of all axes in a particular channel.
The data are returned with respect to the axis name in the channel.

R_Plus_005

```
PROGRAM R_Plus_005
VAR
    oR_Dim_Prog_Plus                    : R_Axis_Prog_Plus;
    Prog_X                              : LREAL;
END_VAR
```

```
oR_Dim_Prog_Plus.CNCAdr                 := ADR(NCK);
oR_Dim_Prog_Plus.byChannel              := 0;
                                // Channels numbered 0 to 7

oR_Dim_Prog_Plus(enable := xEnable);;

IF oR_Dim_Prog_Plus.uiResult = 0 THEN
    xEnable     := FALSE;
    Prog_X      := oR_Dim_Prog_Plus.lrX_Axis;
END_IF
```

5.3.6 Reading axes reference ► R_Axis_Ref_Plus

This request returns the reference value (in the machine referential) of the axes in a particular channel. The data are returned with respect to the axis name.

R_Plus_006

```
PROGRAM R_Plus_006
VAR
    oR_Axis_Ref_Plus                    : R_Axis_Ref_Plus;
    Ref_X                               : LREAL;
END_VAR
```

```
oR_Axis_Ref_Plus.CNCAdr                 := ADR(NCK);
oR_Axis_Ref_Plus.byChannel              := 0;
                                // Channels numbered 0 to 7

oR_Axis_Ref_Plus(enable := xEnable);;

IF oR_Axis_Ref_Plus.uiResult = 0 THEN
    xEnable     := FALSE;
    Ref_X       := oR_Axis_Ref_Plus.lrX_Axis;
END_IF
```

5.3.7 Reading a part program block ► R_Block_Plus

This request returns a string corresponding to a particular part program block.

Input data will be the memory zone where the part program is searched for, the part program number (times 10 and including the index), the block number and the offset (in number of blocks) relative to this number. In case of non numbered part program, the number 0 corresponds to the first line of program.

R_Plus_007

```
PROGRAM R_Plus_007
VAR
    oR_Block_Plus                   : R_Block_Plus;
    Block                           : STRING;
END_VAR
```

```
oR_Block_Plus.CNCAdr                    := ADR(NCK);
oR_Block_Plus.byZone                    := 0;
                                    // memory zone 0
oR_Block_Plus.diPPnumber                := 1560;
oR_Block_Plus.woBlocknber               := 1080;
oR_Block_Plus.woOffset                  := 0;

oR_Block_Plus(enable := xEnable);

IF oR_Block_Plus.uiResult = 0 THEN
    xEnable     := FALSE;
    Block       := oR_Block_Plus.stBlock;
END_IF
```

5.3.8 Reading an axis measure correction (E952xx) ► R_Corr_Dyn_Plus

This request returns the value of parameter E952xx (measure correction) of a particular axis.

R_Plus_008

```
1   PROGRAM R_Plus_008
2   VAR
3       oR_Corr_Dyn_Plus                : R_Corr_Dyn_Plus;
4       Corr_X                          : LREAL;
5   END_VAR
6
```

```
1   oR_Corr_Dyn_Plus.CNCAdr                 := ADR(NCK);
2   oR_Corr_Dyn_Plus.byAxisNo               := 0;
3                                           // axis 0
4
5   oR_Corr_Dyn_Plus(enable := xEnable);
6
7   IF oR_Corr_Dyn_Plus.uiResult = 0 THEN
8       xEnable     := FALSE;
9       corr_X      := oR_Corr_Dyn_Plus.lrCorrDyn;
10  END_IF
11
```

5.3.9 Reading the part program block in execution ► R_Curr_Block_Plus

This request returns a string corresponding to the block in execution. As the answer can take several steps a *Read_Complete* status flag is part of the answer.

Return code of *byReadStatus*:

- 0x06 : Block reading incomplete : Increment *woSequence* to get the rest of the block
- 0x0F : Block read finished *woSequence* = 0 to read new block·
- 0x19 : Sequencing error.

The first sequence number should be 0.
If the block is not read entirely then read the next part after incrementing the sequence number: *woSequence* = *woSequence +1* and so on until *byReadStatus* = 0x0F.

R_Plus_009

```
PROGRAM R_Plus_009
VAR
    oR_Curr_block_Plus              : R_Curr_Block_Plus;
    Block                           : STRING;
    Read_Complete                   : BOOL;
END_VAR
```

```
oR_Curr_block_Plus.CNCAdr                   := ADR(NCK);
oR_Curr_block_Plus.woChannel                := 0;
oR_Curr_block_Plus.woSequence               := 0;
oR_Curr_block_Plus.

oR_Curr_block_Plus(enable := xEnable);

IF oR_Curr_block_Plus.uiResult = 0 THEN
    xEnable     := FALSE;
    Block       := oR_Curr_block_Plus.stCurrBlock;
END_IF

IF oR_Curr_block_Plus.byReadStatus  = 16#0F THEN
    Read_Complete   := TRUE;
    // If NOT incrememnt woSequence and read again
END_IF
```

5.3.10 Reading current Dat1 value ► R_Dat1_Pref_Plus

This request returns the current DAT1 set of values for a particular channel.
The data are returned with respect to the axes names.

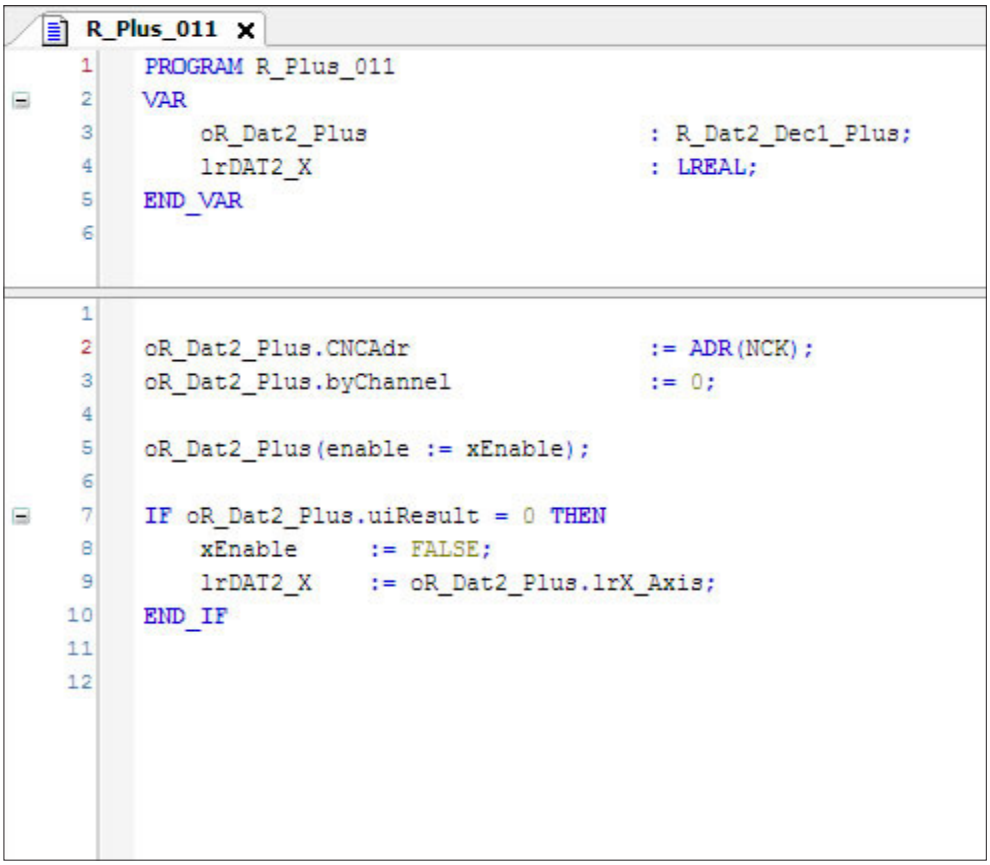
R_Plus_010

```
PROGRAM R_Plus_010
VAR
    oR_Dat1_Plus                    : R_Dat1_Pref_Plus;
    lrDAT1_X                        : LREAL;
END_VAR
```

```

oR_Dat1_Plus.CNCAdr                 := ADR(NCK);
oR_Dat1_Plus.byChannel              := 0;

oR_Dat1_Plus(enable := xEnable);

IF oR_Dat1_Plus.uiResult = 0 THEN
    xEnable     := FALSE;
    lrDAT1_X    := oR_Dat1_Plus.lrX_Axis;
END_IF
```

5.3.11 Reading current Dat2 value ► R_Dat2_Dec1_Plus

This request returns the current DAT2 set of values for a particular channel.
The data are returned with respect to the axes names.



5.3.12 Reading current Dat3 value ► R_Dat3_Dec3_Plus

This request returns the current DAT3 set of values for a particular channel. The data are returned with respect to the axes names.

NB : It is possible to enter a value for any of the linear axes of a channel, this request does not consider which axes are currently involved in DEC3. As well it does not change the current affectation.

R_Plus_012

```
1   PROGRAM R_Plus_012
2   VAR
3       oR_Dat3_Plus                    : R_Dat3_Dec3_Plus;
4       lrDAT3_X                        : LREAL;
5   END_VAR
6
```

```
1
2   oR_Dat3_Plus.CNCAdr                 := ADR(NCK);
3   oR_Dat3_Plus.byChannel              := 0;
4
5   oR_Dat3_Plus(enable := xEnable);
6
7   IF oR_Dat3_Plus.uiResult = 0 THEN
8       xEnable     := FALSE;
9       lrDAT3_X    := oR_Dat3_Plus.lrX_Axis;
10  END_IF
11
12
```

5.3.13 Reading Drives Test points values ► R_Drive_TPValue_V2_Plus

This request returns up to eight test point values of one or several drives.

The test points configuration is defined in the drive parameters V130 to V133 and the values read are expressed in physical units.

R_Plus_020

```
PROGRAM R_Plus_020
VAR
    oR_TP_Value                 : R_Drive_TPValue_V2_Plus;
    Drives                      : ARRAY[1..8] OF BYTE
                                    := [0,0,0,0,1,1,1,1];
    Points                      : ARRAY[1..8] OF BYTE
                                    := [0,1,2,3,0,1,2,3];
    Values                      : ARRAY[1..8] OF LREAL;
END_VAR
```

```

oR_TP_Value.CNCAdr              := ADR(NCK);
oR_TP_Value.DriveAddresse       := ADR(Drives);
oR_TP_Value.DriveTP             := ADR(Points);
oR_TP_Value.Number              := 4;


oR_TP_Value(enable := xEnable);

IF oR_TP_Value.uiResult = 0 THEN
    xEnable     := FALSE;
    Values[1]   := oR_TP_Value.Values[0];
    Values[2]   := oR_TP_Value.Values[1];
    Values[3]   := oR_TP_Value.Values[2];
    Values[4]   := oR_TP_Value.Values[3];
END_IF

```

5.3.14 Reading a set of E8xyyy parameters ► R_E8xyyy_Plus

This request returns a set of 10 E8…. parameters from the series E80, E81, E82 E83 or E84.

R_Plus_030

```
PROGRAM R_Plus_030
VAR
    oR_E8xyyy                       : R_E8xyyy_Plus;
    Para                            : UDINT     := 80000;
    Values                          : ARRAY[1..10] OF LREAL;
END_VAR
```

```
oR_E8xyyy.CNCAdr                    := ADR(NCK);
oR_E8xyyy.udE8xyyyNo                := Para;

oR_E8xyyy(enable := xEnable);

IF oR_E8xyyy.uiResult = 0 THEN
    xEnable     := FALSE;
    Values[1]   := oR_E8xyyy.lrE8_n0;
    Values[2]   := oR_E8xyyy.lrE8_n1;
    Values[3]   := oR_E8xyyy.lrE8_n2;
    Values[4]   := oR_E8xyyy.lrE8_n3;
    Values[5]   := oR_E8xyyy.lrE8_n4;
    Values[6]   := oR_E8xyyy.lrE8_n5;
    Values[7]   := oR_E8xyyy.lrE8_n6;
    Values[8]   := oR_E8xyyy.lrE8_n7;
    Values[9]   := oR_E8xyyy.lrE8_n8;
    Values[10]  := oR_E8xyyy.lrE8_n9;

END_IF
```

5.3.15 Reading a program parameter ► R_E_Param_Plus

This request returns the value of a particular E parameter as well as the type of parameter and the possibility of writing it or not.

R_Plus_031

```
PROGRAM R_Plus_031
VAR
    oR_E_Param                  : R_E_Param_Plus;
    Parameter                   : UDINT     := 50000;
    Para_Type                   : BYTE;
    Writable                    : BOOL;
    pP1                         : POINTER TO USINT;
    Type_1                      : USINT;
    pP8                         : POINTER TO BOOL;
    Type_8                      : BOOL;
END_VAR
```

```

oR_E_Param.CNCAdr               := ADR(NCK);
oR_E_Param.byChannel            := 0;
oR_E_Param.udE_number           := Parameter;

oR_E_Param(enable := xEnable);

IF oR_E_Param.uiResult = 0 THEN
    xEnable     := FALSE;
    Para_Type   := oR_E_Param.dataType;
    Writable    := oR_E_Param.writeable;

    CASE Para_Type OF
                    // Example for parameters type 1 and 8
        1:  pP1     :=  oR_E_Param.ptValue;
            Type_1  :=  pP1^;

        8: pP8      :=  oR_E_Param.ptValue;
           Type_8   := pP8^;
        ELSE
          ;     // To complete for other types
    END_CASE
END_IF

```

5.3.16 Reading the Interpolation data ► R_Interpo_Plus

This request returns current interpolation data of a particular channel. The data are curvilinear distance to go, the feedrate as well as the override value.

R_Plus_040

```
PROGRAM R_Plus_040
VAR
    oR_Interpo                  : R_Interpo_Plus;
    lr_Distance                 : LREAL;
    lr_Feed                     : LREAL;
    lr_Override                 : LREAL;
END_VAR
```

```
oR_Interpo.CNCAdr               := ADR(NCK);
oR_Interpo.byChannel            := 0;

oR_Interpo(enable := xEnable);

IF oR_Interpo.uiResult = 0 THEN
    xEnable         := FALSE;
    lr_Distance     := oR_Interpo.lrDistance;
    lr_Feed         := oR_Interpo.lrFeedProg;
    lr_Override     := oR_Interpo.lrOverride;
END_IF
```

5.3.17 Reading machine origin values ► R_Mach_Orig_Plus

This request returns the machine origin offset values (equivalent to parameter P16) according to the axis physical address. The values are returned for three consecutive addresses.

R_Plus_041

```
PROGRAM R_Plus_041
VAR
    oR_Mach_Origin                  : R_Mach_Orig_Plus;
    lr_Axe1                         : LREAL;
    lr_Axe2                         : LREAL;
    lr_axe3                         : LREAL;
END_VAR
```

```

oR_Mach_Origin.CNCAdr                   := ADR(NCK);
oR_Mach_Origin.byAxisNo                 := 1;

oR_Mach_Origin(enable := xEnable);

IF oR_Mach_Origin.uiResult = 0 THEN
    xEnable         := FALSE;
    lr_Axe1         := oR_Mach_Origin.lrAxe_n1;
    lr_Axe2         := oR_Mach_Origin.lrAxe_n2;
    lr_Axe3         := oR_Mach_Origin.lrAxe_n3;
END_IF

```

5.3.18 Reading Maximum Dynamic Travel values ► R_Max_Dyn_Travel_Plus

This request returns the maximum dynamic travels of the axes in a particular channel. The values are returned in reference to the axes names.

R_Plus_042

```
PROGRAM R_Plus_042
VAR
    oR_Max_Dyn_Travel               : R_Max_Dyn_Travel_Plus;
    lr_MaxDyn_A                     : LREAL;
    lr_MaxDyn_B                     : LREAL;
    lr_MaxDyn_C                     : LREAL;
    lr_MaxDyn_U                     : LREAL;
    lr_MaxDyn_V                     : LREAL;
    lr_MaxDyn_W                     : LREAL;
    lr_MaxDyn_X                     : LREAL;
    lr_MaxDyn_Y                     : LREAL;
    lr_MaxDyn_Z                     : LREAL;
END_VAR
```

```

oR_Max_Dyn_Travel.CNCAdr                    := ADR(NCK);
oR_Max_Dyn_Travel.byChannel                 := 0;

oR_Max_Dyn_Travel(enable := xEnable);

IF oR_Max_Dyn_Travel.uiResult = 0 THEN
    xEnable         := FALSE;
    lr_MaxDyn_A                     := oR_Max_Dyn_Travel.lrA_Axis;
    lr_MaxDyn_B                     := oR_Max_Dyn_Travel.lrB_Axis;
    lr_MaxDyn_C                     := oR_Max_Dyn_Travel.lrC_Axis;;
    lr_MaxDyn_U                     := oR_Max_Dyn_Travel.lrU_Axis;
    lr_MaxDyn_V                     := oR_Max_Dyn_Travel.lrV_Axis;;
    lr_MaxDyn_W                     := oR_Max_Dyn_Travel.lrW_Axis;;
    lr_MaxDyn_X                     := oR_Max_Dyn_Travel.lrX_Axis;;
    lr_MaxDyn_Y                     := oR_Max_Dyn_Travel.lrY_Axis;;
    lr_MaxDyn_Z                     := oR_Max_Dyn_Travel.lrZ_Axis;;
END_IF
```

5.3.19 Reading Minimum Dynamic Travel values ► R_Min_Dyn_Travel_Plus

This request returns the minimum dynamic travels of the axes in a particular channel. The values are returned in reference to the axes names.

R_Plus_043

```
PROGRAM R_Plus_043
VAR
    oR_Min_Dyn_Travel               : R_Min_Dyn_Travel_Plus;
    lr_MinDyn_A                     : LREAL;
    lr_MinDyn_B                     : LREAL;
    lr_MinDyn_C                     : LREAL;
    lr_MinDyn_U                     : LREAL;
    lr_MinDyn_V                     : LREAL;
    lr_MinDyn_W                     : LREAL;
    lr_MinDyn_X                     : LREAL;
    lr_MinDyn_Y                     : LREAL;
    lr_MinDyn_Z                     : LREAL;
END_VAR
```

```
oR_Min_Dyn_Travel.CNCAdr                    := ADR(NCK);
oR_Min_Dyn_Travel.byChannel                 := 0;

oR_Min_Dyn_Travel(enable := xEnable);

IF oR_Min_Dyn_Travel.uiResult = 0 THEN
    xEnable         := FALSE;
    lr_MinDyn_A                     := oR_Min_Dyn_Travel.lrA_Axis;
    lr_MinDyn_B                     := oR_Min_Dyn_Travel.lrB_Axis;
    lr_MinDyn_C                     := oR_Min_Dyn_Travel.lrC_Axis;;
    lr_MinDyn_U                     := oR_Min_Dyn_Travel.lrU_Axis;
    lr_MinDyn_V                     := oR_Min_Dyn_Travel.lrV_Axis;;
    lr_MinDyn_W                     := oR_Min_Dyn_Travel.lrW_Axis;;
    lr_MinDyn_X                     := oR_Min_Dyn_Travel.lrX_Axis;;
    lr_MinDyn_Y                     := oR_Min_Dyn_Travel.lrY_Axis;;
    lr_MinDyn_Z                     := oR_Min_Dyn_Travel.lrZ_Axis;;
END_IF
```

5.3.20 Reading Maximum Static Travel values ► R_Max_Stat_Travel_Plus

This request returns the maximum static travels of axes. The values are given for three consecutives physical addresses.

R_Plus_044

```
1  PROGRAM R_Plus_044
2  VAR
3      oR_Max_Stat_Travel              : R_Max_Stat_Travel_Plus;
4      lr_MaxStat_0                    : LREAL;
5      lr_MaxStat_1                    : LREAL;
6      lr_MaxStat_2                    : LREAL;
7  END_VAR
8
```

```
1
2  oR_Max_Stat_Travel.CNCAdr                   := ADR(NCK);
3  oR_Max_Stat_Travel.byAxisNo                 := 0;
4
5  oR_Max_Stat_Travel(enable := xEnable);
6
7  IF oR_Max_Stat_Travel.uiResult = 0 THEN
8      xEnable              := FALSE;
9      lr_MaxStat_0         := oR_Max_Stat_Travel.lrAxe_n1;
10     lr_MaxStat_1         := oR_Max_Stat_Travel.lrAxe_n2;
11     lr_MaxStat_2         := oR_Max_Stat_Travel.lrAxe_n3;;
12 END_IF
13
```

5.3.21 Reading Minimum Static Travel values ► R_Min_Stat_Travel_Plus

This request returns the maximum static travels of axes. The values are given for three consecutives physical addresses.

R_Plus_045

```
PROGRAM R_Plus_045
VAR
    oR_Min_Stat_Travel              : R_Min_Stat_Travel_Plus;
    lr_MinStat_0                    : LREAL;
    lr_MinStat_1                    : LREAL;
    lr_MinStat_2                    : LREAL;
END_VAR
```

```

oR_Min_Stat_Travel.CNCAdr                   := ADR(NCK);
oR_Min_Stat_Travel.byAxisNo                 := 0;

oR_Min_Stat_Travel(enable := xEnable);

IF oR_Min_Stat_Travel.uiResult = 0 THEN
    xEnable             := FALSE;
    lr_MinStat_0        := oR_Min_Stat_Travel.lrAxe_n1;
    lr_MinStat_1        := oR_Min_Stat_Travel.lrAxe_n2;
    lr_MinStat_2        := oR_Min_Stat_Travel.lrAxe_n3;;
END_IF

```

5.3.22 Reading NCK detected error ► R_Nck_Error_Plus

This request returns the information about an NCK detected error. The data are the block number and line number where the error has occurred as well as the error number and error extension number.

R_Plus_046

```
PROGRAM R_Plus_046
VAR
    oR_NckError                 : R_Nck_Error_Plus;
    ud_Block                    : UDINT;
    ud_Line                     : UDINT;
    wo_Error                    : WORD;
    wo_Extension                : WORD;
END_VAR
```

```
oR_NckError.CNCAdr              := ADR(NCK);
oR_NckError.byChannel           := 0;

oR_NckError(enable := xEnable);

IF oR_NckError.uiResult = 0 THEN
    xEnable             := FALSE;
    ud_Block            := oR_NckError.udBlockNumber;
    ud_Line             := oR_NckError.udLineNumber;
    wo_Error            := oR_NckError.woErrorNumber;
    wo_Extension        := oR_NckError.woExtendedErrorNumber;
END_IF
```

5.3.23 Reading Part program execution tree ► R_Program_Tree _Plus

This request returns information about the part program in execution. The information include block number as well as line number, the current subroutine number with number of times it has been called as well as all the chain of calling programs (up to eight nesting level).

R_Plus_047

```
PROGRAM R_Plus_047
VAR
    oR_Prog_Tree                    : R_Program_Tree_Plus;
    Main                            : UDINT;
    SubPrgs                         : ARRAY [0..7] OF UDINT;
    Calls                           : BYTE;
    Block                           : UDINT;
    Line                            : UDINT;
END_VAR
```

```
oR_Prog_Tree.CNCAdr                 := ADR(NCK);
oR_Prog_Tree.byChannel              := 0;

oR_Prog_Tree(enable := xEnable);

IF oR_Prog_Tree.uiResult = 0 THEN
    xEnable             := FALSE;
    Main                := oR_Prog_Tree.ProgramData.MainProgram;
    SubPrgs             := oR_Prog_Tree.ProgramData.Subprograms;
    Calls               := oR_Prog_Tree.ProgramData.NumberOfSubprogramCalls;
    Block               := oR_Prog_Tree.ProgramData.BlockNumber;
    Line                := oR_Prog_Tree.ProgramData.LineNumber;
END_IF
```

5.3.24 Reading Spindle information ► R_Spindle_Complete_Plus

This request returns all relevant spindle information excluding the range currently active.
The full list of data can be seen in the example below.

R_Plus_048

```
PROGRAM R_Plus_048
VAR
    oR_Spindle                  : R_Spindle_Complete_Plus;
    bNumberOfSpindles           : BYTE;
    bChannel                    : BYTE;
    bFormat                     : BYTE;
    bLogicalAddress             : BYTE;
    bRank                       : BYTE;  // Main or Auxiliary
    xM19                        : BOOL;
    xM3                         : BOOL;
    xM4                         : BOOL;
    xM5                         : BOOL;
    xM62                        : BOOL;
    xM66                        : BOOL;
    uiOverride                  : UINT;
    bPhysicalAddress            : BYTE;
    lrProgSpeed                 : LREAL;
    lrRealSpeed                 : LREAL;
```

```

oR_Spindle.CNCAdr               := ADR(NCK);

oR_Spindle(enable := xEnable);

IF oR_Spindle.uiResult = 0 THEN
    xEnable                     := FALSE;
    bNumberOfSpindles           := oR_Spindle.nrOfSpindles;
    bChannel                    := oR_Spindle.SpindleData[0].Channel;
    bFormat                     := oR_Spindle.SpindleData[0].Format;
    bLogicalAddress             := oR_Spindle.SpindleData[0].LogicalIndexInChannel;
    bRank                       := oR_Spindle.SpindleData[0].LogicalIndexInSystem;
    xM19                        := oR_Spindle.SpindleData[0].M19;
    xM3                         := oR_Spindle.SpindleData[0].M3;
    xM4                         := oR_Spindle.SpindleData[0].M4;
    xM5                         := oR_Spindle.SpindleData[0].M5;
    xM62                        := oR_Spindle.SpindleData[0].M62;
    xM66                        := oR_Spindle.SpindleData[0].M66;
    uiOverride                  := oR_Spindle.SpindleData[0].Override;
    bPhysicalAddress            := oR_Spindle.SpindleData[0].PhysicalAddress;
    lrProgSpeed                 := oR_Spindle.SpindleData[0].SpeedProgrammed;
    lrRealSpeed                 := oR_Spindle.SpindleData[0].SpeedReal;
END_IF

```

5.3.25 Reading Tool offsets data ► R_Tool_Plus

This request returns all data about a particular tool offset, excluding the H value that can be obtained via the *Read_E_Param* request.

R_Plus_049

```
PROGRAM R_Plus_049
VAR
    oR_Tool                         : R_Tool_Plus;
    ToolPar1                        : LREAL;
    ToolPar2                        : LREAL;
    ToolPar3                        : LREAL;
    ToolPar4                        : LREAL;
    ToolPar5                        : LREAL;
    ToolPar6                        : DINT;
    ToolPar7                        : DINT;
END_VAR
```

```
oR_Tool.CNCAdr                  := ADR(NCK);
oR_Tool.byCorrNo                := 1;       // Tool offset number

oR_Tool(enable := xEnable);

IF oR_Tool.uiResult = 0 THEN
    xEnable                     := FALSE;

    ToolPar1                    := oR_Tool.lrE50xxx;
    ToolPar2                    := oR_Tool.lrE51xxx;
    ToolPar3                    := oR_Tool.lrE52xxx;
    ToolPar4                    := oR_Tool.lrE53xxx;
    ToolPar5                    := oR_Tool.lrE54xxx;
    ToolPar6                    := oR_Tool.diE55xxx;
    ToolPar7                    := oR_Tool.diE57xxx;
END_IF
```

Page deliberately empty.

Page deliberately empty.

Summary

Page deliberately empty.

6 Write requests

6.1 General requests

6.1.1 Downloading a part program ► W_Download2_PP

This request replaces the older request *W_Download_PP*.

The part program to load is taken from the mass memory. Be careful to maintain the request until the end of the transfer.
A program of the same number already in memory must be erased first before the download.

The flag *boIgnoreExisting* is used to authorize or cancel the transfer if the program to download already exists in one of the protected zones (1 to 3). The transfer is cancelled when the flag is False.

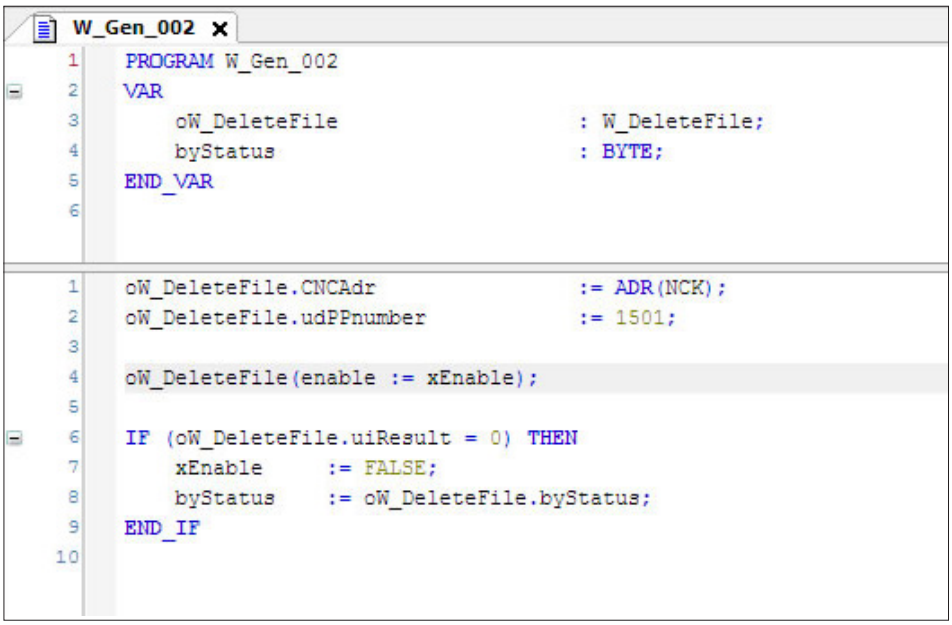
W_Gen_001

```
PROGRAM W_Gen_001
VAR
    oW_Download                 : W_Download2_PP;
END_VAR
```

```
oW_Download.CNCAddress              := ADR(NCK);
oW_Download.stFileName              := 'C:\Temp\DownloadTest.xpi';
oW_Download.udPPnumber              := 10001;
oW_Download.boIgnoreExisting        := TRUE;

IF xEnable THEN
    oW_Download();
END_IF

IF (oW_Download.uiResult    = 0) AND (oW_Download.diError = 0) THEN
    xEnable := FALSE;
END_IF
```

6.1.2 Erasing an existing part program ► W_Delete_File

This request is intended to erase a part program in NC memory generally to permit downloading a program with the same number.



6.1.3 Writing drive parameters in Flash memory ► W_Drive_Flash

This request is used to make permanent the changes that have been made by *W_Drive_Param*.

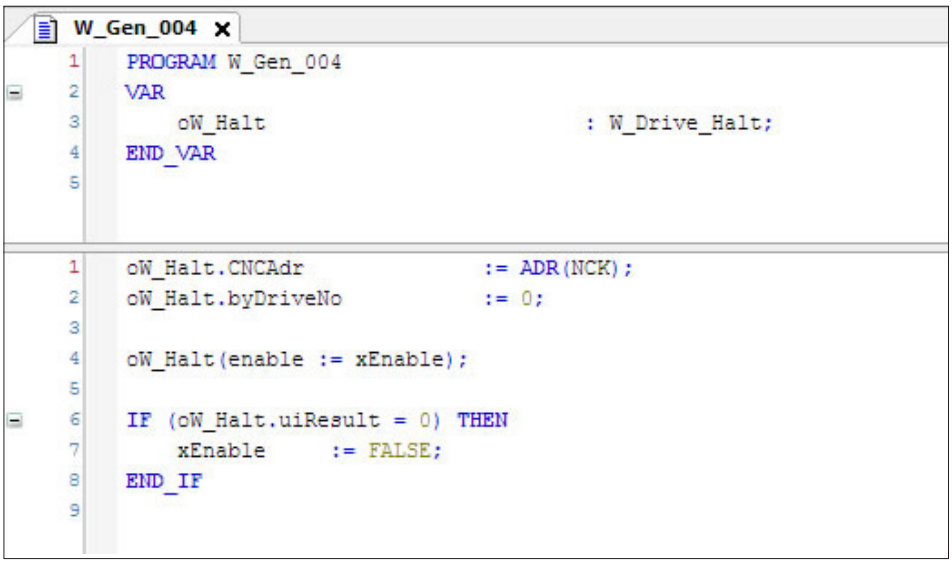
W_Gen_003

```
1  PROGRAM W_Gen_003
2  VAR
3      oW_Flash                        : W_Drive_Flash;
4  END_VAR
5
```

```
1  oW_Flash.CNCAdr             := ADR(NCK);
2  oW_Flash.byDriveNo          := 0;
3
4  oW_Flash(enable := xEnable);
5
6  IF (oW_Flash.uiResult = 0) THEN
7      xEnable     := FALSE;
8  END_IF
9
```

6.1.4 Setting the drive in Halt ► W_Drive_Halt

Some drive parameters can only be changed when the drive is in Halt. Before launching this request the programmer must take all precautions to prevent any incident or accident.



6.1.5 Writing drive parameters ► W_Drive_Param

Some drive parameters may only be changed with Torque Off or Drive in Halt. Moreover the new values must be entered in both Physical values and Drive internal values.

The table in annex gives the relevant information.

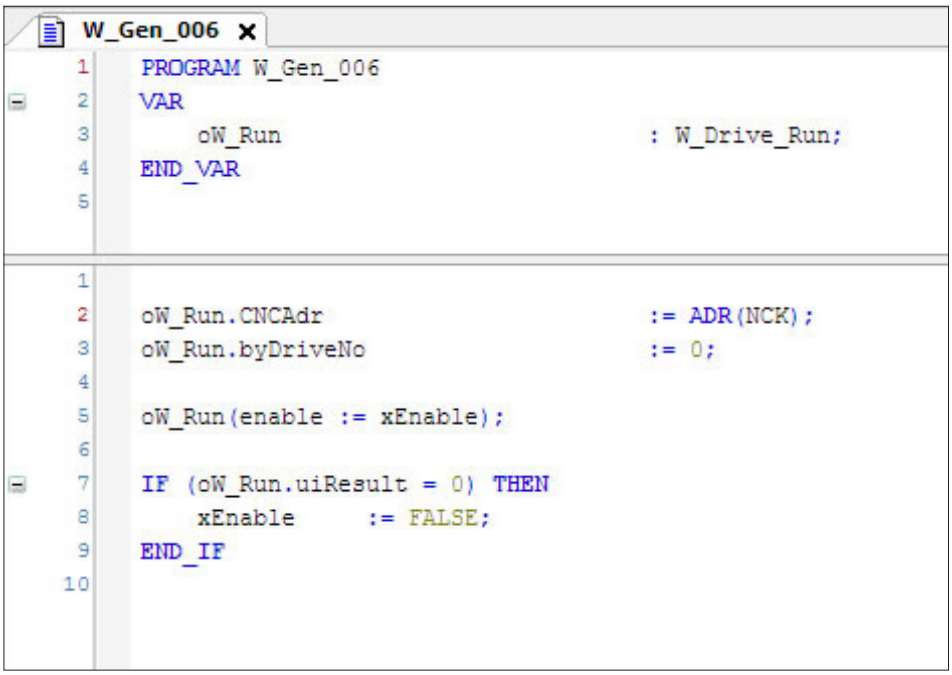
W_Gen_005

```
1  PROGRAM W_Gen_005
2  VAR
3      oW_Drive_Param                  : W_Drive_Param;
4  END_VAR
5
```

```
1
2  oW_Drive_Param.CNCAdr               := ADR(NCK);
3  oW_Drive_Param.byDriveAdd           := 0;
4
5  oW_Drive_Param.diPinValue           := 0;
6  oW_Drive_Param.diPphValue           := 0;
7  oW_Drive_Param.dwPphValue           := 0;
8  oW_Drive_Param.rePphValue           := 0;
9  oW_Drive_Param.woPnumber            := 0;
10
11 oW_Drive_Param(enable := xEnable);
12
13 IF (oW_Drive_Param.uiResult = 0) THEN
14     xEnable      := FALSE;
15 END_IF
16
```

6.1.6 Setting the drive in run ► W_Drive_Run

This request is intended to restart a drive that was put in Halt.
The programmer must take all precautions to prevent any incident or accident.

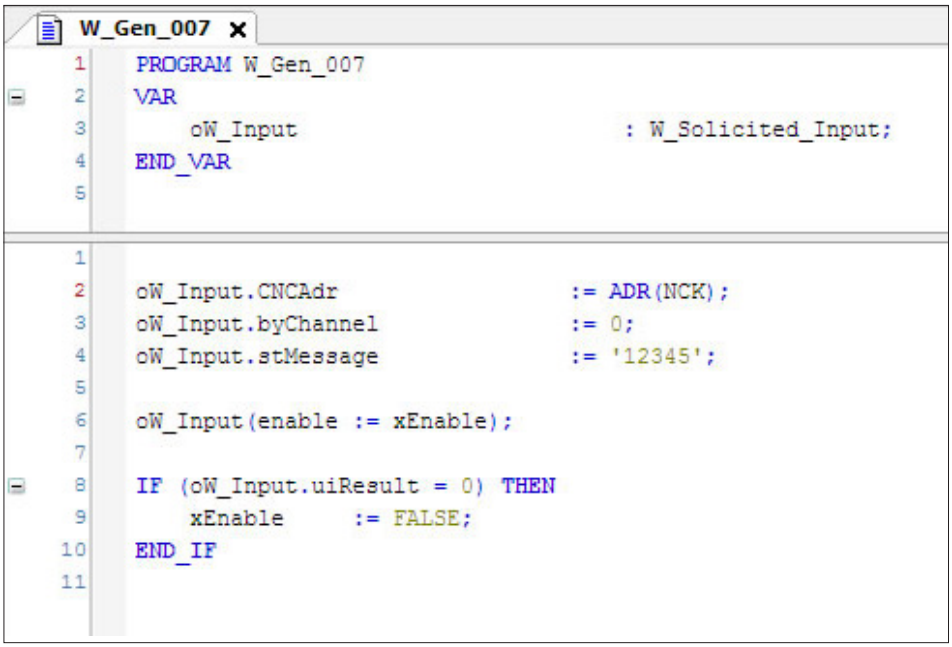


6.1.7 Answering to a solicited input ► W_Solicited_Input

This request allows for sending a text in the Operator's dialog line or answering a part program request *($0)*.

Writing *FF* in the channel number accesses the currently displayed channel.
In Sequence number search and common groups, the channel number is not used.

Important: In *MDI* the request *R_Solicited_Input* should be sent first.



6.1.8 Writing tool H data ► W_Tool_H

This request will enter the H value of up to 10 tool offsets. The data are NCK in internal units.

W_Gen_008

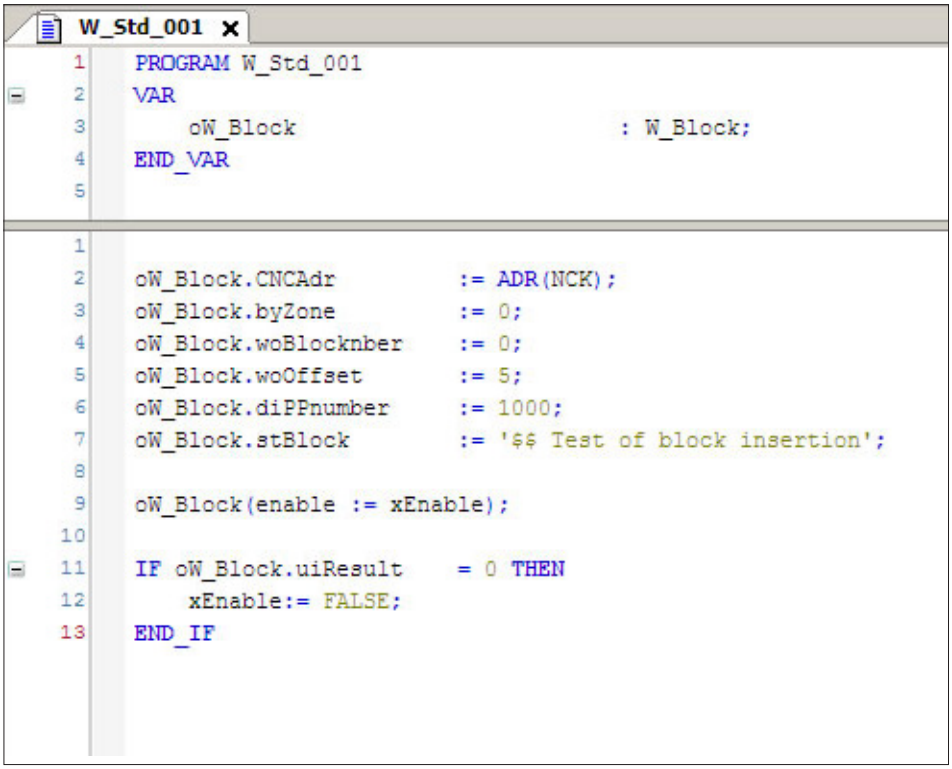
```
PROGRAM W_Gen_008
VAR
    oW_Tool_H                          : W_Tool_H;
END_VAR
```

```
oW_Tool_H.CNCAdr                   := ADR(NCK);
oW_Tool_H.byH_No                   := 1;
oW_Tool_H.diE56_n0                 := 100;
oW_Tool_H.diE56_n1                 := 200;
oW_Tool_H.diE56_n2                 := 300;
oW_Tool_H.diE56_n3                 := 400;
oW_Tool_H.diE56_n4                 := 500;
oW_Tool_H.diE56_n5                 := 600;
oW_Tool_H.diE56_n6                 := 700;
oW_Tool_H.diE56_n7                 := 800;
oW_Tool_H.diE56_n8                 := 900;
oW_Tool_H.diE56_n9                 := 999;

oW_Tool_H(enable := xEnable);

IF (oW_Tool_H.uiResult = 0) THEN
    xEnable     := FALSE;
END_IF
```

6.2 Flexium requests

6.2.1 Writing a part program block ► W_Block

This request allows for writing an NC block. It does not insert a new block just replaces the existing values.



6.2.2 Writing an axis measure correction ► W_Corr_Dyn

This request allows for writing into the parameter E952xx that can be only reached by Dynamic operator or ENA.

W_Std_002

```
1  PROGRAM W_Std_002
2  VAR
3      oW_Corr_Dyn                    : W_Corr_Dyn;   // Write into E952xx
4  END_VAR
5
```

```
1
2  oW_Corr_Dyn.CNCAdr          := ADR(NCK);
3  oW_Corr_Dyn.byAxisNo        := 0;
4  oW_Corr_Dyn.diCorrDyn       := 123;
5
6  oW_Corr_Dyn(enable  := xEnable);
7
8  IF oW_Corr_Dyn.uiResult = 0 THEN
9      xEnable:= FALSE;
10 END_IF
```

6.2.3 Writing DAT1 values ► W_Dat1_Pref

This request allows for writing the values of the current DAT1 of a particular channel.

It is strongly recommended to run this request in Reset status as the new values will not be taken into account immediately to prevent an incident but after an M0 / M2 or a block holding preparation for example M1 (validated or not) or a writing operation into an E parameter.

As this request writes the values for all axes of the channel, be sure to fill in the data for all these axes. Failure to do this will enter erroneous values for the non defined addresses.

W_Std_003

```
PROGRAM W_Std_003
VAR
    W_Dat1                          : W_Dat1_Pref;
END_VAR
```

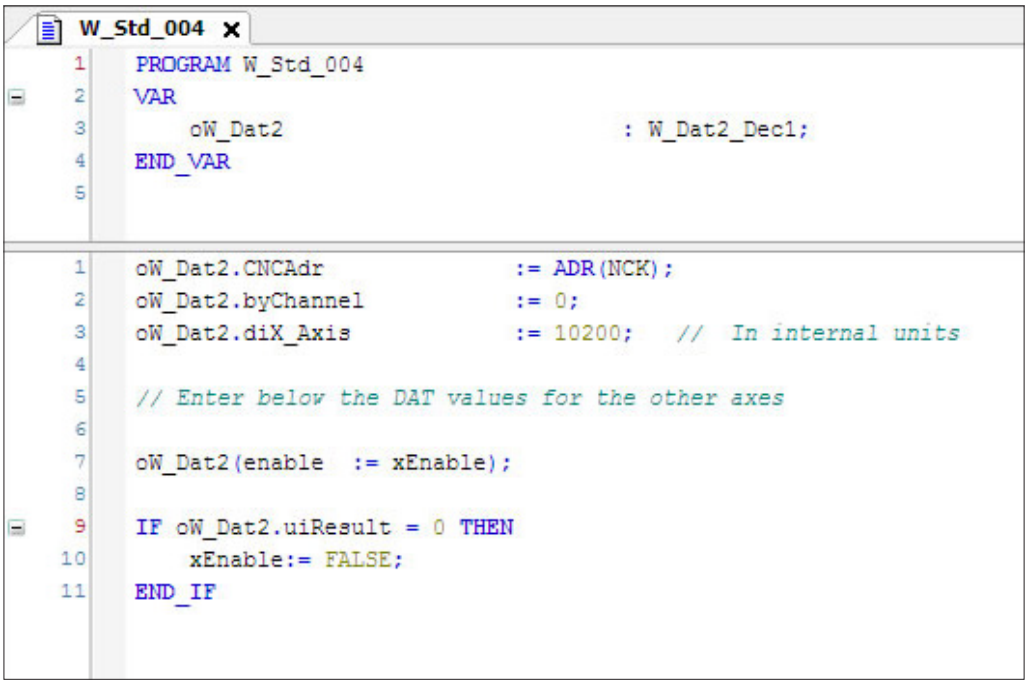
```
W_Dat1.CNCAdr                 := ADR(NCK);
W_Dat1.byChannel              := 0;
W_Dat1.diX_Axis               := 500000;  //  In internal units
// Enter below the DAT values for the other axes

W_Dat1(enable    := xEnable);

IF W_Dat1.uiResult  = 0 THEN
    xEnable:= FALSE;
END_IF
```

6.2.4 Writing DAT2 values ► W_Dat2_Dec1

This request allows for writing the values of the current DAT2 of a particular channel.

It is strongly recommended to run this request in Reset status as the new values will not be taken into account immediately but at the next M0 / M1 M2 or E parameter write operation.

As this request writes the values for all axes of the channel, be sure to fill in the data for all these axes. Failure to do this will enter erroneous values for the non defined addresses.

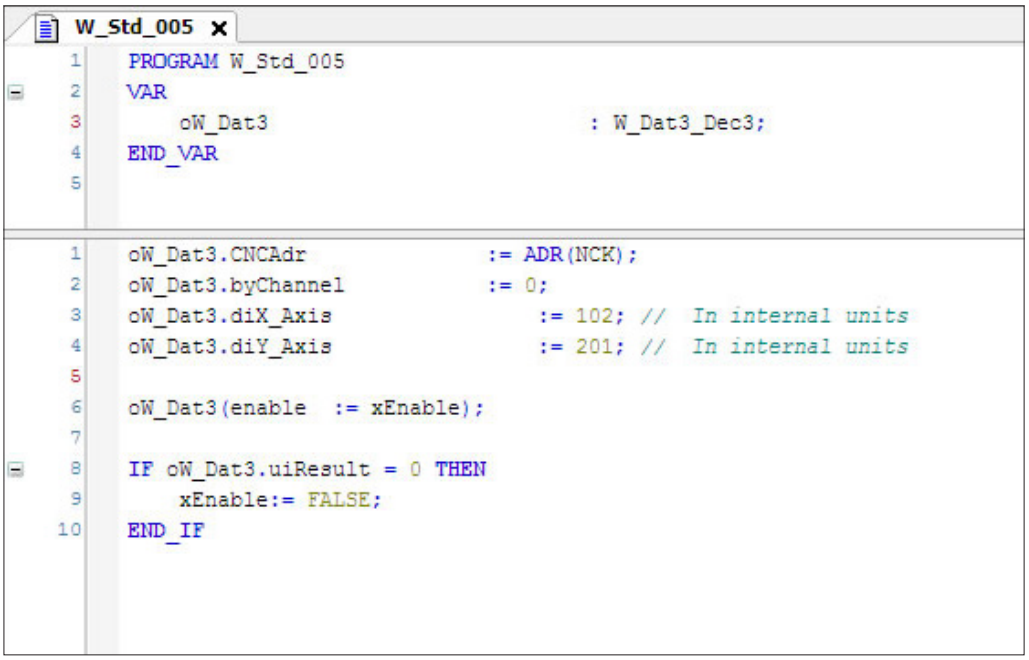


6.2.5 Writing DAT3 values ► W_Dat3_Dec3

This request allows for writing the values of the current DAT3 of a particular channel.

It is strongly recommended to run this request in Reset status as the new values will not be taken into account immediately but at the next M0 / M1 M2 or E parameter write operation.

As this request writes the values for all axes of the channel, be sure to fill in the data for all these axes. Failure to do this will enter erroneous values for the non defined addresses.



6.2.6 Writing an E8… parameter value ► W_E8xyyy

This request allows for writing one E8xyyy parameter.
The integer input argument will be written in the parameter.

W_Std_006

```
1  PROGRAM W_Std_006
2  VAR
3      oW_E8xyyy                    : W_E8xyyy;
4  END_VAR
5
```

```
1
2  oW_E8xyyy.CNCAdr             := ADR(NCK);
3  oW_E8xyyy.udE8xyyyNo         := 83004;
4  oW_E8xyyy.diE8_n0            := 1250;
5
6  oW_E8xyyy(enable    := xEnable);
7
8  IF oW_E8xyyy.uiResult   = 0 THEN
9      xEnable:= FALSE;
10 END_IF
```

6.2.7 Writing an E5 or E6 parameter value ► W_EParam_5_6

This request allows for writing one parameter of the series E5 or E6. Only the integer part of the input argument will be taken into account as internal units.

W_Std_007

```
1  PROGRAM W_Std_007
2  VAR
3      oW_E_5_6                    : W_E_Param_5_6;
4      Value                       : LREAL;
5  END_VAR
6
```

```
1
2  oW_E_5_6.CNCAdr          := ADR(NCK);
3  oW_E_5_6.byChannel       := 0;
4
5  oW_E_5_6.udE_number      := 50001;
6  oW_E_5_6.diValue_E       := 25000;
7
8  IF oW_E_5_6.uiResult     = 0 THEN
9      xEnable:= FALSE;
10 END_IF
```

6.2.8 Writing the home position (P16) ► W_Mach_Orig

This request allows for writing the home position for an axis (equivalent of P16).

Please note that this value is not written permanently in the parameter. The value indicated in the project will be taken at the next system start-up.

The request writes the values at three consecutive axes addresses. Therefore be sure to enter these three values.

W_Std_008

```
PROGRAM W_Std_008
VAR
    oW_Home                         : W_Mach_Orig;
END_VAR
```

```

oW_Home.CNCAdr          := ADR(NCK);
oW_Home.byAxisNo            := 0;
oW_Home.diAxe_n1            := 200;
oW_Home.diAxe_n2            := 100;
oW_Home.diAxe_n3            := 100;

oW_Home(enable  := xEnable);

IF oW_Home.uiResult = 0 THEN
    xEnable:= FALSE;
END_IF
```

6.2.9 Writing Maximum Dynamic Travel values ► W_Max_Dyn_Travel

This request allows for writing the maximum dynamic travels of the axes in a particular channel. The values are given in reference to the axes names.

As this request writes the values for all axes of the channel, be sure to fill in the data for all these axes. Failure to do this will enter erroneous values for the non defined addresses.

W_Std_009

```
PROGRAM W_Std_009
VAR
    oW_Max_Dyn                      : W_Max_Dyn_Travel;
END_VAR
```

```
oW_Max_Dyn.CNCAdr           := ADR(NCK);
oW_Max_Dyn.byChannel        := 0;
oW_Max_Dyn.diX_Axis         := 2000;
oW_Max_Dyn.diY_Axis         := 2000;
oW_Max_Dyn.diZ_Axis         := 2000;
oW_Max_Dyn.diA_Axis         := 2000;
oW_Max_Dyn.diB_Axis         := 2000;
oW_Max_Dyn.diC_Axis         := 2000;
oW_Max_Dyn.diU_Axis         := 2000;
oW_Max_Dyn.diV_Axis         := 2000;
oW_Max_Dyn.diW_Axis         := 2000;

oW_Max_Dyn(enable   := xEnable);

IF oW_Max_Dyn.uiResult  = 0 THEN
    xEnable:= FALSE;
END_IF
```

6.2.10 Writing Maximum Static Travel values ► W_Max_Stat_Travel

This request allows for writing the maximum static travels of axes.

The values are given for three consecutives axes addresses. Therefore be sure to enter these three values.

W_Std_010

```
PROGRAM W_Std_010
VAR
    oW_Max_Stat                    : W_Max_Stat_Travel;
END_VAR
```

```
oW_Max_Stat.CNCAdr          := ADR(NCK);
oW_Max_Stat.byAxisNo            := 0;
oW_Max_Stat.diAxe_n1            := 2000;
oW_Max_Stat.diAxe_n2            := 2000;
oW_Max_Stat.diAxe_n3            := 2000;

oW_Max_Stat(enable  := xEnable);

IF oW_Max_Stat.uiResult = 0 THEN
    xEnable:= FALSE;
END_IF
```

6.2.11 Writing Minimum Dynamic Travel values ► W_Min_Dyn_Travel

This request allows for writing the minimum dynamic travels of the axes in a particular channel. The values are given in reference to the axes names.

As this request writes the values for all axes of the channel, be sure to fill in the data for all these axes. Failure to do this will enter erroneous values for the non defined addresses.

W_Std_011

```
1   PROGRAM W_Std_011
2   VAR
3       oW_Min_Dyn                      : W_Min_Dyn_Travel;
4   END_VAR
5
```

```
1
2   oW_Min_Dyn.CNCAdr           := ADR(NCK);
3   oW_Min_Dyn.byChannel        := 0;
4   oW_Min_Dyn.diX_Axis         := 2000;
5   oW_Min_Dyn.diY_Axis         := 2000;
6   oW_Min_Dyn.diZ_Axis         := 2000;
7   oW_Min_Dyn.diA_Axis         := 2000;
8   oW_Min_Dyn.diB_Axis         := 2000;
9   oW_Min_Dyn.diC_Axis         := 2000;
10  oW_Min_Dyn.diU_Axis         := 2000;
11  oW_Min_Dyn.diV_Axis         := 2000;
12  oW_Min_Dyn.diW_Axis         := 2000;
13
14  oW_Min_Dyn(enable    := xEnable);
15
16  IF oW_Min_Dyn.uiResult  = 0 THEN
17      xEnable:= FALSE;
18  END_IF
```

6.2.12 Writing Minimum Static Travel values ► W_Min_Stat_Travel

This request allows for writing the minimum static travels of axes.

The values are given for three consecutives axes addresses. Therefore be sure to enter all these three values.

W_Std_012

```
PROGRAM W_Std_012
VAR
    oW_Min_Stat                    : W_Min_Stat_Travel;
END_VAR
```

```

oW_Min_Stat.CNCAdr          := ADR(NCK);
oW_Min_Stat.byAxisNo            := 0;
oW_Min_Stat.diAxe_n1            := 2000;
oW_Min_Stat.diAxe_n2            := 2000;
oW_Min_Stat.diAxe_n3            := 2000;

oW_Min_Stat(enable  := xEnable);

IF oW_Min_Stat.uiResult = 0 THEN
    xEnable:= FALSE;
END_IF
```

6.2.13 Writing an NCK parameter ► W_NCK_Par

This request allows for writing a word of a particular machine parameter.
The word number is given by the property *uiPoffset*.

Please refer to the annex A4 for the list of parameters that are taken into account immediately.

The change of other parameters has no effect as a restart will use the parameters value defined in the application. Be cautious when modifying a parameter as some others may depend on it.

W_Std_013

```
1  PROGRAM W_Std_013
2  VAR
3      oW_Nck_Par                    : W_Nck_Par;
4  END_VAR
5
```

```
1
2  oW_Nck_Par.CNCAdr           := ADR(NCK);
3  oW_Nck_Par.uiPnumber        := 30;
4  oW_Nck_Par.uiPoffset        := 5;
5  oW_Nck_Par.diPvalue         := 5500;
6
7  oW_Nck_Par(enable := xEnable);
8
9  IF oW_Nck_Par.uiResult  = 0 THEN
10     xEnable:= FALSE;
11 END_IF
```

6.2.14 Writing a tool offset ► W_Tool

This request allows for writing a particular tool offset.
All components are written with the exception of the H value which is given by a separate request.

W_Std_014

```
PROGRAM W_Std_014
VAR
    oW_Tool                     : W_Tool;
END_VAR
```

```

oW_Tool.CNCAdr          := ADR(NCK);
oW_Tool.byCorrNo            := 2;
oW_Tool.diE50xxx            := 100;
oW_Tool.diE51xxx            := 200;
oW_Tool.diE52xxx            := 300;
oW_Tool.diE53xxx            := 400;
oW_Tool.diE54xxx            := 500;
oW_Tool.diE55xxx            := 3;
oW_Tool.diE57xxx            := 700;

oW_Tool(enable := xEnable);

IF oW_Tool.uiResult = 0 THEN
    xEnable:= FALSE;
END_IF
```

6.3 Flexium+ requests

6.3.1 Writing a part program block ► W_Block_Plus

This request allows for writing an NC block.
It does not insert a new block just replaces the existing values.

W_Plus_001

```
PROGRAM W_Plus_001
VAR
    oW_Block                        : W_Block_Plus;
END_VAR
```

```

oW_Block.CNCAdr          := ADR(NCK);
oW_Block.byZone          := 0;
oW_Block.diBlocknber         := 0;
oW_Block.diOffset        := 5;
oW_Block.diPPnumber      := 1000;
oW_Block.stBlock             := '$$ Test of block insertion';

oW_Block(enable := xEnable);

IF oW_Block.uiResult     = 0 THEN
    xEnable:= FALSE;
END_IF
```

6.3.2 Writing an axis measure correction ► W_Corr_Dyn_Plus

This request allows for writing into the parameter E952xx that can be only reached by Dynamic operator or ENA.

W_Plus_002

```
1  PROGRAM W_Plus_002
2  VAR
3      oW_Corr_Dyn                    : W_Corr_Dyn_Plus;  // Write into E952xx
4  END_VAR
5
```

```
1
2  oW_Corr_Dyn.CNCAdr          := ADR(NCK);
3  oW_Corr_Dyn.byAxisNo            := 0;
4  oW_Corr_Dyn.lrCorrDyn       := 0.123;
5
6  oW_Corr_Dyn(enable  := xEnable);
7
8  IF oW_Corr_Dyn.uiResult = 0 THEN
9      xEnable:= FALSE;
10 END_IF
```

6.3.3 Writing DAT1 values ► W_Dat1_Pref_Plus

This request allows for writing the values of the current DAT1 of a particular channel.

It is strongly recommended to run this request in Reset status as the new values will not be taken into account immediately to prevent an incident but after an M0 / M2 or a block holding preparation for example M1 (validated or not) or a writing operation into an E parameter.

As this request write the values for all axes of the channel be sure to fill-in the data for all these axes.

W_Plus_003

```
1  PROGRAM W_Plus_003
2  VAR
3      W_Dat1                          : W_Dat1_Pref_Plus;
4  END_VAR
5
```

```
1  W_Dat1.CNCAdr                 := ADR(NCK);
2  W_Dat1.byChannel              := 0;
3  W_Dat1.lrX_Axis               := 102; //  In internal units
4  // You can also enter DAT values for the other axes
5
6  W_Dat1(enable    := xEnable);
7
8  IF W_Dat1.uiResult  = 0 THEN
9      xEnable:= FALSE;
10 END_IF
```

6.3.4 Writing DAT2 values ► W_Dat2_Dec1_Plus

This request allows for writing the values of the current DAT2 of a particular channel.

It is strongly recommended to run this request in Reset status as the new values will not be taken into account immediately to prevent an incident but after an M0 / M2 or a block holding preparation for example M1 (validated or not) or a writing operation into an E parameter.

As this request write the values for all axes of the channel be sure to fill-in the data for all these axes.

W_Plus_004

```
PROGRAM W_Plus_004
VAR
    oW_Dat2                         : W_Dat2_Dec1_Plus;
END_VAR
```

```
oW_Dat2.CNCAdr              := ADR(NCK);
oW_Dat2.byChannel           := 0;
oW_Dat2.lrX_Axis                := 102; //  In internal units

// Enter below the DAT values for the other axes

oW_Dat2(enable  := xEnable);

IF oW_Dat2.uiResult = 0 THEN
    xEnable:= FALSE;
END_IF
```

6.3.5 Writing DAT3 values ► W_Dat3_Dec3_Plus

This request allows for writing the values of the current DAT3 of a particular channel.

It is strongly recommended to run this request in Reset status as the new values will not be taken into account immediately to prevent an incident but after an M0 / M2 or a block holding preparation for example M1 (validated or not) or a writing operation into an E parameter.

As this request write the values for all axes of the channel be sure to fill-in the data for all these axes.

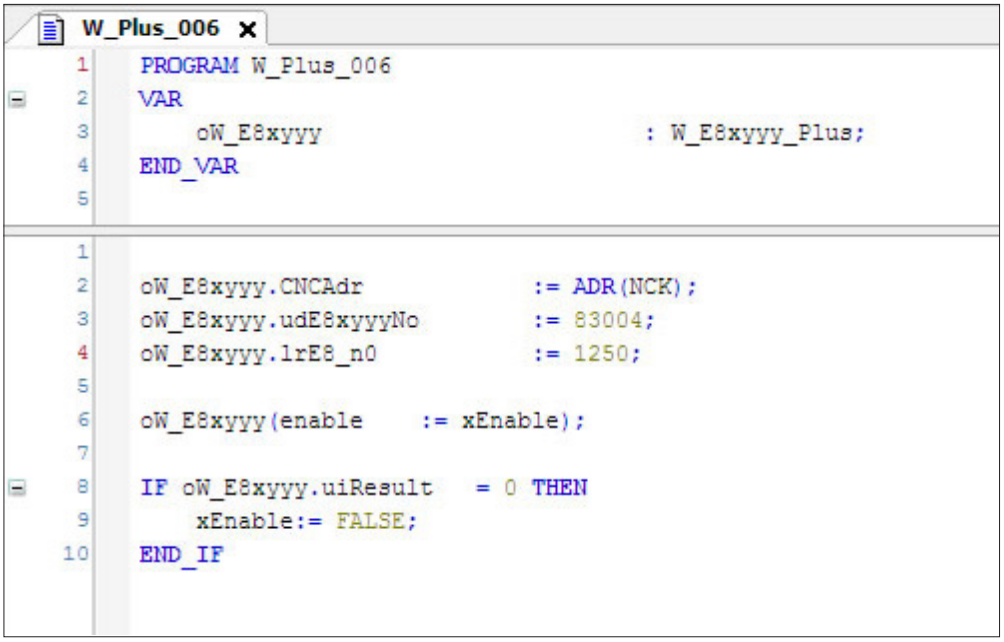
W_Plus_005

```
1  PROGRAM W_Plus_005
2  VAR
3      oW_Dat3                          : W_Dat3_Dec3_Plus;
4  END_VAR
5
```

```
1  oW_Dat3.CNCAdr              := ADR(NCK);
2  oW_Dat3.byChannel           := 0;
3  oW_Dat3.lrX_Axis                := 102; //  In internal units
4  oW_Dat3.lrY_Axis                := 201; //  In internal units
5  // You can also enter DAT values for the other axes
6
7  oW_Dat3(enable  := xEnable);
8
9  IF oW_Dat3.uiResult = 0 THEN
10     xEnable:= FALSE;
11 END_IF
```

6.3.6 Writing an E8… parameter value ► W_E8xyyy_Plus

This request allows for writing one E8xyyy parameter.
The integer part of the input argument will be written in the parameter.



6.3.7 Writing an E5 or E6 parameter value ► W_E_Param_5_6_Plus

This request allows for writing one parameter of the series E5 or E6.
Only the integer part of the input argument will be taken into account as internal units.

W_Plus_007

```
PROGRAM W_Plus_007
VAR
    oW_E_5_6                    : W_E_Param_5_6_Plus;
    Value                       : LREAL;
END_VAR
```

```

oW_E_5_6.CNCAdr         := ADR(NCK);
oW_E_5_6.byChannel      := 0;

oW_E_5_6.udE_number     := 70001;
oW_E_5_6.dataType       := 6;

Value                   := 123.456;
oW_E_5_6.ptValue            := ADR(Value);

oW_E_5_6(enable := xEnable);

IF oW_E_5_6.uiResult    = 0 THEN
    xEnable:= FALSE;
END_IF
```

6.3.8 Writing the home position (P16) ► W_Mach_Orig_Plus

This request allows for writing the home position for an axis (equivalent of P16).
Please note that this value is not written permanently in the parameter.

The value indicated in the project will be taken at the next system start-up. The request write the values at three consecutive axes addresses. **Therefore be sure to enter these three values.**

W_Plus_008

```
PROGRAM W_Plus_008
VAR
    oW_Home                     : W_Mach_Orig_Plus;
END_VAR
```

```

oW_Home.CNCAdr          := ADR(NCK);
oW_Home.byAxisNo            := 0;
oW_Home.lrAxe_n1            := 200;
oW_Home.lrAxe_n2            := 100;
oW_Home.lrAxe_n3            := 100;

oW_Home(enable  := xEnable);

IF oW_Home.uiResult = 0 THEN
    xEnable:= FALSE;
END_IF
```

6.3.9 Writing Maximum Dynamic Travel values ► W_Max_Dyn_Travel_Plus

This request allows for writing the maximum dynamic travels of the axes in a particular channel. The values are given in reference to the axes names.

As this request write the values for all axes of the channel be sure to fill-in the data for all these axes.

W_Plus_009

```
1  PROGRAM W_Plus_009
2  VAR
3      oW_Max_Dyn                    : W_Max_Dyn_Travel_Plus;
4  END_VAR
5
```

```
1
2  oW_Max_Dyn.CNCAdr           := ADR(NCK);
3  oW_Max_Dyn.byChannel        := 0;
4  oW_Max_Dyn.lrX_Axis         := 2000;
5  oW_Max_Dyn.lrY_Axis         := 2000;
6  oW_Max_Dyn.lrZ_Axis         := 2000;
7  oW_Max_Dyn.lrA_Axis         := 2000;
8  oW_Max_Dyn.lrB_Axis         := 2000;
9  oW_Max_Dyn.lrC_Axis         := 2000;
10 oW_Max_Dyn.lrU_Axis         := 2000;
11 oW_Max_Dyn.lrV_Axis         := 2000;
12 oW_Max_Dyn.lrW_Axis         := 2000;
13
14 oW_Max_Dyn(enable   := xEnable);
15
16 IF oW_Max_Dyn.uiResult  = 0 THEN
17     xEnable:= FALSE;
18 END_IF
```

6.3.10 Writing Maximum Static Travel values ► W_Max_Stat_Travel_Plus

This request allows for writing the maximum static travels of axes.
The values are given for three consecutives axes addresses.

As this request write the values for three consecutive axes addresses be sure to fill-in the data for all these axes.

W_Plus_010

```
1  PROGRAM W_Plus_010
2  VAR
3      oW_Max_Stat                    : W_Max_Stat_Travel_Plus;
4  END_VAR
5
```

```
1
2  oW_Max_Stat.CNCAdr          := ADR(NCK);
3  oW_Max_Stat.byAxisNo            := 0;
4  oW_Max_Stat.lrAxe_n1            := 2000;
5  oW_Max_Stat.lrAxe_n2            := 2000;
6  oW_Max_Stat.lrAxe_n3            := 2000;
7
8  oW_Max_Stat(enable  := xEnable);
9
10 IF oW_Max_Stat.uiResult = 0 THEN
11     xEnable:= FALSE;
12 END_IF
```

6.3.11 Writing Minimum Dynamic Travel values ► W_Min_Dyn_Travel_Plus

This request allows for writing the minimum dynamic travels of the axes in a particular channel. The values are given in reference to the axes names.

As this request write the values for all axes of the channel be sure to fill-in the data for all these axes.

W_Plus_011

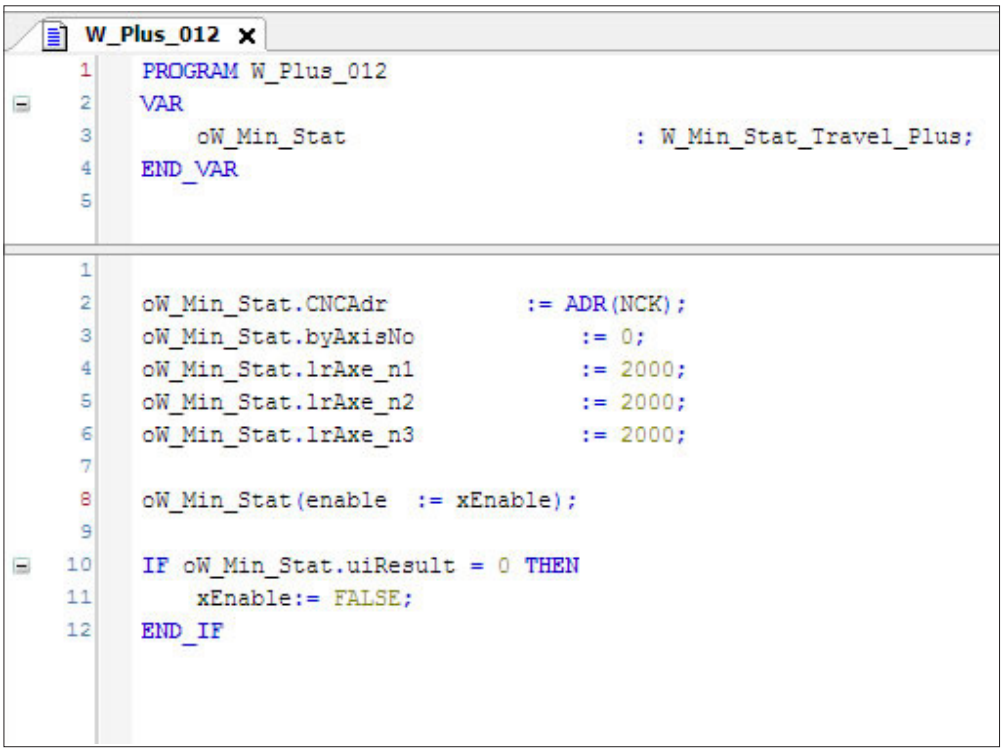
```
1   PROGRAM W_Plus_011
2   VAR
3       oW_Min_Dyn                      : W_Min_Dyn_Travel_Plus;
4   END_VAR
5
```

```
1
2   oW_Min_Dyn.CNCAdr           := ADR(NCK);
3   oW_Min_Dyn.byChannel        := 0;
4   oW_Min_Dyn.lrX_Axis         := 2000;
5   oW_Min_Dyn.lrY_Axis         := 2000;
6   oW_Min_Dyn.lrZ_Axis         := 2000;
7   oW_Min_Dyn.lrA_Axis         := 2000;
8   oW_Min_Dyn.lrB_Axis         := 2000;
9   oW_Min_Dyn.lrC_Axis         := 2000;
10  oW_Min_Dyn.lrU_Axis         := 2000;
11  oW_Min_Dyn.lrV_Axis         := 2000;
12  oW_Min_Dyn.lrW_Axis         := 2000;
13
14  oW_Min_Dyn(enable   := xEnable);
15
16  IF oW_Min_Dyn.uiResult  = 0 THEN
17      xEnable:= FALSE;
18  END_IF
```

6.3.12 Writing Minimum Static Travel values ► W_Min_Stat_Travel_Plus

This request allows for writing the minimum static travels of axes.
The values are given for three consecutives axes addresses.

As this request write the values for three consecutive axes addresses, be sure to fill-in the data for all these axes.



6.3.13 Writing an NCK parameter ► W_NCK_Par_Plus

This request allows for writing a word of a particular machine parameter.
The word number is given by the property *uiPoffset* (not shown in example).

The type of parameter must be given, the possibility of writing and type of parameters is given in annex.

W_Plus_013

```
1  PROGRAM W_Plus_013
2  VAR
3      oW_Nck_Par                    : W_Nck_Par_Plus;
4      Value                         : LREAL      := 500;
5  END_VAR
6
```

```
1  oW_Nck_Par.CNCAdr             := ADR(NCK);
2  oW_Nck_Par.byPType            := 10;
3  oW_Nck_Par.uiPnumber          := 30;
4  oW_Nck_Par.ptValue            := ADR(Value);
5
6  oW_Nck_Par(enable := xEnable);
7
8  IF oW_Nck_Par.uiResult  = 0 THEN
9      xEnable:= FALSE;
10 END_IF
```

6.3.14 Writing a tool offset ► W_Tool_Plus

This request allows for writing a particular tool offset.
All components are written with the exception of the H value which is given by a separate request.

W_Plus_014

```
PROGRAM W_Plus_014
VAR
    oW_Tool                     : W_Tool_Plus;
END_VAR
```

```
oW_Tool.CNCAdr          := ADR(NCK);
oW_Tool.byCorrNo            := 2;
oW_Tool.lrE50xxx            := 100;
oW_Tool.lrE51xxx            := 200;
oW_Tool.lrE52xxx            := 300;
oW_Tool.lrE53xxx            := 400;
oW_Tool.lrE54xxx            := 500;
oW_Tool.diE55xxx            := 3;
oW_Tool.diE57xxx            := 700;

oW_Tool(enable := xEnable);

IF oW_Tool.uiResult = 0 THEN
    xEnable:= FALSE;
END_IF
```

Page deliberately empty.

Page deliberately empty.

Summary new

Page deliberately empty.

A Appendix

A.1 Runtime Error Codes

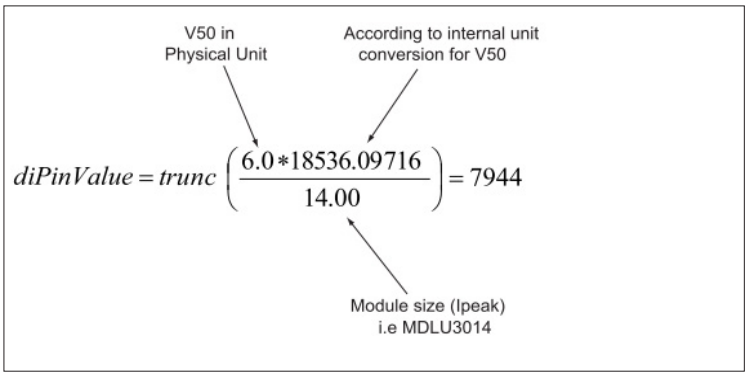
N°	Variable name	Value Description
0	ERR_OK	No error
1	ERR_FAILED	General error - to be used only for internal errors
2	ERR_PARAMETER	An invalid parameter was passed to a function.
3	ERR_NOTINITIALIZED	Function cannot be executed, since component has not been initialized yet. It may work later, though.
4	ERR_VERSION	Version conflict
5	ERR_TIMEOUT	Operation timed out
6	ERR_NOBUFFER	Insufficient memory to carry out the request
10	ERR_PENDING	For asynchronous calls: call not yet complete
11	ERR_NUMPENDING	Too many pending calls. Try later
12	ERR_NOTIMPLEMENTED	The function is not implemented
13	ERR_INVALIDID	No object with the provided id found
14	ERR_OVERFLOW	Integer overflow
15	ERR_BUFFERSIZE	The size of a buffer is to small or invalid
16	ERR_NO_OBJECT	No object with this specified name available
17	ERR_NOMEMORY	No heap memory available
18	ERR_DUPLICATE	An object with the same name is still available
19	ERR_MEMORY_OVERWRITE	Heap memory was written out of bounds! Memory overwrite error
20	ERR_INVALID_HANDLE	Invalid handle to an object
21	ERR_END_OF_OBJECT	End of object reached
22	ERR_NO_CHANGE	No changes done
23	ERR_INVALID_INTERFACE	Invalid or unknown interface
24	ERR_NOT_SUPPORTED	Functionality not supported
25	ERR_NO_ACCESS_RIGHTS	No access rights for this operation
26	ERR_OUT_OF_LIMITS	Specified limits of a resource exceeded
27	ERR_ENTRIES_REMAINING	Remaining entries that could not be transmitted because of buffer limitation
28	ERR_INVALID_SESSION_ID	Invalid online session id
29	ERR_EXCEPTION	Exception occurred
30	ERR_SIGNATURE_MISMATCH	Signature mismatch of an API function
31	ERR_VERSION_MISMATCH	Version mismatch
32	ERR_TYPE_MISMATCH	Type mismatch
33	ERR_ID_MISMATCH	ID mismatch
34	ERR_NO_CONSISTENCY	Consistency error

A.2 Drive parameters specificities

The drive parameters must be expressed in internal units (for rapidity) as well as in physical units (for coherence)

The relation between the physical units and internal units can be different according to the parameter and are described in the two following tables (For NUMDriveC and NUMDriveX)

As an example for V50 the relation is:



Ipeak and Inom drive current table

Module size	Ipeak (A)	Inom (Arms)	
		V271=100	V271=50
MDLU3007N	7.00	2.00	2.00
MDLU3014N	14.00	4.00	4.00
MDLU3021N	21.00	7.00	7.00
MDLU3034N	34.00	14.00	14.00
MDLU3050N	50.00	20.00	20.00
MDLU3075N	75.00	35.00	35.00
MDLU3100N	100.00	45.00	45.00
MDLU3150N	150.00	60.00	60.00
MDLU3014A	14.00	8.90	6.00
MDLU3021A	21.00	13.00	8.00
MDLU3034A	34.00	13.00	8.00
MDLU3050A	50.00	28.28	18.38
MDLU3075A	75.00	34.00	23.00
MDLU3130A	130.00	60.00	42.00
MDLU3200A	200.00	100.00	72.00
MDLU3400A	400.00	200.00	130.00
MDLU3014B	14.00	8.90	6.00
MDLU3021B	21.00	8.90	6.00
MDLU3050B	50.00	28.28	18.38

For more detailed informations on the NUMDrive parameters please refer to NUMDrive C Parameters manual AMOMAN007.

A.2.1 Case NUMDrive C Sw version 3.5x, 4.5x parameters conversions table

The conversions table may be different for other drive Software versions.
Therefore, please consult NUM for the latest informations.
The parameter value in internal unit [IU] is obtained by applying the conversion rule to the value in physical units [PU].

Example: V37 [IU] = V37 [PU] * 1000

Param. N°	Type	Halt	[PU]	Resol.	Conversion rule Vxx in physical unit	Description
V20	DWORD	Yes	[ ]	1	V20	Sensor connectors X3, X4 configuration
V21	DWORD	No	[ ]	1	V21	TP1/2 output configuration
V22	DWORD	Yes	[ ]	1	V22	PROGO1/2 output configuration
V23	DINT	Yes	[ ]	1	V23	Drives addresses confirm and check
V30	DWORD	Yes	[ ]	1	V30	General Drive Configuration
V32	DWORD	Yes	[ ]	1	V32	Config. Braking following alarms [1=Brake]
V33	DWORD	Yes	[ ]	1	V33	Alarm reset configuration [1=Reset]
V34	DWORD	Yes	[ ]	1	V34	Configuration DF1/DF2 relay [1=Break]
V35	DWORD	Yes	[ ]	1	V35	SW functions [0=Not available][1=Available]
V36	DINT	Yes	[ ]	1	V36	SAM scaling factor
V37	REAL	Yes	s	0.01	V37*1000	Delay time of power detection
V40	REAL	No	%	0.01	V40*65536/100	Speed reached dynamic threshold
V41	REAL	No	mm/min	0.01	V41*163.840	Motor zero speed threshold, enable, brake
V42	REAL	No	Rpm	0.01	V42*1342.18	Motor zero speed threshold, enable, brake
V43	REAL	Yes	s	0.01	V43*1000	Braking timeout on config. V30 V32
V44	DINT	No	%	1	V44*1024/100	Braking current on config. V30 V32
V45	REAL	No	mm/min	0.01	V45*163.840	Speed dynamic check [0]
V46	REAL	No	Rpm	0.01	V46*1342.18	Speed dynamic check [0]
V50	REAL	No	Arms	0.01	V50*18536.09716/Ipeak	Torque current limit [0]
V51	REAL	No	mA/Rpm	0.01	V51*56.56854/Ipeak	Proportional speed loop gain [0]
V52	REAL	No	ms	0.01	32767*V264/(V52*1000)	Integral speed loop gain [0]
V53	REAL	No	ms	0.01	V264*32767/(V53*1000)	AP10: 2^ Integral speed loop gain [0]
V57	REAL	No	Arms	0.01	V57*18536.09716/Ipeak	Torque current limit h.r. [0]
V58	REAL	No	mA/Rpm	0.01	V58*56.56854/Ipeak	Proportional speed loop gain h.r. [0]
V59	REAL	No	ms	0.01	32767*V264/(V59*1000)	Integral speed loop gain h.r. [0]
V60	REAL	No	Arms/m/min	0.01	V60*463.4095/Ipeak	Proportional speed loop gain [0]
V75	REAL	No	ms	0.01	V264*32767/(V75*1000)	AP10: 2^ Integral speed loop gain h.r. [0]
V80	REAL	No	[ ]	0.01	32767.5*V80	Set Point Weight speed loop gain
V81	REAL	No	[ ]	0.01	16383.5*V81	Anti-windup speed loop gain
V91	DINT	No	Rpm/s	1	V91*V265*1.34218/1000	Acceleration/deceleration ramp [0]
V92	DINT	No	Rpm/s	1	V92*V265*1.34218/1000	Acceleration/deceleration ramp h.r.[0]
V93	DINT	No	mm/s^2	1	V93*60*V265*0.163840/1000	Acceleration/deceleration ramp [0]
V94	DINT	No	%	1	V94*163840*V95/100	Speed reference Limit [% maximum speed] [0]
V95	DINT	No	m/mn	1	V95*163840	Maximum motor speed [0]
V96	DINT	No	m/mn	1	V96*163840	Threshold for overspeed alarm [0]
V97	DINT	No	%	1	V97*1342.18*V98/100	Speed reference limit [% maximum speed] [0]

Param. N°	Type	Halt	[PU]	Resol.	Conversion rule Vxx in physical unit	Description
V98	DINT	No	Rpm	1	V98*1342.18	Maximum motor speed [0]
V99	DINT	No	Rpm	1	V99*1342.18	Overspeed alarm threshold [0]
V100	REAL	No	Arms	0.01	V100*18536.09716/Ipeak	Deliverable current limit
V101	DINT	No	V	1	V101*8	Voltage limits Vd and Vq
V102	REAL	No	V/Arms	0.01	(V102*Ipeak/0.2828427)/4	Proportional current loop gain
V103	REAL	No	ms	0.01	32767*V271/(1000*V103)	Integral current loop gain
V104	REAL	No	mVrms/ Rpm	0.01	V104*16.237976*2/V257	Motor EMF phase compensation
V105	REAL	No	mH	0.01	(V105+V408)*Ipeak*1.503011016	Compensation of motor Lq inductance
V106	REAL	No	mH	0.01	(V106+V408)*Ipeak*1.503011016	Compensation of motor Ld inductance
V107	DINT	Yes	Hz	1	(0.102943708*V271*V107)/ (1+0.00000314159*V271*V107)	Lowpass Filter elettrical speed
V108	REAL	Yes	Arms	0.01	V108*18536.09716/Ipeak	Rated motor current [I^2t]
V109	REAL	No	Arms	0.01	V109*18536.09716/Ipeak	Vertical load compensation current
V110	REAL	No	Arms	0.01	V110*18536.09716/Ipeak	Deliverable current limit h.r.
V111	REAL	No	V/Arms	0.01	(V111*Ipeak/2828.427)/4	Proportional current gain h.r.
V112	REAL	No	ms	0.01	32767*V271/(1000*V112)	Integral current gain h.r.
V114	REAL	No	mH	0.01	((V105*1.00/V407) +V408)*Ipeak*1.503011016	Motor Lq inductance compensation h.r.
V115	REAL	Yes	Arms	0.01	V115*18536.09716/Ipeak	Rated motor current [I^2t] h.r.
V116	REAL	No	mH	0.01	V116*Ipeak*1.503011016	Lm (for compensation) l.r.
V117	REAL	No	mH	0.01	V116*Ipeak*1.503011016/V407	Lm (for compensation) h.r.
V118	REAL	No	mH	0.01	((V106*1.00/ V407)+V408)*Ipeak*1.503011016	Motor Ld inductance compensation h.r.
V119	REAL	No	cycles	0.01	V119*(V271/V263)*1024	Phase lead
V120	REAL	No	Vrms/m/s	0.01	V120*V259*54.12659/(50*V257)	Linear Motor EMF phase compensation
V121	REAL	No	[ ]	0.01	32767.5*V121	Set Point Weight current loop gain
V122	REAL	No	[ ]	0.01	16383.5*V122	Anti-windup speed current gain
V123	DINT	No	[ ]	1	V123	Step of Iq ref. interpolation
V125	DINT	No	%	1	V125*V108*185.3609716/Ipeak	Current percentage V108 for EPSM
V128	DINT	Yes	%	1	V128*V108*185.3609716/Ipeak	Locked rotor current I^2t (% V108)
V129	DINT	Yes	%	1	V129*V115*185.3609716/Ipeak	Locked rotor current I^2t (% V115)
V130	DINT	No	[ ]	1	V130	Associate MP with TP1
V131	DINT	No	[ ]	1	V131	Associate MP with TP2
V132	DINT	No	[ ]	1	V132	Associate MP with TP3
V133	DINT	No	[ ]	1	V133	Associate MP with TP4
V134	DINT	No	[ ]	1	V134	Multiplication factor for TP1
V135	DINT	No	[ ]	1	V135	Multiplication factor for TP2
V136	DINT	No	[ ]	1	V136	Offset for analog TP1
V137	DINT	No	[ ]	1	V137	Offset for analog TP2
V138	DWORD	No	[ ]	1	P138	Configuration of analog TPs
V139	DWORD	Yes	[ ]	1	V139	Configuration of Programmable Relays
V140	DINT	No	[ ]	1	V140	Associate MP with programmable relay 1
V141	REAL	No	[ ]	0.01	V141*V143	Programmable relay 1. threshold 1
V142	REAL	No	[ ]	0.01	V142*V143	Programmable relay 1. threshold 2
V143	REAL	No	[ ]	0.01	V143	Programmable relay 1. MP scale factor
V144	DINT	No	ms	1	V144/(V263/1000)	Programm. relay 1. tripping time constant
V150	DINT	No	[ ]	1	V150	Associate MP with programmable relay 2
V151	REAL	No	[ ]	0.01	V151*V153	Programmable relay 2. threshold 1
V152	REAL	No	[ ]	0.01	V152*V153	Programmable relay 2. threshold 2
V153	REAL	No	[ ]	0.01	V153	Programmable relay 2. MP scale factor
V154	DINT	No	ms	1	V154/(V263/1000)	Programm. relay 2. tripping time constant

Param. N°	Type	Halt	[PU]	Resol.	Conversion rule Vxx in physical unit	Description
V160	DWORD	Yes	[ ]	1	V160	Special applications selection
V165	DINT	No	mm/min	1	V165*163.840	AV3 Mst: Max compens. torque equalizer
V166	REAL	No	mm/min/ Arms	0.01	V166*Ipeak*565.6941/1000	AV3 Mst: Proportional torq. equaliz. gain
V168	DINT	Yes	[ ]	1	V168	AV3 Mst: N.of slaves on the local link
V169	DWORD	Yes	[ ]	1	V169	Link Tandem configuration
V170	DWORD	Yes	[ ]	1	V170	Tandem application configuration
V171	REAL	No	Arms	0.01	-2*V171*18536.09716/Ipeak	AV3 Master: Preload Tandem current
V172	REAL	No	Rpm/Arms	0.01	V172*Ipeak/0.215792	AV3 Mst: Proportional torq. equaliz. gain
V173	REAL	No	ms	0.01	32767*V265/(1000*V173)	AV3 Mst: Integral torque equalizer gain
V175	DINT	No	Rpm	1	V175*1342.18	AV3 Mst: Max compens. torque equalizer
V176	DINT	No	ms	1	(V176*1000)/V265	AV3 Mst: T.out A52.41 (saturated trq eq.)
V177	REAL	No	Arms/s	0.01	2*V177*0.01.853609716*65536 *V265/Ipeak	AV3: Master: Preload ramp speed
V180	DINT	Yes	%	1	(V180*V108*131.07)/Ipeak	Current percentage (V108) for EPS
V182	DINT	Yes	%	1	V182*32767/100	Phase U/W relative weight for EPS
V183	DINT	Yes	%	1	V183*32767/100	Phase V/W relative weight for EPS
V190	DINT	No	[ ]	1	V190	AV6: Max displacement meas. S1-S2 Ph.U.
V191	DINT	No	ms	1	V191	AV6: Max duration measure displac. S1-S2
V205	REAL	No	Hz	0.01	(0.102943708*V265*V205)/ (1.0+0.00000314159*V265*V205)	AP13: RMS extractor filter
V206	REAL	No	Rpm	0.01	V206*1342.18	AP13: RMS filter output saturation
V207	REAL	No	%	0.01	V206*1342.18*V207/100.0	AP13: Offset of modulation factor
V208	REAL	No	mm/min	0.01	V208*163.840	AP13: RMS filter output saturation
V209	REAL	No	%	0.01	V208*163.840*V209/100.0	AP13: Offset of modulation factor
V210	REAL	No	[ ]	0.01	-V210*1024	AV8: Active damping 1 factor
V217	REAL	No	[ ]	0.01	32.0*V217	AP12: Active damping 2 K1
V218	REAL	No	Hz	0.01	(0.102943708*V265*V218)/ (V217+0.00000314159*V265*V218)	AP12: Active damping 2 filter
V219	DINT	No	[ ]	1	32767.0*(V217- 0.00000314159*V265*V218)/ (V217+0.00000314159*V265*V218)	AP12: coeff.A Active damping 2 filter
V220	DWORD	Yes	[ ]	1	V220	Reference configuration
V221	DWORD	Yes	[ ]	1	V221	Enable configurations
V222	DINT	No	[ ]	1	V222	Shift number of analog stimulus
V223	DINT	Yes	Hz	1	(0.102943708*V265*V223)/ (1+0.00000314159*V265*V223)	Low pass Filter analog reference
V224	DINT	Yes	Hz	1	32768 - 2*(0.102943708*V265*V223)/ (1+0.00000314159*V265*V223)	Low pass Filter analog reference [hidden]
V225	DWORD	Yes	[ ]	1	V225	Digital input configuration [GP_INA/B]
V238	DWORD	Yes	[ ]	1	V238	UGA filter on S1 sensor (RCMDS20)
V239	DWORD	Yes	[ ]	1	V239	UGA filter on S2 sensor (RCMDS21)
V240	REAL	Yes	s	0.01	V240*100	First time constant for I^2t
V241	DINT	Yes	s	1	V241*1	Second time constant for I^2t
V245	REAL	No	Hz	0.01	V245*13.1072	Freq. threshold. Change of time const. I^2t
V246	REAL	No	Hz	0.01	V246*13.1072	Frequency threshold hysteresis V245
V250	DWORD	Yes	[ ]	1	V250	ST Revision
V251	DINT	Yes	[ ]	1	V251	Parameter Compatibility
V252	STRING	Yes	[ ]		V252	Drive type
V253	DWORD	Yes	[ ]	1	V253	Drive size
V254	DINT	Yes	[ ]	1	V254	Motor number
V255	STRING	Yes	[ ]		V255	Motor type
V256	STRING	Yes	[ ]		V256	Motor power Connection type

Param. N°	Type	Halt	[PU]	Resol.	Conversion rule Vxx in physical unit	Description
V257	DINT	Yes	[ ]	1	V257	Motor poles pairs number
V258	STRING	Yes	[ ]		V258	Motor sensor type
V259	REAL	Yes	mm	0.01	V259	Linear Motor Pole Pitch [180 degrees el.]
V260	DINT	No	[ ]	1	V260	Speed regulator Parameters exchange
V261	DINT	No	[ ]	1	V261	Position regulator Parameters exchange
V263	DINT	Yes	µs	50	V263	System sampling time
V264	DINT	Yes	µs	100	V264/V263	Speed loop sampling time
V265	DINT	Yes	µs	200	V265/V263	Position loop sampling time
V266	REAL	Yes	ms	0.2	V266*1000/V263	CNC Task Sampling time
V267	DINT	Yes	A	1	V267*100	ST Peak current
V268	DINT	Yes	mDeg	1	V268*65.536/360.0	Electrical phase offset Hall effect sensor
V269	DINT	Yes	mDeg	1	V269*65.536/360.0	Position offset for electrical phasing
V270	DINT	Yes	Ohm	1	V270	Motor PTC-NTC value
V271	DINT	Yes	µs	50	V271/V263	Current loop sampling time
V272	DWORD	Yes	[ ]	1	V272	Motor thermal sensor configuration
V273	DWORD	Yes	[ ]	1	P273	Electrical phase offset management
V301	DINT	No	mm/min/ mm	1	V301	Proportional position gain [0]
V305	DINT	Yes	[ ]	1	V305	Measure module for rot. axis/spindle abs[0]
V306	DINT	Yes	[ ]	1	V306	S1 sensor MULTI coefficient [0]
V307	DINT	Yes	[ ]	1	V307	S1 sensor DIVI coefficient [0]
V310	DINT	Yes	[ ]	1	V310	S1 sensor offset in CNC Phys. Unit [0]
V311	DINT	Yes	[ ]	1	V311	S2 sensor MULTI coefficient [0]
V312	DINT	Yes	[ ]	1	V312	S2 sensor DIVI coefficient [0]
V313	DINT	Yes	[ ]	1	V313	S2 sensor offset in CNC Phys. Unit [0]
V314	DWORD	Yes	[ ]	1	P314	Sinusoidal signals range of S1 Sensor
V315	DWORD	Yes	[ ]	1	P315	Sinusoidal signals range of S2 Sensor
V317	REAL	Yes	V	0.01	V317*1638.4	S1 sensor power supply voltage
V318	REAL	Yes	V	0.01	V318*1638.4	S2 sensor power supply voltage
V340	DINT	No	[ ]	1	V340	Step of speed calculation S1
V341	DINT	No	[ ]	1	V341	Step of speed calculation S2
V342	REAL	No	Hz	0.01	(0.102943708*V264*V342)/ (1+0.00000314159*V264*V342)	Speed filter by S1 sensor
V343	REAL	No	Hz	0.01	(0.102943708*V264*V343)/ (1+0.00000314159*V264*V343)	Speed filter by S2 sensor
V344	DINT	Yes	[ ]	1	V344	S1 Sensor code ID
V345	DINT	Yes	[ ]	1	V345	S2 Sensor code ID
V346	DINT	Yes	[ ]	1	V346	S1 Sensor number
V347	DINT	Yes	[ ]	1	V347	S2 Sensor number
V348	DWORD	Yes	[ ]	1	P348	S1 Sensor configuration
V349	DWORD	Yes	[ ]	1	V349	S2 Sensor configuration
V350	DWORD	Yes	[ ]	1	P350	Position loop sensor config.
V351	DWORD	Yes	[ ]	1	P351	Speed loop sensor config.
V352	DWORD	Yes	[ ]	1	P352	Current loop sensor config.
V353	REAL	Yes	[ ]	0.01	V353*100	Period of position sensor/motor turn [0]
V355	DINT	Yes	[ ]	1	V355	AV2: Mechanical ratio: load turns
V356	DINT	Yes	[ ]	1	V356	AV2: Mechanical ratio: sensor turns
V359	DINT	Yes	[ ]	1	(65536*3*4*1024/V353)*100/V264	k_velrot120
V360	DINT	Yes	[ ]	1	V360	Period by turn sensor S1
V361	DINT	Yes	[ ]	1	V361	Absolute resolution by turn sensor S1
V362	DINT	No	[ ]	1	V301*1.31072*32768*4/V360	kpposrot10

Param. N°	Type	Halt	[PU]	Resol.	Conversion rule Vxx in physical unit	Description
V363	DINT	Yes	[ ]	1	(65536*3*1024*4/V360)*100/V264	k_velrot10
V364	DINT	Yes	[ ]	1	V361/V360	sinuarot10
V365	DINT	Yes	[ ]	1	V365	Number of absolute sensor turns S1
V367	DINT	No	[ ]	1	V367	Not used
V368	DINT	Yes	[ ]	1	V368	Intervals between coded references S1
V370	DINT	Yes	[ ]	1	V370	Period by turn sensor S2
V371	DINT	Yes	[ ]	1	V371	Absolute resolution by turn sensor S2
V372	DINT	No	[ ]	1	V301*1.31072*32768*4/V353	kpposrot120
V373	DINT	Yes	[ ]	1	(65536*3*1024*4/V370)*100/V264	k_velrot20
V374	DINT	Yes	[ ]	1	V371/V370	sinuarot20
V375	DINT	Yes	[ ]	1	V375	Number of absolute sensor turns S2
V376	DINT	No	[ ]	1	V301*1.31072*32768*4/V370	kpposrot20
V377	DINT	No	[ ]	1	V377	Not used
V378	DINT	Yes	[ ]	1	V378	Intervals between coded references S2
V380	DINT	Yes	µm	1	V380	Period of linear sensor S1
V381	DINT	Yes	nm	1	V381	Absolute resolution of linear sensor S1
V382	DINT	Yes	[ ]	1	(V259*2000/V380)+0.1	npslin10
V383	DINT	No	[ ]	1	V301*5.24288*V380*4	kpposlin10
V384	DINT	Yes	[ ]	1	(24576*V380*4)*100/V264	k_vellin10
V385	DINT	Yes	[ ]	1	V380*1000/V381	sinualin10
V386	DINT	Yes	nm	1	(V259*2000000/V381)+0.1	ralinppm10
V390	DINT	Yes	µm	1	V390	Period of linear sensor S2
V391	DINT	Yes	nm	1	V391	Absolute resolution of linear sensor S2
V392	DINT	Yes	[ ]	1	(V259*2000/V390)+0.1	npslin20
V393	DINT	No	[ ]	1	V301*5.24288*V390*4	kpposlin20
V394	DINT	Yes	[ ]	1	(24576*V390*4)*100/V264	k_vellin20
V395	DINT	Yes	[ ]	1	V390*1000/V391	sinualin20
V396	DINT	Yes	nm	1	(V259*2000000/V391)+0.1	ralinppm20
V400	DINT	No	mOhm	1	V400*3.2768*V406*(V271/50)/V403	Rs at 120 degrees C
V401	DINT	No	mOhm	1	V401*3.2768*V406*(V271/50)/V403	Rr at 150 degrees C
V402	DINT	No	µH	1	(V402+V408*1000)*65.536/V403	sigmaLs
V403	REAL	No	mH	0.01	V403	Lr
V404	DINT	No	[ ]	1	133774.75*32.768*(V271/50)/ (Ipeak*V403)	[I.U.=(Ki/Kv) /Lr * (Ts*2^16)=a04] Ki/Kv
V405	DINT	No	[ ]	1	133774.75*32.768*V407*(V271/50)/ (Ipeak*V403)	[U.I.=a04b*ratio_gamma=a04a] a04a
V406	DINT	No	[ ]	1	V406	twotosh0
V407	DINT	No	[ ]	1	V407	ratio_gamma
V408	REAL	No	mH	0.01	V408	Additional extern inductance
V409	DINT	No	V	1	V409*8	ST flux-weakening voltage
V410	REAL	No	[]	0.01	V410*256	Proportional gain flux loop l.r.
V411	REAL	No	ms	0.01	32.767*V264/V411	Integral gain flux loop l.r.
V412	DINT	No	%	1	0.64*V412	Lr saturation level l.r.
V413	DINT	No	[ ]	1	32768/(0.64*V412)	ksatinv l.r.
V414	DINT	No	Rpm	1	V414*0.3431457*V257*(V271/100)	Observer engagement speed l.r.
V415	DINT	No	Rpm	1	V415*0.3431457*V257*(V271/100)	Observer engagement hysteresis l.r.
V417	DINT	No	Rpm	1	V417*V257*0.21845333	Defluxing starting speed l.r.
V425	DINT	No	%	1	V425*V427 *V412*1.31072*1.4142135/Ipeak	min_flux as a %age of V427 l.r.
V426	REAL	No	Arms/Vs	0.01	V426*75.923856*V264/Ipeak	K deflux l.r.

Param. N°	Type	Halt	[PU]	Resol.	Conversion rule Vxx in physical unit	Description
V427	REAL	No	Arms	0.01	V427*V412*131.072*1.4142135/ Ipeak	Magnetization current l.r.
V428	REAL	No	Arms	0.01	V428*18536.09716/Ipeak	Magnetization current limit l.r.
V448	DINT	No	µH	1	(V402+V407*V408*1000)*65.536/ V403	sigmaLs h.r.
V450	REAL	No	[ ]	0.01	V450*256	Proportional flux loop gain h.r.
V451	REAL	No	ms	0.01	32.767*V264/V451	Integral flux loop gain h.r.
V452	DINT	No	%	1	0.64*V452	Lr saturation level h.r.
V453	DINT	No	[ ]	1	32768/(0.64*V452)	ksatinv h.r.
V454	DINT	No	Rpm	1	V454*0.3431457*V257*(V271/100)	Observer engagement speed h.r.
V455	DINT	No	Rpm	1	V455*0.3431457*V257*(V271/100)	Observer engagement hysteresis h.r.
V457	DINT	No	Rpm	1	V457*V257*0.21845333	Defluxing starting speed h.r.
V465	DINT	No	%	1	V465*V467 *V452*1.31072*1.4142135/Ipeak	min_flux as a %age of V467 h.r.
V466	REAL	No	Arms/Vs	0.01	V466*75.923856*V264/Ipeak	K deflux h.r.
V467	REAL	No	Arms	0.01	V467*V452*131.07.2*1.4142135/ Ipeak	Magnetization current h.r.
V468	REAL	No	Arms	0.01	V468*18536.09716/Ipeak	Magnetization current limit h.r.
V470	REAL	No	mH	0.01	((V402/1000.0)+V408)*Ipe ak*1.503011016	SigmaLs flux-weakening l.r.
V471	REAL	No	mH	0.01	((V402/(V407*1000.0))+V408)* Ipeak*1.503011016	SigmaLs flux-weakening h.r.
V472	DINT	No	[ ]	1	(V400+V401)* Ipeak*6553.6/133772.76	Total resistance power lost Ri2 l.r.
V473	DINT	No	[ ]	1	(V400+V401)*Ipeak*6553.6/ (V407*133772.76)	Total resistance power lost Ri2 h.r.
V480	REAL	No	%	0.01	V480*655.35	Correction factor for MV6 Delivered Power
V481	DWORD	Yes	[ ]	1	V481	Control mode selection
V488	DINT	No	mOhm	1	V488*Ipeak*6553.6*(234.5+120.0)/ ((234.5+20.0)*2*133772.76)	Resistance tt at 20 degrees C
V489	REAL	No	mH	0.01	(V408+V489)*Ipeak*1.503011016	Motor Lq inductance
V491	REAL	No	Arms	0.01	V491*18536.09716/Ipeak	Id_Max (defluxing parameter)
V492	REAL	No	Arms	0.01	V492*18536.09716/Ipeak	Id current without defluxing
V496	DINT	No	Rpm	1	V496*V257*0.21845333	Defluxing starting speed
V497	REAL	No	m/mn	0.01	V497*109.22667*V257/V259	Defluxing starting speed
V498	REAL	No	Arms/Vs	0.01	V498*75.923856*V264/Ipeak	K Iq reduction
V499	REAL	No	Arms/Vs	0.01	V499*75.923856*V264/Ipeak	K flux-weakening Id
V500	DWORD	No	[ ]	1	P500	Filter configuration
V501	DINT	No	[ ]	1	V501	ph11 (FILTER 1 for torque current ref.)
V502	DINT	No	[ ]	1	V502	ph12 (FILTER 1 for torque current ref.)
V503	DINT	No	[ ]	1	V503	ph21 (FILTER 1 for torque current ref.)
V504	DINT	No	[ ]	1	V504	ph22 (FILTER 1 for torque current ref.)
V505	DINT	No	[ ]	1	V505	ga1 (FILTER 1 for torque current ref.)
V506	DINT	No	[ ]	1	V506	ga2 (FILTER 1 for torque current ref.)
V507	DINT	No	[ ]	1	V507	r*r (FILTER 1 for torque current ref.)
V508	DINT	No	[ ]	1	V508	C1 (FILTER 1 for torque current ref.)
V509	DINT	No	[ ]	1	V509	C2 (FILTER 1 for torque current ref.)
V511	DINT	No	[ ]	1	V511	ph11 (FILTER 2 for torque current ref.)
V512	DINT	No	[ ]	1	V512	ph12 (FILTER 2 for torque current ref.)
V513	DINT	No	[ ]	1	V513	ph21 (FILTER 2 for torque current ref.)
V514	DINT	No	[ ]	1	V514	ph22 (FILTER 2 for torque current ref.)
V515	DINT	No	[ ]	1	V515	ga1 (FILTER 2 for torque current ref.)
V516	DINT	No	[ ]	1	V516	ga2 (FILTER 2 for torque current ref.)

Param. N°	Type	Halt	[PU]	Resol.	Conversion rule Vxx in physical unit	Description
V517	DINT	No	[ ]	1	V517	r*r (FILTER 2 for torque current ref.)
V518	DINT	No	[ ]	1	V518	C1 (FILTER 2 for torque current ref.)
V519	DINT	No	[ ]	1	V519	C2 (FILTER 2 for torque current ref.)
V521	DINT	No	[ ]	1	V521	ph11 (FILTER 3 for torque current ref.)
V522	DINT	No	[ ]	1	V522	ph12 (FILTER 3 for torque current ref.)
V523	DINT	No	[ ]	1	V523	ph21 (FILTER 3 for torque current ref.)
V524	DINT	No	[ ]	1	V524	ph22 (FILTER 3 for torque current ref.)
V525	DINT	No	[ ]	1	V525	ga1 (FILTER 3 for torque current ref.)
V526	DINT	No	[ ]	1	V526	ga2 (FILTER 3 for torque current ref.)
V527	DINT	No	[ ]	1	V527	r*r (FILTER 3 for torque current ref.)
V528	DINT	No	[ ]	1	V528	C1 (FILTER 3 for torque current ref.)
V529	DINT	No	[ ]	1	V529	C2 (FILTER 3 for torque current ref.)
V531	DINT	No	[ ]	1	V531	ph11 (FILTER 4 for torque current ref.)
V532	DINT	No	[ ]	1	V532	ph12 (FILTER 4 for torque current ref.)
V533	DINT	No	[ ]	1	V533	ph21 (FILTER 4 for torque current ref.)
V534	DINT	No	[ ]	1	V534	ph22 (FILTER 4 for torque current ref.)
V535	DINT	No	[ ]	1	V535	ga1 (FILTER 4 for torque current ref.)
V536	DINT	No	[ ]	1	V536	ga2 (FILTER 4 for torque current ref.)
V537	DINT	No	[ ]	1	V537	r*r (FILTER 4 for torque current ref.)
V538	DINT	No	[ ]	1	V538	C1 (FILTER 4 for torque current ref.)
V539	DINT	No	[ ]	1	V539	C2 (FILTER 4 for torque current ref.)
V541	DINT	No	[ ]	1	V541	ph11 (FILTER 1 for speed ref.)
V542	DINT	No	[ ]	1	V542	ph12 (FILTER 1 for speed ref.)
V543	DINT	No	[ ]	1	V543	ph21 (FILTER 1 for speed ref.)
V544	DINT	No	[ ]	1	V544	ph22 (FILTER 1 for speed ref.)
V545	DINT	No	[ ]	1	V545	ga1 (FILTER 1 for speed ref.)
V546	DINT	No	[ ]	1	V546	ga2 (FILTER 1 for speed ref.)
V547	DINT	No	[ ]	1	V547	r*r (FILTER 1 for speed ref.)
V548	DINT	No	[ ]	1	V548	C1 (FILTER 1 for speed ref.)
V549	DINT	No	[ ]	1	V549	C2 (FILTER 1 for speed ref.)
V551	DINT	No	[ ]	1	V551	ph11 (FILTER notch for AP13)
V552	DINT	No	[ ]	1	V552	ph12 (FILTER notch for AP13)
V553	DINT	No	[ ]	1	V553	ph21 (FILTER notch for AP13)
V554	DINT	No	[ ]	1	V554	ph22 (FILTER notch for AP13)
V555	DINT	No	[ ]	1	V555	ga1 (FILTER notch for AP13)
V556	DINT	No	[ ]	1	V556	ga2 (FILTER notch for AP13)
V557	DINT	No	[ ]	1	V557	r*r (FILTER notch for AP13)
V558	DINT	No	[ ]	1	V558	C1 (FILTER notch for AP13)
V559	DINT	No	[ ]	1	V559	C2 (FILTER notch for AP13)
V600	DINT	No	%	1	V600*V108*185.3609716/Ipeak	V/f: Maximal motor current (% of V108)
V601	DINT	No	Hz	1	V601*13.1072	V/f: Rated frequency
V602	REAL	No	V	0.01	V602*8	V/f: Rated voltage
V603	REAL	No	V	0.01	V603*8	V/f: V
V604	DINT	No	mOhm	1	V604*Ipeak*6553.6*1.4/133772.76	V/f: Stator resistance @ 20 degrees C
V605	DINT	No	%	1	V605*V108*185.3609716/Ipeak	V/f: Intermittent overload (% of V108)
V606	DINT	No	Hz	1	V606*13.1072	V/f: Freq. start of overload reduction
V1045	REAL	No	mm/min	0.01	P1045*163.840	Speed dynamic check [1]
V1046	REAL	No	Rpm	0.01	P1046*1342.18	Speed dynamic check [1]
V1050	REAL	No	Arms	0.01	P1050*18536.09716/Ipeak	Torque current limit [1]
V1051	REAL	No	mA/Rpm	0.01	P1051*56.56854/Ipeak	Proportional speed loop gain [1]

Param. N°	Type	Halt	[PU]	Resol.	Conversion rule Vxx in physical unit	Description
V1052	REAL	No	ms	0.01	32767*V264/(P1052*1000)	Integral speed loop gain [1]
V1053	REAL	No	ms	0.01	V264*32767/(P1053*1000)	AP10: 2^ Integral speed loop gain [1]
V1057	REAL	No	Arms	0.01	P1057*18536.09716/Ipeak	Torque current limit h.r. [1]
V1058	REAL	No	mA/Rpm	0.01	P1058*56.56854/Ipeak	Proportional speed loop gain h.r. [1]
V1059	REAL	No	ms	0.01	32767*V264/(P1059*1000)	Integral speed loop gain h.r. [1]
V1060	REAL	No	Arms/m/ min	0.01	P1060*463.4095/Ipeak	Proportional speed loop gain [1]
V1075	REAL	No	ms	0.01	V264*32767/(P1075*1000)	AP10: 2^ Integral speed loop gain h.r. [1]
V1091	DINT	No	Rpm/s	1	P1091*V265*1.34218/1000	Acceleration/deceleration ramp [1]
V1092	DINT	No	Rpm/s	1	P1092*V265*1.34218/1000	Acceleration/deceleration ramp h.r.[1]
V1093	DINT	No	mm/s^2	1	P1093*60*V265*0.163840/1000	Acceleration/deceleration ramp [1]
V1094	DINT	No	%	1	P1094*163840*P1095/100	Speed reference Limit [% maximum speed] [1]
V1095	DINT	No	m/mn	1	P1095*163840	Maximum motor speed [1]
V1096	DINT	No	m/mn	1	P1096*163840	Threshold for overspeed alarm [1]
V1097	DINT	No	%	1	P1097*1342.18*P1098/100	Speed reference Limit [% maximum speed] [1]
V1098	DINT	No	Rpm	1	P1098*1342.18	Maximum motor speed [1]
V1099	DINT	No	Rpm	1	P1099*1342.18	Overspeed alarm threshold [1]
V1301	DINT	No	mm/min/ mm	1	P1301	Proportional position gain [1]
V1305	DINT	Yes	[ ]	1	P1305	Measure module for rot. axis/spindle abs[1]
V1306	DINT	Yes	[ ]	1	P1306	S1 sensor MULTI coefficient [1]
V1307	DINT	Yes	[ ]	1	P1307	S1 sensor DIVI coefficient [1]
V1310	DINT	Yes	[ ]	1	P1310	S1 sensor offset in CNC Phys. Unit [1]
V1311	DINT	Yes	[ ]	1	P1311	S2 sensor MULTI coefficient [1]
V1312	DINT	Yes	[ ]	1	P1312	S2 sensor DIVI coefficient [1]
V1313	DINT	Yes	[ ]	1	P1313	S2 sensor offset in CNC Phys. Unit [1]
V1353	REAL	Yes	[ ]	0.01	P1353*100	Period of position sensor/motor turn [1]
V1359	DINT	Yes	[ ]	1	(65536*3*4*1024/P1353)*100/V264	k_velrot121
V1362	DINT	No	[ ]	1	P1301*1.31072*32768*4/V360	kpposrot11
V1372	DINT	No	[ ]	1	P1301*1.31072*32768*4/P1353	kpposrot121
V1376	DINT	No	[ ]	1	P1301*1.31072*32768*4/V370	kpposrot21
V1383	DINT	No	[ ]	1	P1301*5.24288*V380*4	kpposlin11
V1393	DINT	No	[ ]	1	P1301*5.24288*V390*4	kpposlin21

(1): Applicable to the Mono-Axis drives only.
Whenever is needed to change the parameters, due to Multi-Spindle with synchronous motors application, such parameters may be changed with the drive in Run Status and with Torque off.

A.2.2 Dependencies table

Param.	Parameters impacted by this change
V52	V53
V95	V94
V98	V98
V100	V125, V180, V491, V492
V108	V109, V125, V128, V171, V180, V600, V605
V115	V115
V143	V141, V142
V153	V151, V152
V206	V207
V208	V208
V217	V218
V245	V246
V257	V97, V98, V99, V104, V120, V414, V415,V417, V454, V455, V457, V496, V497, V1097,V1098, V1099
V259	V120, V497
V263	V98, V99, V144, V154, V264, V265, V266, V271, V1097, V1098, V1099
V264	V52, V53, V59, V75, V342, V343, V411, V426, V451, V466, V498, V499, V1052, V1053, V1059, V1075
V265	V91, V92, V93, V173, V176, V177, V205, V218, V223, V1091, V1092, V1093
V270	V119
V271	V98, V99, V103, V107, V112, V119, V400, V401,V402, V403, V414, V415, V454, V455, V1097, V1098, V1099
V403	V400, V401, V402
V406	V400, V401
V407	V402
V408	V105, V106, V402, V489
V412	V425, V427
V427	V425
V452	V465, V467
V467	V465
V491	V492
V600	V605
V601	V606
V1059	V1075
V1095	V1094
V1098	V1097

The “Parameters impacted” are as parameters which have to be controlled after the main parameter is modified.
For instance whenever V52 is modified you must modify V53 as indicated in the NUMDrive conversion table.

A.2.3 NUMDrive C parameters format and limits

Parameter N°	Type	Resolution	Min value [PU]	Max value [PU]
V20	DWORD	1	0	3h
V21	DWORD	1	0	3h
V22	DWORD	1	0	3h
V23	DINT	1	0	31030
V30	DWORD	1	0	7FFFFFFFh
V32	DWORD	1	1610612736	7FFFFFFFh
V33	DWORD	1	0	7FFFFFFh
V34	DWORD	1	0	7FFFFh
V35	DWORD	1	0	1FFFh
V36	DINT	1	1	16
V37	REAL	0.01	0.2	2
V40	REAL	0.01	0	50
V41	REAL	0.01	1	3000
V42	REAL	0.01	0.1	5000
V43	REAL	0.01	1	120
V44	DINT	1	0	100
V45	REAL	0.01	0	20000
V46	REAL	0.01	0	2000
V50	REAL	0.01	0	Ipeak/141.42
V51	REAL	0.01	0	32767*Ipeak/5656.854
V52	REAL	0.01	2*V264/1000	32767*V264/(0.99*1000)
V53	REAL	0.01	V52	V264*32767/(0.99*1000)
V57	REAL	0.01	0	Ipeak/141.42
V58	REAL	0.01	0	32767*Ipeak/5656.854
V59	REAL	0.01	2*V264/1000	32767*V264/(0.99*1000)
V60	REAL	0.01	0	32767*Ipeak/46340.95
V75	REAL	0.01	V59	V264*32767/(0.99*1000)
V80	REAL	0.01	0	1
V81	REAL	0.01	1	2
V91	DINT	1	4	1000000
V92	DINT	1	4	1000000
V93	DINT	1	5	100000
V94	DINT	1	0	200
V95	DINT	1	0	1000
V96	DINT	1	0	1000
V97	DINT	1	0	200
V98	DINT	1	0	140000*(V263+0.25*(V271-V263))/(V257*V271)
V99	DINT	1	0	140000*(V263+0.25*(V271-V263))/(V257*V271)
V100	REAL	0.01	0	Ipeak/141.42
V101	DINT	1	0	1000
V102	REAL	0.01	0	926790*4/Ipeak
V103	REAL	0.01	2*V271/1000	32767*V271/(1000*0.99)
V104	REAL	0.01	0	32767*V257/(16.237976*2)
V105	REAL	0.01	0	(32767/(Ipeak*0.01503011016))-V408
V106	REAL	0.01	0	(32767/(Ipeak*0.01503011016))-V408
V107	DINT	1	1	3000
V108	REAL	0.01	Ipeak/4715.0	H1a5a/100
V109	REAL	0.01	-(V108)	V108
V110	REAL	0.01	0	Ipeak/141.42

Parameter N°	Type	Resolution	Min value [PU]	Max value [PU]
V111	REAL	0.01	0	926790*4/Ipeak
V112	REAL	0.01	2*V271/1000	32767*V271/(1000*0.99)
V114	REAL	0.01	0	32767
V115	REAL	0.01	Ipeak/4715.0	H1a5a/100
V116	REAL	0.01	0	32767/(Ipeak*0.01503011016)
V117	REAL	0.01	0	32767
V118	REAL	0.01	0	32767
V119	REAL	0.01	0	3
V120	REAL	0.01	0	32767*50*V257/(V259*54.12659)
V121	REAL	0.01	0	1
V122	REAL	0.01	1	2
V123	DINT	1	1	3
V125	DINT	1	0	(100.0*V100)/V108
V128	DINT	1	13	100
V129	DINT	1	13	100
V130	DINT	1	1	500
V131	DINT	1	1	500
V132	DINT	1	1	500
V133	DINT	1	1	500
V134	DINT	1	-2147483648	2147483647
V135	DINT	1	-2147483648	2147483647
V136	DINT	1	-2147483648	2147483647
V137	DINT	1	-2147483648	2147483647
V138	DWORD	1	0	Fh
V139	DWORD	1	0	1FFh
V140	DINT	1	1	500
V141	REAL	0.01	-2147483648	2147483647
V142	REAL	0.01	-2147483648	2147483647
V143	REAL	0.01	-2147483648	2147483647
V144	DINT	1	5	1500
V150	DINT	1	1	500
V151	REAL	0.01	-2147483648	2147483647
V152	REAL	0.01	-2147483648	2147483647
V153	REAL	0.01	-2147483648	2147483647
V154	DINT	1	5	1500
V160	DWORD	1	0	7FFh
V165	DINT	1	0	4000
V166	REAL	0.01	0	32767*1000/(5.656941*Ipeak)
V168	DINT	1	1	1
V169	DWORD	1	0	3h
V170	DWORD	1	0	3Fh
V171	REAL	0.01	-(V108)	V108
V172	REAL	0.01	0	32767*21.5792/Ipeak
V173	REAL	0.01	2*V265/1000	32767*V265/(1000*0.99)
V175	DINT	1	0	400
V176	DINT	1	0	(32767*V265)/1000
V177	REAL	0.01	0.1	10000
V180	DINT	1	0	V100*127/V108
V182	DINT	1	0	200
V183	DINT	1	0	200
V190	DINT	1	1	10000

Parameter N°	Type	Resolution	Min value [PU]	Max value [PU]
V305	DINT	1	0	2147483647
V306	DINT	1	-32767	32767
V307	DINT	1	1	32767
V310	DINT	1	-2147483648	2147483647
V311	DINT	1	-32767	32767
V312	DINT	1	1	32767
V313	DINT	1	-2147483648	2147483647
V314	DWORD	1	0	3Fh
V315	DWORD	1	0	3Fh
V317	REAL	0.01	1	9.2
V318	REAL	0.01	2	9.2
V340	DINT	1	0	4
V341	DINT	1	0	4
V342	REAL	0.01	50	3000.00*100.0/V264
V343	REAL	0.01	50	3000.00*100.0/V264
V344	DINT	1	-2147483648	2147483647
V345	DINT	1	-2147483648	2147483647
V346	DINT	1	0	9999
V347	DINT	1	0	9999
V348	DWORD	1	0	1FFFFFh
V349	DWORD	1	0	1FFFFFh
V350	DWORD	1	0	1h
V351	DWORD	1	0	1h
V352	DWORD	1	0	1h
V353	REAL	0.01	1	2147483000/100
V355	DINT	1	0	32767
V356	DINT	1	1	32767
V359	DINT	1	0	32767
V360	DINT	1	1	131072
V361	DINT	1	1	2147483647
V362	DINT	1	0	32767
V363	DINT	1	0	32767
V364	DINT	1	0	32767
V365	DINT	1	1	32767
V367	DINT	1	0	2147483647/4
V368	DINT	1	1	32767
V370	DINT	1	1	131072
V371	DINT	1	1	2147483647
V372	DINT	1	0	32767
V373	DINT	1	0	32767
V374	DINT	1	0	32767
V375	DINT	1	1	32767
V376	DINT	1	0	32767
V377	DINT	1	0	2147483647/4
V378	DINT	1	1	32767
V380	DINT	1	1	8000
V381	DINT	1	1	32767
V382	DINT	1	0	2147483647
V383	DINT	1	0	32767
V384	DINT	1	0	32767
V385	DINT	1	0	32767

Parameter N°	Type	Resolution	Min value [PU]	Max value [PU]
V386	DINT	1	0	32767
V390	DINT	1	1	8000
V391	DINT	1	1	32767
V392	DINT	1	0	2147483647
V393	DINT	1	0	32767
V394	DINT	1	0	2147483647
V395	DINT	1	0	32767
V396	DINT	1	0	32767
V400	DINT	1	0	32767*50*V403/(3.2768*V406*V271)
V401	DINT	1	0	32767*50*V403/(3.2768*V406*V271)
V402	DINT	1	0	(32767*V403/65.536)-V408*1000*V407
V403	REAL	0.01	133774.75*3276.8*(V271/50)/ (Ipeak*32767.00)	32767
V404	DINT	1	0	32767
V405	DINT	1	0	32767
V406	DINT	1	1	16
V407	DINT	1	3	4
V408	REAL	0.01	0	20
V409	DINT	1	0	700/1.4142135
V410	REAL	0.01	0	127.9
V411	REAL	0.01	2*V264/1000	V264*32.767/0.99
V412	DINT	1	2	100
V413	DINT	1	0	32767
V414	DINT	1	0	32767*100/(0.3431457*V257*V271)
V415	DINT	1	0	32767*100/(0.3431457*V257*V271)
V417	DINT	1	0	32767/(V257*0.21845333)
V425	DINT	1	1	20
V426	REAL	0.01	0	32767*Ipeak/(7592.3856*V264)
V427	REAL	0.01	0	13107*Ipeak/(V412*13107.2*1.4142135)
V428	REAL	0.01	0	Ipeak/141.42
V448	DINT	1	0	32767
V450	REAL	0.01	0	127.9
V451	REAL	0.01	2*V264/1000	V264*32.767/0.99
V452	DINT	1	0	100
V453	DINT	1	0	32767
V454	DINT	1	0	32767*100/(0.3431457*V257*V271)
V455	DINT	1	0	32767*100/(0.3431457*V257*V271)
V457	DINT	1	0	32767/(V257*0.21845333)
V465	DINT	1	1	20
V466	REAL	0.01	0	32767*Ipeak/(7592.3856*V264)
V467	REAL	0.01	0	13107*Ipeak/(V452*13107.2*1.4142135)
V468	REAL	0.01	0	Ipeak/141.42
V470	REAL	0.01	0	32767
V471	REAL	0.01	0	32767
V472	DINT	1	0	32767
V473	DINT	1	0	32767
V480	REAL	0.01	0	100
V481	DWORD	1	0	7h
V488	DINT	1	0	32767*((234.5+20.0)*2*133772.76)/ (Ipeak*65.536*(234.5+120.0))
V489	REAL	0.01	0	(32767/(Ipeak*0.01503011016))-V408
V491	REAL	0.01	-V100	0

Parameter N°	Type	Resolution	Min value [PU]	Max value [PU]
V492	REAL	0.01	V491	0
V496	DINT	1	0	32767/(0.21845333*V257)
V497	REAL	0.01	0	32767/(109.22667*V257/V259)
V498	REAL	0.01	0	32767*Ipeak/(7592.3856*V264)
V499	REAL	0.01	0	32767*Ipeak/(7592.3856*V264)
V500	DWORD	1	0	1FFh
V501	DINT	1	-32767	32767
V502	DINT	1	0	65535
V503	DINT	1	0	65535
V504	DINT	1	-32767	32767
V505	DINT	1	-32767	32767
V506	DINT	1	-32767	32767
V507	DINT	1	0	32767
V508	DINT	1	-32767	32767
V509	DINT	1	-32767	32767
V511	DINT	1	-32767	32767
V512	DINT	1	0	65535
V513	DINT	1	0	65535
V514	DINT	1	-32767	32767
V515	DINT	1	-32767	32767
V516	DINT	1	-32767	32767
V517	DINT	1	0	32767
V518	DINT	1	-32767	32767
V519	DINT	1	-32767	32767
V521	DINT	1	-32767	32767
V522	DINT	1	0	65535
V523	DINT	1	0	65535
V524	DINT	1	-32767	32767
V525	DINT	1	-32767	32767
V526	DINT	1	-32767	32767
V527	DINT	1	0	32767
V528	DINT	1	-32767	32767
V529	DINT	1	-32767	32767
V531	DINT	1	-32767	32767
V532	DINT	1	0	65535
V533	DINT	1	0	65535
V534	DINT	1	-32767	32767
V535	DINT	1	-32767	32767
V536	DINT	1	-32767	32767
V537	DINT	1	0	32767
V538	DINT	1	-32767	32767
V539	DINT	1	-32767	32767
V541	DINT	1	-32767	32767
V542	DINT	1	0	65535
V543	DINT	1	0	65535
V544	DINT	1	-32767	32767
V545	DINT	1	-32767	32767
V546	DINT	1	-32767	32767
V547	DINT	1	0	32767
V548	DINT	1	-32767	32767
V549	DINT	1	-32767	32767

Parameter N°	Type	Resolution	Min value [PU]	Max value [PU]
V551	DINT	1	-32767	32767
V552	DINT	1	0	65535
V553	DINT	1	0	65535
V554	DINT	1	-32767	32767
V555	DINT	1	-32767	32767
V556	DINT	1	-32767	32767
V557	DINT	1	0	32767
V558	DINT	1	-32767	32767
V559	DINT	1	-32767	32767
V600	DINT	1	0	Ipeak/(V108*1.4142)
V601	DINT	1	10	1500
V602	REAL	0.01	100	500
V603	REAL	0.01	0	50
V604	DINT	1	0	32767*133772.76/(Ipeak*65.536*1.4)
V605	DINT	1	0	V600*91/100
V606	DINT	1	V601/2	1500
V1045	REAL	0.01	0	20000
V1046	REAL	0.01	0	2000
V1050	REAL	0.01	0	Ipeak/141.42
V1051	REAL	0.01	0	32767*Ipeak/5656.854
V1052	REAL	0.01	2*V264/1000	32767*V264/(0.99*1000)
V1053	REAL	0.01	P1052	V264*32767/(0.99*1000)
V1057	REAL	0.01	0	Ipeak/141.42
V1058	REAL	0.01	0	32767*Ipeak/5656.854
V1059	REAL	0.01	2*V264/1000	32767*V264/(0.99*1000)
V1060	REAL	0.01	0	32767*Ipeak/46340.95
V1075	REAL	0.01	P1059	V264*32767/(0.99*1000)
V1091	DINT	1	4	1000000
V1092	DINT	1	4	1000000
V1093	DINT	1	5	100000
V1094	DINT	1	0	200
V1095	DINT	1	0	1000
V1096	DINT	1	0	1000
V1097	DINT	1	0	200
V1098	DINT	1	0	140000*(V263+0.25*(V271-V263))/(V257*V271)
V1099	DINT	1	0	140000*(V263+0.25*(V271-V263))/(V257*V271)
V1301	DINT	1	0	50000
V1305	DINT	1	0	2147483647
V1306	DINT	1	-32767	32767
V1307	DINT	1	1	32767
V1310	DINT	1	-2147483648	2147483647
V1311	DINT	1	-32767	32767
V1312	DINT	1	1	32767
V1313	DINT	1	-2147483648	2147483647
V1353	REAL	0.01	1	2147483000/100
V1359	DINT	1	0	32767
V1362	DINT	1	0	32767
V1372	DINT	1	0	32767
V1376	DINT	1	0	32767
V1383	DINT	1	0	32767
V1393	DINT	1	0	32767

A.3 NUMDrive X parameters Conversion rules, Dependencies, Limits

- Modifying a drive parameter requires to define its value in both Physical and Internal units. The following table describes the conversion rule between them.
- The Halt column indicates if the drive must be in Halt or not to accept the new value Please note that some of those parameters are not visible with Flexium Tools or WPM; they are listed here for dependencies reasons.
- The second table defines the dependencies between parameters. That is a parameter value has an effect on the values of other parameters
- The third table defines the limit values of all parameters. When, changing the value of a parameter, the programmer must pay attention not to exceed the limit of all depending parameters.

A.3.1 Conversion rules table

Param. N°	Type	Halt	[PU]	Resolution	Conversion rule from Vxx in physical unit to Vxx in internal unit	Description
V25	DWORD	Yes	[ ]		V25	Port Ix1 configuration
V26	DWORD	Yes	[ ]		V26	Port Ix2 configuration
V27	DWORD	Yes	[ ]		V27	Port Ix3 configuration
V28	DWORD	Yes	[ ]		V28	Port I/Ox4 configuration
V29	DWORD	Yes	[ ]		V29	Port Ox5 configuration
V30	DWORD	Yes	[ ]	1	V30	General drive configuration
V32	DWORD	Yes	[ ]	1	V32	Config. braking following alarms [1=Brake]
V33	DWORD	Yes	[ ]	1	V33	Alarm reset configuration [1=Reset]
V34	DWORD	Yes	[ ]	1	V34	Configuration DF1/DF2 relay [1=Break]
V35	DWORD	Yes	[ ]	1	V35	SW functions [0=Not available][1=Available]
V36	UDINT	Yes	[ ]	1	V36	FSoE address SAMX
V37	REAL	Yes	s	0.01	V37*1000	Delay time of power detection
V40	REAL	No	%	0.01	V40*65536/100	Speed reached dynamic threshold
V41	REAL	No	mm/min	0.01	V41*163.840*(V263/50)	Motor zero speed threshold, enable, brake
V42	REAL	No	rpm	0.01	V42*1342.18*(V263/50)	Motor zero speed threshold, enable, brake
V43	REAL	Yes	s	0.01	V43*1000	Braking Timeout on config. V30, V32
V45	REAL	No	mm/min	0.01	V45*163.840*(V263/50)	Speed dynamic check [0]
V46	REAL	No	rpm	0.01	V46*1342.18*(V263/50)	Speed dynamic check [0]
V50	REAL	No	Arms	0.01	V50*1853609.716/ (Ipeak*100)	Torque current limit [0]
V51	REAL	No	mArms/ rpm	0.01	V51*90509.668*(50/V263)/ (Ipeak*100)	Proportional speed loop gain [0]
V52	REAL	No	ms	0.01	32767*(2*V263)/(V52*1000)	Integral speed loop gain [0]
V57	REAL	No	Arms	0.01	V57*1853609.716/ (Ipeak*100)	Torque current limit h.r. [0]
V58	REAL	No	mArms/ rpm	0.01	V58*90509.668*(50/V263)/ (Ipeak*100)	Proportional speed loop gain h.r. [0]
V59	REAL	No	ms	0.01	32767*(2*V263)/(V59*1000)	Integral speed loop gain h.r. [0]
V60	REAL	No	Arms/m/ min	0.01	V60*741455.2*(50/V263)/ (Ipeak*100)	Proportional speed loop gain [0]
V80	REAL	No	[ ]	0.01	32768.0*V80	Set Point Weight speed loop gain
V81	REAL	No	[ ]	0.01	16384.0*V81	Anti-windup speed loop gain
V91	UDINT	No	rpm/s	1	V91*(4*V263)*1.34218* (V263/50)/1000	Acceleration/deceleration ramp [0]
V92	UDINT	No	rpm/s	1	V92*(4*V263)*1.34218* (V263/50)/1000	Acceleration/deceleration ramp h.r.[0]

Param. N°	Type	Halt	[PU]	Resolution	Conversion rule from Vxx in physical unit to Vxx in internal unit	Description
V93	UDINT	No	mm/s^2	1	V93*60*(4*V263)*0.163840* (V263/50)/1000	Acceleration/deceleration ramp [0]
V94	UDINT	No	%	1	V94*163840*V95/100	Speed reference Limit (% V95) [0]
V95	UDINT	No	m/min	1	V95*163840*(V263/50)	Maximum motor speed [0]
V96	UDINT	No	m/min	1	V96*163840*(V263/50)	Overspeed alarm threshold [0]
V97	UDINT	No	%	1	V97*1342.18*V98/100	Speed reference Limit (% V98) [0]
V98	UDINT	No	rpm	1	V98*1342.18*(V263/50)	Maximum motor speed [0]
V99	UDINT	No	rpm	1	V99*1342.18*(V263/50)	Overspeed alarm threshold [0]
V100	REAL	No	Arms	0.01	V100*1853609.716/ (Ipeak*100)	Deliverable current limit
V101	UDINT	No	Vrms	1	V101*8	Voltage limits Vd and Vq
V102	REAL	No	Vrms/ Arms	0.01	(V102*(Ipeak*100)/28.28427)/4	Proportional Iq current loop gain
V103	REAL	No	ms	0.01	32767*V271/(1000*V103)	Integral Iq current loop gain
V104	REAL	No	mVrms/ rpm	0.001	V104*16.237976*2*(50/V263)/V257	Motor EMF phase compensation
V105	REAL	No	mH	0.01	(V105+V408)* (Ipeak*100)*0.01503011016* (50/V263)	Compensation of motor Lq inductance
V106	REAL	No	mH	0.01	(V106+V408)* (Ipeak*100)*0.01503011016* (50/V263)	Compensation of motor Ld inductance
V107	UDINT	Yes	Hz	1	(0.102943708*V271*V107)/ (1+0.00000314159*V271*V107)	Lowpass Filter elettrical speed
V108	REAL	No	Arms	0.01	V108*1853609.716/(Ipeak*100)	Motor rated current [I^2t]
V109	REAL	No	Arms	0.01	V109*1853609.716/(Ipeak*100)	Vertical load compensation current
V110	REAL	No	Arms	0.01	V110*1853609.716/(Ipeak*100)	Deliverable current limit h.r.
V111	REAL	No	Vrms/ Arms	0.01	(V111* (Ipeak*100)/28.28427)/4	Proportional Iq current gain h.r.
V112	REAL	No	ms	0.01	32767*V271/(1000*V112)	Integral Iq current gain h.r.
V114	REAL	No	mH	0.01	((V105*1.00/V407)+V408)* (Ipeak*100)*0.01503011016* (50/V263)	Motor Lq inductance compensation h.r.
V115	REAL	No	Arms	0.01	V115*1853609.716/ (Ipeak*100)	Motor rated current [I^2t] h.r.
V116	REAL	No	mH	0.01	V116*(Ipeak*100)*0.01503011016* (50/V263)	Lm (for compensation)
V117	REAL	No	mH	0.01	V116*(Ipeak*100)*0.01503011016* (50/V263)/V407	Lm (for compensation) h.r.
V118	REAL	No	mH	0.01	((V106*1.00/V407)+V408)* (Ipeak*100)*0.01503011016* (50/V263)	Motor Ld inductance compensation h.r.
V119	REAL	No	cycles	0.01	V119*(V271/50)*1024	Phase lead
V120	REAL	No	Vrms/ m/s	0.01	V120*V259*54.12659/(V257*V263)	Linear Motor EMF phase compensation
V121	REAL	No	[ ]	0.01	(1.0-V121)*((Ipeak*100)* (Ipeak*100))/(100.0*V123*V123)	Iq gain reduction at V123 current
V122	REAL	No	[ ]	0.01	(1.0-V122)*((Ipeak*100)* (Ipeak*100))/(100.0*V124*V124)	Iq gain reduction at V124 current h.r.
V123	REAL	No	Arms	0.01	V123*1853609.716/ (Ipeak*100)	Maximal motor current
V124	REAL	No	Arms	0.01	V124*1853609.716/ (Ipeak*100)	Maximal motor current h.r.
V125	UDINT	No	%	1	V125*V108*18536.09716/ (Ipeak*100)	Current percentage V108 for EPSM
V126	UDINT	No	%	1	(V126*V108*13107)/(Ipeak*100)	Current percentage V108 for EPS
V128	UDINT	No	%	1	V128*V108*18536.09716/ (Ipeak*100)	Locked rotor current I^2t (% V108)
V129	UDINT	No	%	1	V129*V115*18536.09716/ (Ipeak*100)	Locked rotor current I^2t (% V115) h.r.
V130	UDINT	No	[ ]	1	V130	Associate MP with TP1
V131	UDINT	No	[ ]	1	V131	Associate MP with TP2

Param. N°	Type	Halt	[PU]	Resolution	Conversion rule from Vxx in physical unit to Vxx in internal unit	Description
V132	UDINT	No	[ ]	1	V132	Associate MP with TP3
V133	UDINT	No	[ ]	1	V133	Associate MP with TP4
V135	DWORD	Yes	[ ]	1	V135	Configuration of braking relay
V136	UDINT	No	ms	1	V136	Brake ON delay
V137	UDINT	No	ms	1	V137	Brake OFF delay
V140	UDINT	No	[ ]	1	V140	Associate MP with programmable relay 1
V141	DINT	No	[ ]	1	V141	Programmable relay 1, threshold 1
V142	DINT	No	[ ]	1	V142	Programmable relay 1, threshold 2
V144	UDINT	No	ms	1	V144/(V263/1000)	Programm. relay 1, tripping time constant
V145	DWORD	No	[ ]	1	V145	Configuration of programmable relay 1
V150	UDINT	No	[ ]	1	V150	Associate MP with programmable relay 2
V151	DINT	No	[ ]	1	V151	Programmable relay 2, threshold 1
V152	DINT	No	[ ]	1	V152	Programmable relay 2, threshold 2
V154	UDINT	No	ms	1	V154/(V263/1000)	Programm. relay 2, tripping time constant
V155	DWORD	No	[ ]	1	V155	Configuration of programmable relay 2
V160	DWORD	Yes	[ ]	1	V160	Special applications selection
V162	UDINT	Yes	[ ]	1	V162	AP02: Mechanical ratio: load turns
V163	UDINT	Yes	[ ]	1	V163	AP02: Mechanical ratio: sensor turns
V164	DWORD	Yes	[ ]	1	V164	AP02: configuration
V165	UDINT	No	mm/min	1	V165*163.840*(V263/50)	AP03 Mst: Max compens. torque equalizer
V166	REAL	No	mm/min/ Arms	0.01	V166* (Ipeak*100)*5.656941* (V263/50)/1000	AP03 Mst: Proportional torq. equaliz. gain
V169	DWORD	Yes	[ ]	1	V169	AP03: Anti-backlash role
V170	DWORD	Yes	[ ]	1	V170	AP03: Anti-backlash configuration
V171	REAL	No	Arms	0.01	-2*V171*1853609.716/(Ipeak*100)	AP03 Master: Preload Tandem current
V172	REAL	No	rpm/ Arms	0.01	V172* (Ipeak*100)* (V263/50)/21.5792	AP03 Mst: Proportional torq. equaliz. gain
V173	REAL	No	ms	0.01	32767*(4*V263)/(1000*V173)	AP03 Mst: Integral torque equalizer gain
V175	UDINT	No	rpm	1	V175*1342.18*(V263/50)	AP03 Mst: Max compens. torque equalizer
V176	UDINT	No	ms	1	(V176*1000)/(4*V263)	AP03 Mst: T.out A52.41 (saturated trq eq.)
V177	REAL	No	Arms/s	0.01	2*V177*1.853609716*65536* (4*V263)/(Ipeak*100)	AP03: Master: Preload ramp speed
V180	UDINT	Yes	[ ]	1	V180	AP04: Torque duplication total drives nbr
V181	UDINT	Yes	[ ]	1	V181	AP04: Torque duplication address
V182	DWORD	Yes	[ ]	1	V182	AP04: Torque duplication configuration
V185	UDINT	Yes	[ ]	1	V185	AP05: Winding duplication total drives nbr
V186	UDINT	Yes	[ ]	1	V186	AP05: Winding duplication address
V187	DWORD	Yes	[ ]	1	V187	AP05: Winding duplication configuration
V190	UDINT	No	[ ]	1	V190	AP06: Max displacement meas. S1-S2 Ph.U.
V191	UDINT	No	ms	1	V191	AP06: Max duration measure displac. S1-S2
V200	DWORD	Yes	[ ]	1	V200	AP08: Inter-drive DEMX data exchange role
V201	DWORD	Yes	[ ]	1	V201	AP08: Inter-drive DEMX data exchange configuration
V202	UDINT	Yes	[ ]	1	V202	AP08: Inter-drive DEMX data exchange MP
V217	REAL	No	[ ]	0.01	32.0*V217	AP12: Active damping 2 K1
V218	REAL	No	Hz	0.01	(0.102943708*(4*V263)*V218)/(V217+0.00000314159*(4*V263)*V218)	AP12: Active damping 2 filter
V219	UDINT	No	[ ]	1	32767.0*(V217-0.00000314159*(4*V263)*V218)/(V217+0.00000314159*(4*V263) *V218)	AP12: coeff.A Active damping 2 filter
V220	DWORD	Yes	[ ]	1	V220	Reference configuration
V221	DWORD	Yes	[ ]	1	V221	Enable configurations

Param. N°	Type	Halt	[PU]	Resolution	Conversion rule from Vxx in physical unit to Vxx in internal unit	Description
V238	DWORD	Yes	[ ]		V238	Filter on squared counter input S1
V239	DWORD	Yes	[ ]		V239	Filter on squared counter input S2
V240	REAL	No	s	0.01	(0.5+(262144*0.2048/V240))	I^2t motor: first time constant
V241	UDINT	No	s	1	(0.5+(1048576*0.2048/V241))	I^2t motor: second time constant
V242	REAL	No	Arms	0.01	4096*V108/V123	Ratio nominal motor current / maximum motor current
V243	REAL	No	Arms	0.01	4096*V115/V124	Ratio nominal motor current / maximum motor current h.r.
V244	REAL	No	Arms	0.01	4096*V108*V128/(100*V123)	Ratio locked rotor current / maximum motor current
V245	REAL	No	Hz	0.01	V245*13.1072*(V263/50)	I^2t motor: threshold change of time const.
V246	REAL	No	Hz	0.01	V246*13.1072*(V263/50)	I^2t motor: frequency threshold hysteresis (V245)
V247	REAL	No	Arms	0.01	4096*V115*V129/(100*V124)	Ratio locked rotor current / maximum motor current h.r.
V248	REAL	Yes	s	0.01	(0.5+(262144*0.2048/V248))	I^2t drive: first time constant
V249	UDINT	Yes	s	1	(0.5+(1048576*0.2048/V249))	I^2t drive: second time constant
V250	DWORD	Yes	[ ]	1	V250	Motor XML information
V251	UDINT	Yes	[ ]	1	V251	Parameter compatibility
V252	STRING	Yes	[ ]		V252	Drive type
V253	DWORD	Yes	[ ]	1	V253	Drive size
V255	STRING	Yes	[ ]		V255	Motor type
V256	STRING	Yes	[ ]		V256	Motor power connection type
V257	UDINT	No	[ ]	1	V257	Motor poles pairs number
V259	REAL	Yes	mm	0.01	V259	Linear motor pole pitch [180 degrees el.]
V263	UDINT	Yes	us	25	V263	System sampling time
V268	UDINT	Yes	mDeg	1	V268*65.536/360.0	Electrical phase offset Hall effect sensor
V269	UDINT	Yes	mDeg	1	V269*65.536/360.0	Position offset for electrical phasing
V270	UDINT	Yes	Ohm	1	V270	Motor thermal sensor PTC-NTC value
V271	UDINT	Yes	us	25	V271/V263	Current loop sampling time
V272	DWORD	Yes	[ ]	1	V272	Motor thermal sensor configuration
V273	DWORD	Yes	[ ]	1	V273	Electrical phase offset management
V301	UDINT	No	mm/min/ mm	1	V301	Proportional position gain [0]
V305	UDINT	Yes	[ ]	1	V305	Measure module for rot. axis/spindle abs[0]
V306	DINT	Yes	[ ]	1	V306	S1 sensor MULTI coefficient [0]
V307	UDINT	Yes	[ ]	1	V307	S1 sensor DIVI coefficient [0]
V310	DINT	Yes	[ ]	1	V310	S1 sensor offset in CNC Phys. Unit [0]
V311	DINT	Yes	[ ]	1	V311	S2 sensor MULTI coefficient [0]
V312	UDINT	Yes	[ ]	1	V312	S2 sensor DIVI coefficient [0]
V313	DINT	Yes	[ ]	1	V313	S2 sensor offset in CNC Phys. Unit [0]
V314	DWORD	Yes	[ ]	1	V314	S1 sensor sinusoidal / TTL signals range
V315	DWORD	Yes	[ ]	1	V315	S2 sensor sinusoidal / TTL signals range
V317	REAL	Yes	V	0.01	V317*100.0	S1 sensor power supply voltage
V318	REAL	Yes	V	0.01	V318*100.0	S2 sensor power supply voltage
V320	DWORD	Yes	[ ]		V320	S1 sensor incremental traces configuration
V321	DWORD	Yes	[ ]		V321	S2 sensor incremental traces configuration
V322	DWORD	Yes	[ ]		V322	S1 sensor communication protocol
V323	DWORD	Yes	[ ]		V323	S2 sensor communication protocol
V324	DWORD	Yes	[ ]	1	V324	S1 sensor initialization configuration
V325	DWORD	Yes	[ ]	1	V325	S2 sensor initialization configuration
V326	DWORD	No	[ ]	1	V326	S1 sensor run-time options

Param. N°	Type	Halt	[PU]	Resolution	Conversion rule from Vxx in physical unit to Vxx in internal unit	Description
V327	DWORD	No	[ ]	1	V327	S2 sensor run-time options
V328	DWORD	No	[ ]	1	V328	S1 Sensor alarms handling
V329	DWORD	No	[ ]	1	V329	S2 Sensor alarms handling
V330	DWORD	Yes	[ ]		V330	S1 sensor homing configuration
V331	DWORD	Yes	[ ]		V331	S2 sensor homing configuration
V340	UDINT	No	[ ]	1	V340	Step of speed calculation S1 sensor
V341	UDINT	No	[ ]	1	V341	Step of speed calculation S2 sensor
V342	REAL	No	Hz	0.01	(0.102943708*(2*V263)*V342)/ (1+0.00000314159*(2*V263)*V342)	Speed filter by S1 sensor
V343	REAL	No	Hz	0.01	(0.102943708*(2*V263)*V343)/ (1+0.00000314159*(2*V263)*V343)	Speed filter by S2 sensor
V344	UDINT	Yes	[ ]	1	V344	S1 sensor code ID
V345	UDINT	Yes	[ ]	1	V345	S2 sensor code ID
V350	DWORD	Yes	[ ]	1	V350	Position loop sensor config.
V353	REAL	Yes	[ ]	0.01	V353*100	Period of position sensor/motor turn [0]
V359	UDINT	Yes	[ ]	1	(65536*3*4*1024/V353)	k_velrot120
V360	UDINT	Yes	[ ]	1	V360	Period by turn S1 sensor
V361	UDINT	Yes	[ ]	1	V361	Absolute resolution by turn S1 sensor
V362	UDINT	No	[ ]	1	V301*1.31072*327.68*4*(2*V263)/ V360	kpposrot10
V363	UDINT	Yes	[ ]	1	(65536*3*1024*4/V360)	k_velrot10
V364	UDINT	Yes	[ ]	1	V361/V360	sinuarot10
V365	UDINT	Yes	[ ]	1	V365	Number of absolute S1 sensor turns
V368	UDINT	Yes	[ ]	1	V368	Intervals between S1 coded references
V369	UDINT	Yes	[ ]	1	V369	Multiplier of S1 coded references
V370	UDINT	Yes	[ ]	1	V370	Period by turn S2 sensor
V371	UDINT	Yes	[ ]	1	V371	Absolute resolution by turn S2 sensor
V372	UDINT	No	[ ]	1	V301*1.31072*327.68*4*(2*V263)/ V353	kpposrot120
V373	UDINT	Yes	[ ]	1	(65536*3*1024*4/V370)	k_velrot20
V374	UDINT	Yes	[ ]	1	V371/V370	sinuarot20
V375	UDINT	Yes	[ ]	1	V375	Number of absolute S2 sensor turns
V376	UDINT	No	[ ]	1	V301*1.31072*327.68*4*(2*V263)/ V370	kpposrot20
V378	UDINT	Yes	[ ]	1	V378	Intervals between S2 coded references
V379	UDINT	Yes	[ ]	1	V379	Multiplier of S2 coded references
V380	UDINT	Yes	um	1	V380	Period of linear S1 sensor
V381	UDINT	Yes	nm	1	V381	Absolute resolution of linear S1 sensor
V382	UDINT	Yes	[ ]	1	(V259*2000/V380)+0.1	npslin10
V383	UDINT	No	[ ]	1	V301*5.24288*V380*4* (2*V263/100)	kpposlin10
V384	UDINT	Yes	[ ]	1	(24576*V380*4)	k_vellin10
V385	UDINT	Yes	[ ]	1	V380*1000/V381	sinualin10
V386	UDINT	Yes	[ ]	1	(V259*2000000/V381)+0.1	ralinppm10
V389	UDINT	Yes	[ ]	1	V389	S1 sensor serial position datum bit size
V390	UDINT	Yes	um	1	V390	Period of linear S2 sensor
V391	UDINT	Yes	nm	1	V391	Absolute resolution of linear S2 sensor
V392	UDINT	Yes	[ ]	1	(V259*2000/V390)+0.1	npslin20
V393	UDINT	No	[ ]	1	V301*5.24288*V390*4* (2*V263/100)	kpposlin20

Param. N°	Type	Halt	[PU]	Resolution	Conversion rule from Vxx in physical unit to Vxx in internal unit	Description
V394	UDINT	Yes	[ ]	1	(24576*V390*4)	k_vellin20
V395	UDINT	Yes	[ ]	1	V390*1000/V391	sinualin20
V396	UDINT	Yes	nm	1	(V259*2000000/V391)+0.1	ralinppm20
V399	UDINT	Yes	[ ]	1	V399	S2 sensor serial position datum bit size
V400	UDINT	No	mOhm	1	V400*3.2768*V406*(V271/50)/V403	Rs at 120 °C
V401	UDINT	No	mOhm	1	V401*3.2768*V406*(V271/50)/V403	Rr at 150 °C
V402	UDINT	No	uH	1	(V402+V408*1000)*65.536/V403	sigmaLs
V403	REAL	No	mH	0.01	V403	Lr
V404	UDINT	No	[ ]	1	133774.75*3276.8*(V271/50)/ ((Ipeak*100)*V403)	[U.I.=(Ki/Kv) /Lr * (Ts*2^16)=a04] Ki/Kv
V405	UDINT	No	[ ]	1	133774.75*3276.8*V407*(V271/50)/ ((Ipeak*100)*V403)	[U.I.=a04b*ratio_gamma=a04a] a04a
V406	UDINT	No	[ ]	1	V406	twotosh0
V407	UDINT	No	[ ]	1	V407	ratio_gamma
V408	REAL	No	mH	0.01	V408	Additional extern inductance
V409	UDINT	No	Vrms	1	V409*8	ST flux-weakening voltage
V410	REAL	No	[ ]	0.01	V410*256	Proportional gain flux loop
V411	REAL	No	ms	0.01	32.767*(2*V263)/V411	Integral gain flux loop
V412	UDINT	No	%	1	0.64*V412	Lr saturation level
V413	UDINT	No	[ ]	1	32768/(0.64*V412)	ksatinv
V414	UDINT	No	rpm	1	V414*0.3431457*V257*(V271/100)	Observer engagement speed
V415	UDINT	No	rpm	1	V415*0.3431457*V257*(V271/100)	Observer engagement hysteresis
V417	UDINT	No	rpm	1	V417*V257*0.21845333*(V263/50)	Defluxing starting speed
V425	UDINT	No	%	1	V425*V427 *V412*131.072*1.4142135/ (Ipeak*100)	Min_flux as a %age of V427
V426	REAL	No	Arms/ Vrms*s	0.01	V426*7592.3856*100*(V263/50)/ (Ipeak*100)	K deflux
V427	REAL	No	Arms	0.01	V427*V412*13107.2*1.4142135/ (Ipeak*100)	Magnetization current
V428	REAL	No	Arms	0.01	V428*1853609.716/(Ipeak*100)	Magnetization current limit
V434	REAL	No	Vrms/ Arms	0.01	(V434*(Ipeak*100)/28.28427)/4	Proportional Id current loop gain
V435	REAL	No	ms	0.01	32767*V271/(1000*V435)	Integral Id current loop gain
V448	UDINT	No	uH	1	(V402+V407*V408*1000)*65.536/ V403	sigma Ls h.r.
V450	REAL	No	[ ]	0.01	V450*256	Proportional flux loop gain h.r.
V451	REAL	No	ms	0.01	32.767*(2*V263)/V451	Integral flux loop gain h.r.
V452	UDINT	No	%	1	0.64*V452	Lr saturation level h.r.
V453	UDINT	No	[ ]	1	32768/(0.64*V452)	ksatinv h.r.
V454	UDINT	No	rpm	1	V454*0.3431457*V257*(V271/100)	Observer engagement speed h.r.
V455	UDINT	No	rpm	1	V455*0.3431457*V257*(V271/100)	Observer engagement hysteresis h.r.
V457	UDINT	No	rpm	1	V457*V257*0.21845333*(V263/50)	Defluxing starting speed h.r.
V465	UDINT	No	%	1	V465*V467* V452*131.072*1.4142135/ (Ipeak*100)	Min_flux as a %age of V467 h.r.
V466	REAL	No	Arms/ Vrms*s	0.01	V466*7592.3856*100*(V263/50)/ (Ipeak*100)	K deflux h.r.
V467	REAL	No	Arms	0.01	V467*V452*13107.2*1.4142135/ (Ipeak*100)	Magnetization current h.r
V468	REAL	No	Arms	0.01	V468*1853609.716/(Ipeak*100)	Magnetization current limit h.r.

Param. N°	Type	Halt	[PU]	Resolution	Conversion rule from Vxx in physical unit to Vxx in internal unit	Description
V470	REAL	No	mH	0.01	((V402/1000.0)+V408)* (Ipeak*100)*0.01503011016* (50/V263)	Sigma Ls flux-weakening
V471	REAL	No	mH	0.01	((V402/(V407*1000.0))+V408)* (Ipeak*100)*0.01503011016* (50/V263)	Sigma Ls flux-weakening h.r.
V472	UDINT	No	[ ]	1	(V400+V401)* (Ipeak*100)* 65.536/133772.76	Total resistance power lost Ri2
V473	UDINT	No	[ ]	1	(V400+V401)*(Ipeak*100)*65.536/ (V407*133772.76)	Total resistance power lost Ri2 h.r.
V474	REAL	No	Vrms/ Arms	0.01	(V474*(Ipeak*100)/28.28427)/4	Proportional Id current gain h.r.
V475	REAL	No	ms	0.01	32767*V271/(1000*V475)	Integral Id current gain h.r.
V479	REAL	No	Hz	0.01	(0.102943708*(4*V263)*V479)/ (1.0+0.00000314159*(4*V263) *V479)	Low-pass filtering of Delivered Power MP6 and LOAD MP11
V480	REAL	No	%	0.01	V480*655.35	Correction factor for MP6
V481	DWORD	Yes	[]	1	V481	Control mode selection
V488	UDINT	No	mOhm	1	V488* (Ipeak*100)*65.536* (234.5+120.0)/ ((234.5+20.0)*2*133772.76)	Resistance tt at 20 °C
V489	REAL	No	mH	0.01	(V408+V489)* (Ipeak*100)* 0.01503011016*(50/V263)	Motor Lq inductance
V491	REAL	No	Arms	0.01	V491*1853609.716/(Ipeak*100)	Id_Max (defluxing parameter)
V492	REAL	No	Arms	0.01	V492*1853609.716/(Ipeak*100)	Id current without defluxing
V496	UDINT	No	rpm	1	V496*V257*0.21845333*(V263/50)	Defluxing starting speed
V497	REAL	No	m/min	0.01	V497*109.22667*V257*(V263/50)/ V259	Defluxing starting speed
V498	REAL	No	Arms/ Vrms*s	0.01	V498*7592.3856*100*(V263/50)/ (Ipeak*100)	K Iq reduction
V499	REAL	No	Arms/ Vrms*s	0.01	V499*7592.3856*100*(V263/50)/ (Ipeak*100)	K flux-weakening Id
V500	DWORD	No	[ ]	1	V500	Filter configuration
V501	DINT	No	[ ]	1	V501	ph11 (FILTER 1 for torque current ref.)
V502	UDINT	No	[ ]	1	V502	ph12 (FILTER 1 for torque current ref.)
V503	UDINT	No	[ ]	1	V503	ph21 (FILTER 1 for torque current ref.)
V504	DINT	No	[ ]	1	V504	ph22 (FILTER 1 for torque current ref.)
V505	DINT	No	[ ]	1	V505	ga1 (FILTER 1 for torque current ref.)
V506	DINT	No	[ ]	1	V506	ga2 (FILTER 1 for torque current ref.)
V507	UDINT	No	[ ]	1	V507	r*r (FILTER 1 for torque current ref.)
V508	DINT	No	[ ]	1	V508	C1 (FILTER 1 for torque current ref.)
V509	DINT	No	[ ]	1	V509	C2 (FILTER 1 for torque current ref.)
V511	DINT	No	[ ]	1	V511	ph11 (FILTER 2 for torque current ref.)
V512	UDINT	No	[ ]	1	V512	ph12 (FILTER 2 for torque current ref.)
V513	UDINT	No	[ ]	1	V513	ph21 (FILTER 2 for torque current ref.)
V514	DINT	No	[ ]	1	V514	ph22 (FILTER 2 for torque current ref.)
V515	DINT	No	[ ]	1	V515	ga1 (FILTER 2 for torque current ref.)
V516	DINT	No	[ ]	1	V516	ga2 (FILTER 2 for torque current ref.)
V517	UDINT	No	[ ]	1	V517	r*r (FILTER 2 for torque current ref.)
V518	DINT	No	[ ]	1	V518	C1 (FILTER 2 for torque current ref.)
V519	DINT	No	[ ]	1	V519	C2 (FILTER 2 for torque current ref.)
V521	DINT	No	[ ]	1	V521	ph11 (FILTER 3 for torque current ref.)
V522	UDINT	No	[ ]	1	V522	ph12 (FILTER 3 for torque current ref.)

Param. N°	Type	Halt	[PU]	Resolution	Conversion rule from Vxx in physical unit to Vxx in internal unit	Description
V523	UDINT	No	[ ]	1	V523	ph21 (FILTER 3 for torque current ref.)
V524	DINT	No	[ ]	1	V524	ph22 (FILTER 3 for torque current ref.)
V525	DINT	No	[ ]	1	V525	ga1 (FILTER 3 for torque current ref.)
V526	DINT	No	[ ]	1	V526	ga2 (FILTER 3 for torque current ref.)
V527	UDINT	No	[ ]	1	V527	r*r (FILTER 3 for torque current ref.)
V528	DINT	No	[ ]	1	V528	C1 (FILTER 3 for torque current ref.)
V529	DINT	No	[ ]	1	V529	C2 (FILTER 3 for torque current ref.)
V531	DINT	No	[ ]	1	V531	ph11 (FILTER 4 for torque current ref.)
V532	UDINT	No	[ ]	1	V532	ph12 (FILTER 4 for torque current ref.)
V533	UDINT	No	[ ]	1	V533	ph21 (FILTER 4 for torque current ref.)
V534	DINT	No	[ ]	1	V534	ph22 (FILTER 4 for torque current ref.)
V535	DINT	No	[ ]	1	V535	ga1 (FILTER 4 for torque current ref.)
V536	DINT	No	[ ]	1	V536	ga2 (FILTER 4 for torque current ref.)
V537	UDINT	No	[ ]	1	V537	r*r (FILTER 4 for torque current ref.)
V538	DINT	No	[ ]	1	V538	C1 (FILTER 4 for torque current ref.)
V539	DINT	No	[ ]	1	V539	C2 (FILTER 4 for torque current ref.)
V541	DINT	No	[ ]	1	V541	ph11 (FILTER 1 for speed ref.)
V542	UDINT	No	[ ]	1	V542	ph12 (FILTER 1 for speed ref.)
V543	UDINT	No	[ ]	1	V543	ph21 (FILTER 1 for speed ref.)
V544	DINT	No	[ ]	1	V544	ph22 (FILTER 1 for speed ref.)
V545	DINT	No	[ ]	1	V545	ga1 (FILTER 1 for speed ref.)
V546	DINT	No	[ ]	1	V546	ga2 (FILTER 1 for speed ref.)
V547	UDINT	No	[ ]	1	V547	r*r (FILTER 1 for speed ref.)
V548	DINT	No	[ ]	1	V548	C1(FILTER 1 for speed ref.)
V549	DINT	No	[ ]	1	V549	C2(FILTER 1 for speed ref.)
V600	UDINT	No	%	1	V600*V108*18536.09716/ (Ipeak*100)	V/f: Maximal motor current (% of V108)
V601	UDINT	No	Hz	1	V601*13.1072*(V263/50)	V/f: Rated frequency
V602	REAL	No	Vrms	0.01	V602*8	V/f: Rated voltage
V603	REAL	No	V	0.01	V603*8	V/f: V0
V604	UDINT	No	mOhm	1	V604*(Ipeak*100)*65.536* 1.4/133772.76	V/f: Stator resistance @ 20 degrees C
V605	UDINT	No	%	1	V605*V108*18536.09716/ (Ipeak*100)	V/f: Intermittent overload (% of V108)
V606	UDINT	No	Hz	1	V606*13.1072*(V263/50)	V/f: Freq. start of overload reduction
V680	DWORD	No	[ ]	1	V680	Oscilloscope control
V681	DWORD	Yes	[ ]		V681	Oscilloscope configuration
V682	UDINT	Yes	[ ]	1	V682	Associate MP with oscilloscope channel 1
V683	UDINT	Yes	[ ]	1	V683	Associate MP with oscilloscope channel 2
V684	UDINT	Yes	[ ]	1	V684	Associate MP with oscilloscope trigger
V685	UDINT	Yes	us	50	V685/(2*V263)	Oscilloscope sample time
V686	UDINT	Yes	[ ]	1	V686	Samples number to be recorded
V687	UDINT	Yes	[ ]	1	V687	Samples number to be recorded post trigger
V688	DINT	Yes	[ ]	1	V688	Threshold level (I.U.) for MP value
V700	DWORD	Yes	[ ]		V700	Motor part number 1° characters group
V701	DWORD	Yes	[ ]		V701	Motor part number 2° characters group
V702	DWORD	Yes	[ ]		V702	Motor part number 3° characters group
V703	DWORD	Yes	[ ]		V703	Motor part number 4° characters group

Param. N°	Type	Halt	[PU]	Resolution	Conversion rule from Vxx in physical unit to Vxx in internal unit	Description
V704	DWORD	Yes	[ ]		V704	Motor part number 5° characters group
V705	DWORD	Yes	[ ]		V705	Motor part number 6° characters group
V706	DWORD	Yes	[ ]		V706	Motor part number 7° characters group
V707	DWORD	Yes	[ ]		V707	Motor part number 8° characters group
V708	DWORD	Yes	[ ]		V708	Motor part number 9° characters group
V709	DWORD	Yes	[ ]		V709	Motor part number 10° characters group
V712	DWORD	Yes	[ ]		V712	S1 sensor part number 1° characters group
V713	DWORD	Yes	[ ]		V713	S1 sensor part number 2° characters group
V714	DWORD	Yes	[ ]		V714	S1 sensor part number 3° characters group
V715	DWORD	Yes	[ ]		V715	S1 sensor part number 4° characters group
V716	DWORD	Yes	[ ]		V716	S1 sensor part number 5° characters group
V717	DWORD	Yes	[ ]		V717	S1 sensor part number 6° characters group
V718	DWORD	Yes	[ ]		V718	S1 sensor part number 7° characters group
V719	DWORD	Yes	[ ]		V719	S1 sensor part number 8° characters group
V720	DWORD	Yes	[ ]		V720	S1 sensor part number 9° characters group
V721	DWORD	Yes	[ ]		V721	S1 sensor part number 10° characters group
V724	DWORD	Yes	[ ]		V724	S2 sensor part number 1° characters group
V725	DWORD	Yes	[ ]		V725	S2 sensor part number 2° characters group
V726	DWORD	Yes	[ ]		V726	S2 sensor part number 3° characters group
V727	DWORD	Yes	[ ]		V727	S2 sensor part number 4° characters group
V728	DWORD	Yes	[ ]		V728	S2 sensor part number 5° characters group
V729	DWORD	Yes	[ ]		V729	S2 sensor part number 6° characters group
V730	DWORD	Yes	[ ]		V730	S2 sensor part number 7° characters group
V731	DWORD	Yes	[ ]		V731	S2 sensor part number 8° characters group
V732	DWORD	Yes	[ ]		V732	S2 sensor part number 9° characters group
V733	DWORD	Yes	[ ]		V733	S2 sensor part number 10° characters group
V900	DINT	Yes	[ ]	1	V900	AP07: DEMX instruction #0
V901	DINT	Yes	[ ]	1	V901	AP07: DEMX instruction #1
V902	DINT	Yes	[ ]	1	V902	AP07: DEMX instruction #2
V903	DINT	Yes	[ ]	1	V903	AP07: DEMX instruction #3
V904	DINT	Yes	[ ]	1	V904	AP07: DEMX instruction #4
V905	DINT	Yes	[ ]	1	V905	AP07: DEMX instruction #5
V906	DINT	Yes	[ ]	1	V906	AP07: DEMX instruction #6
V907	DINT	Yes	[ ]	1	V907	AP07: DEMX instruction #7
V908	DINT	Yes	[ ]	1	V908	AP07: DEMX instruction #8
V909	DINT	Yes	[ ]	1	V909	AP07: DEMX instruction #9
V910	DINT	Yes	[ ]	1	V910	AP07: DEMX instruction #10
V911	DINT	Yes	[ ]	1	V911	AP07: DEMX instruction #11
V912	DINT	Yes	[ ]	1	V912	AP07: DEMX instruction #12
V913	DINT	Yes	[ ]	1	V913	AP07: DEMX instruction #13
V914	DINT	Yes	[ ]	1	V914	AP07: DEMX instruction #14
V915	DINT	Yes	[ ]	1	V915	AP07: DEMX instruction #15
V920	DINT	No	[ ]	1	V920	AP07: DEMX parameter #0
V921	DINT	No	[ ]	1	V921	AP07: DEMX parameter #1
V922	DINT	No	[ ]	1	V922	AP07: DEMX parameter #2
V923	DINT	No	[ ]	1	V923	AP07: DEMX parameter #3

Param. N°	Type	Halt	[PU]	Resolution	Conversion rule from Vxx in physical unit to Vxx in internal unit	Description
V924	DINT	No	[ ]	1	V924	AP07: DEMX parameter #4
V925	DINT	No	[ ]	1	V925	AP07: DEMX parameter #5
V926	DINT	No	[ ]	1	V926	AP07: DEMX parameter #6
V927	DINT	No	[ ]	1	V927	AP07: DEMX parameter #7
V928	DINT	No	[ ]	1	V928	AP07: DEMX parameter #8
V929	DINT	No	[ ]	1	V929	AP07: DEMX parameter #9
V930	DINT	No	[ ]	1	V930	AP07: DEMX parameter #10
V931	DINT	No	[ ]	1	V931	AP07: DEMX parameter #11
V932	DINT	No	[ ]	1	V932	AP07: DEMX parameter #12
V933	DINT	No	[ ]	1	V933	AP07: DEMX parameter #13
V934	DINT	No	[ ]	1	V934	AP07: DEMX parameter #14
V935	DINT	No	[ ]	1	V935	AP07: DEMX parameter #15
V1045	REAL	No	mm/min	0.01	V1045*163.840*(V263/50)	Speed dynamic check [1]
V1046	REAL	No	rpm	0.01	V1046*1342.18*(V263/50)	Speed dynamic check [1]
V1050	REAL	No	Arms	0.01	V1050*1853609.716/(Ipeak*100)	Torque current limit [1]
V1051	REAL	No	mArms/ rpm	0.01	V1051*90509.668*(50/V263)/ (Ipeak*100)	Proportional speed loop gain [1]
V1052	REAL	No	ms	0.01	32767*(2*V263)/(V1052*1000)	Integral speed loop gain [1]
V1057	REAL	No	Arms	0.01	V1057*1853609.716/ (Ipeak*100)	Torque current limit h.r. [1]
V1058	REAL	No	mArms/ rpm	0.01	V1058*90509.668*(50/V263)/ (Ipeak*100)	Proportional speed loop gain h.r. [1]
V1059	REAL	No	ms	0.01	32767*(2*V263)/(V1059*1000)	Integral speed loop gain h.r. [1]
V1060	REAL	No	Arms/m/ min	0.01	V1060*741455.2*(50/V263)/ (Ipeak*100)	Proportional speed loop gain [1]
V1091	UDINT	No	rpm/s	1	V1091*(4*V263)*1.34218* (V263/50)/1000	Acceleration/deceleration ramp [1]
V1092	UDINT	No	rpm/s	1	V1092*(4*V263)*1.34218* (V263/50)/1000	Acceleration/deceleration ramp h.r.[1]
V1093	UDINT	No	mm/s^2	1	V1093*60*(4*V263)*0.163840* (V263/50)/1000	Acceleration/deceleration ramp [1]
V1094	UDINT	No	%	1	V1094*163840*V1095/100	Speed reference Limit (% V1095) [1]
V1095	UDINT	No	m/min	1	V1095*163840*(V263/50)	Maximum motor speed [1]
V1096	UDINT	No	m/min	1	V1096*163840*(V263/50)	Overspeed alarm threshold [1]
V1097	UDINT	No	%	1	V1097*1342.18*V1098/100	Speed reference Limit (% V1098) [1]
V1098	UDINT	No	rpm	1	V1098*1342.18*(V263/50)	Maximum motor speed [1]
V1099	UDINT	No	rpm	1	V1099*1342.18*(V263/50)	Overspeed alarm threshold [1]
V1301	UDINT	No	mm/min/ mm	1	V1301	Proportional position gain [1]
V1305	UDINT	Yes	[ ]	1	V1305	Measure module for rot. axis/spindle abs[1]
V1306	DINT	Yes	[ ]	1	V1306	S1 sensor MULTI coefficient [1]
V1307	UDINT	Yes	[ ]	1	V1307	S1 sensor DIVI coefficient [1]
V1310	DINT	Yes	[ ]	1	V1310	S1 sensor offset in CNC Phys. Unit [1]
V1311	DINT	Yes	[ ]	1	V1311	S2 sensor MULTI coefficient [1]
V1312	UDINT	Yes	[ ]	1	V1312	S2 sensor DIVI coefficient [1]
V1313	DINT	Yes	[ ]	1	V1313	S2 sensor offset in CNC Phys. Unit [1]
V1353	REAL	Yes	[ ]	0.01	V1353*100	Period of position sensor/motor turn [1]
V1359	UDINT	No	[ ]	1	(65536*3*4*1024/V1353)	k_velrot121
V1362	UDINT	No	[ ]	1	V1301*1.31072*327.68*4*(2*V263)/ V360	kpposrot11

Param. N°	Type	Halt	[PU]	Resolution	Conversion rule from Vxx in physical unit to Vxx in internal unit	Description
V1372	UDINT	No	[ ]	1	V1301*1.31072*327.68*4*(2*V263)/ V1353	kpposrot121
V1376	UDINT	No	[ ]	1	V1301*1.31072*327.68*4*(2*V263)/ V370	kpposrot21
V1383	UDINT	No	[ ]	1	V1301*5.24288*V380*4* (2*V263/100)	kpposlin11
V1393	UDINT	No	[ ]	1	V1301*5.24288*V390*4* (2*V263/100)	kpposlin21

A.3.2 Dependencies table

This table indicates the parameters for which a change on another has an effect.

Param. modified	Parameters impacted by this change
V95	V94
V98	V97
V100	V125, V126, V180, V491, V602
V105	V114
V106	V118
V108	V109, V123, V125,V126, V128, V171, V242, V244, V600, V605
V115	V1097, V1098
V116	V1097, V1098
V123	V123, V241, V244
V124	V122, V243, V247
V128	V244
V129	V247
V217	V218, V219
V218	V219
V245	V246
V257	V104, V120, V414, V415, V417, V454, V455, V457, V496, V497
V259	V120, V382, V386, V392, V396, V497
V263	V41, V42, V45, V46, V51, V52, V58, V59, V60, V91, V92, V93, V95, V96, V98, V99, V104, V105, V106, V114, V116, V117, V118, V120, V144, V154, V165, V166, V172, V173, V175, V176, V177, V218, V219, V245, V246, V263, V271, V342, V343, V362, V372, V376, V383, V393, V411, V417, V426, V451, V457, V466, V470, V471, V479, V489, V496, V497, V498, V499, V601, V606, V685, V1045, V1046, V1051, V1052, V1058, V1059, V1060, V1091, V1092, V1093, V1095, V1096, V1098, V1099, V1362, V1372, V1376, V1383, V1393
V271	V98, V99, V103, V107, V112, V119, V271, V400, V401, V403, V404, V405, V414, V415, V435, V454, V455, V475, V1098
V301	V362, V372, V376, V383, V393
V353	V359, V372
V360	V362, V363, V364, V1362
V361	V364
V370	V373, V374, V376, V1376
V371	V374
V380	V382, V383, V384, V385, V1383
V381	V385, V386
V390	V392, V393, V394, V395, V1393
V391	V395, V396
V400	V472, V473
V401	V472, V473
V402	V448, V470, V471
V403	V400, V401, V402, V404, V405, V448
V406	V400, V401
V407	V114, V117, V118, V402, V405, V407, V448, V471, V473
V408	V105, V106, V114, V118, V402, V408, V448, V470, V471, V489,
V412	V413, V425, V427
V427	V425
V452	V453, V465, V467,
V467	V465

Param. modified	Parameters impacted by this change
V491	V492
V600	V605
V601	V606
V686	V687
V1095	V1094,
V1098	V1097
V1301	V1362, V1372, V1376, V1383, V1393
V1353	V1359, V1372

A.3.3 NUMDrive X parameters format and limits

Parameter N°	Type	Resolution	Min value [PU]	Max value [PU]
V25	DWORD		1	80000000h
V26	DWORD		1	80000000h
V27	DWORD		1	80000000h
V28	DWORD		1	80000000h
V29	DWORD		1	80000000h
V30	DWORD	1	0	FFFFFFFFh
V32	DWORD	1	0	7FFFFFFFh
V33	DWORD	1	0	3FFFFFFFh
V34	DWORD	1	0	1FFFFFh
V35	DWORD	1	0	7FFFh
V36	UDINT	1	1	1023
V37	REAL	0.01	0.2	2.0
V40	REAL	0.01	0.0	50.0
V41	REAL	0.01	1.0	3000.0
V42	REAL	0.01	0.1	5000.0
V43	REAL	0.01	1.0	120.0
V45	REAL	0.01	0.0	20000.0
V46	REAL	0.01	0.0	2000.0
V50	REAL	0.01	0.0	Ipeak/141.42
V51	REAL	0.01	0.0	16777200*(Ipeak*100)/(90509.668*(50/V263))
V52	REAL	0.01	2*(2*V263)/1000	32767*(2*V263)/(0.99*1000)
V57	REAL	0.01	0.0	(Ipeak*100)/141.42
V58	REAL	0.01	0.0	16777200*(Ipeak*100)/(90509.668*(50/V263))
V59	REAL	0.01	2*(2*V263)/1000	32767*(2*V263)/(0.99*1000)
V60	REAL	0.01	0.0	16777200*(Ipeak*100)/(741455.2*(50/V263))
V80	REAL	0.01	0.0	1.0
V81	REAL	0.01	1.0	2.0
V91	UDINT	1	4	1000000
V92	UDINT	1	4	1000000
V93	UDINT	1	5	100000
V94	UDINT	1	0	200
V95	UDINT	1	0	1000
V96	UDINT	1	0	1000
V97	UDINT	1	0	200
V98	UDINT	1	0	140000*50*(1+0.25*((V271/V263)-1))/(V257*V271)
V99	UDINT	1	0	140000*50*(1+0.25*((V271/V263)-1))/(V257*V271)
V100	REAL	0.01	0.0	(Ipeak*100)/141.42
V101	UDINT	1	0	1000
V102	REAL	0.01	0.0	926790*4/(Ipeak*100)
V103	REAL	0.01	2*V271/1000	32767*V271/(1000*0.99)
V104	REAL	0.001	0.0	32760*V257/(16.237976*2*(50/V263))
V105	REAL	0.01	0.0	(32760/((Ipeak*100)*0.01503011016*(50/V263)))-V408
V106	REAL	0.01	0.0	(32760/((Ipeak*100)*0.01503011016*(50/V263)))-V408
V107	UDINT	1	1	3000
V108	REAL	0.01	(Ipeak*100)/4715.0	(Ipeak*100)*0.9/141.42
V109	REAL	0.01	-(V108)	V108
V110	REAL	0.01	0.0	(Ipeak*100)/141.42
V111	REAL	0.01	0.0	926790*4/(Ipeak*100)
V112	REAL	0.01	2*V271/1000	32767*V271/(1000*0.99)

Parameter N°	Type	Resolution	Min value [PU]	Max value [PU]
V114	REAL	0.01	0.0	32767.0
V115	REAL	0.01	(Ipeak*100)/4715.0	(Ipeak*100)*0.9/141.42
V116	REAL	0.01	0.0	32760/((Ipeak*100)*0.01503011016*(50/V263))
V117	REAL	0.01	0.0	32767.0
V118	REAL	0.01	0.0	32767.0
V119	REAL	0.01	0.0	3.00
V120	REAL	0.01	0.0	32760*V263*V257/(V259*54.12659)
V121	REAL	0.01	0.4	1.0
V122	REAL	0.01	0.4	1.0
V123	REAL	0.01	1.12*V108	5*V108
V124	REAL	0.01	1.12*V115	5*V115
V125	UDINT	1	0	(100.0*V100)/V108
V126	UDINT	1	0	V100*127/V108
V128	UDINT	1	50	100
V129	UDINT	1	50	100
V130	UDINT	1	1	500
V131	UDINT	1	1	500
V132	UDINT	1	1	500
V133	UDINT	1	1	500
V135	DWORD	1	0	7h
V136	UDINT	1	25	1500
V137	UDINT	1	25	1500
V140	UDINT	1	1	500
V141	DINT	1	-2147483648	2147483647
V142	DINT	1	-2147483648	2147483647
V144	UDINT	1	5	1500
V145	DWORD	1	0	FFFFFFFFh
V150	UDINT	1	1	500
V151	DINT	1	-2147483648	2147483647
V152	DINT	1	-2147483648	2147483647
V154	UDINT	1	5	1500
V155	DWORD	1	0	FFFFFFFFh
V160	DWORD	1	0	FFFh
V162	UDINT	1	0	32767
V163	UDINT	1	1	32767
V164	DWORD	1	0	3h
V165	UDINT	1	0	4000
V166	REAL	0.01	0.0	32760*1000/(5.656941*(Ipeak*100)*(V263/50))
V169	DWORD	1	0	1h
V170	DWORD	1	0	7h
V171	REAL	0.01	-(V108)	V108
V172	REAL	0.01	0.0	32760*21.5792/((Ipeak*100)*(V263/50))
V173	REAL	0.01	2*(4*V263)/1000	32767*(4*V263)/(1000*0.99)
V175	UDINT	1	0	400
V176	UDINT	1	0	32760*(4*V263)/1000
V177	REAL	0.01	0.1	10000.0
V180	UDINT	1	2	4
V181	UDINT	1	0	3
V182	DWORD	1	0	7h
V185	UDINT	1	2	4
V186	UDINT	1	0	3

Parameter N°	Type	Resolution	Min value [PU]	Max value [PU]
V187	DWORD	1	0	7h
V190	UDINT	1	1	10000
V191	UDINT	1	0	1000
V200	DWORD	1	0	1h
V201	DWORD	1	0	3h
V202	UDINT	1	1	500
V217	REAL	0.01	0.1	6.0
V218	REAL	0.01	2.0	150.0
V219	UDINT	1	0	32767
V220	DWORD	1	0	1Fh
V221	DWORD	1	0	3Fh
V238	DWORD		1	80000000h
V239	DWORD		1	80000000h
V240	REAL	0.01	1.0	5.0
V241	UDINT	1	5	1000
V242	REAL	0.01	0.0	32767.0
V243	REAL	0.01	0.0	32767.0
V244	REAL	0.01	0.0	32767.0
V245	REAL	0.01	1.25	32760/(13.1072*(V263/50))
V246	REAL	0.01	0.25	V245-1.00
V247	REAL	0.01	0.0	32767.0
V248	REAL	0.01	1.0	5.0
V249	UDINT	1	5	1000
V250	DWORD	1	0	FFFFh
V251	UDINT	1	1	1
V252	STRING		I	L
V253	DWORD	1	0	Fh
V255	STRING		0	2
V256	STRING		0	P
V257	UDINT	1	1	32767
V259	REAL	0.01	0.0	32760.0
V263	UDINT	25	25	50
V268	UDINT	1	0	359999
V269	UDINT	1	0	359999
V270	UDINT	1	0	80000
V271	UDINT	25	V263	2*V263
V272	DWORD	1	0	7h
V273	DWORD	1	0	FFh
V301	UDINT	1	0	50000
V305	UDINT	1	0	2147483647
V306	DINT	1	-67108863	67108863
V307	UDINT	1	1	65535
V310	DINT	1	-2147483648	2147483647
V311	DINT	1	-67108863	67108863
V312	UDINT	1	1	65535
V313	DINT	1	-2147483648	2147483647
V314	DWORD	1	0	Fh
V315	DWORD	1	0	Fh
V317	REAL	0.01	2.5	11.5
V318	REAL	0.01	2.5	11.5
V320	DWORD		1	80000000h

Parameter N°	Type	Resolution	Min value [PU]	Max value [PU]
V321	DWORD		1	80000000h
V322	DWORD		1	80000000h
V323	DWORD		1	80000000h
V324	DWORD	1	0	1FFh
V325	DWORD	1	0	1FFh
V326	DWORD	1	0	7h
V327	DWORD	1	0	7h
V328	DWORD	1	0	3Fh
V329	DWORD	1	0	3Fh
V330	DWORD		1	80000000h
V331	DWORD		1	80000000h
V340	UDINT	1	0	6
V341	UDINT	1	0	6
V342	REAL	0.01	50.0	3000.00*(50/V263)
V343	REAL	0.01	50.0	3000.00*(50/V263)
V344	UDINT	1	0	4294967295
V345	UDINT	1	0	4294967295
V350	DWORD	1	0	1h
V353	REAL	0.01	1.0	100000.0
V359	UDINT	1	0	32767
V360	UDINT	1	1	360000
V361	UDINT	1	1	2147483647
V362	UDINT	1	0	32767
V363	UDINT	1	0	32767
V364	UDINT	1	0	32767
V365	UDINT	1	1	65536
V368	UDINT	1	1	32767
V369	UDINT	1	1	32
V370	UDINT	1	1	360000
V371	UDINT	1	1	2147483647
V372	UDINT	1	0	32767
V373	UDINT	1	0	32767
V374	UDINT	1	0	32767
V375	UDINT	1	1	65536
V376	UDINT	1	0	32767
V378	UDINT	1	1	32767
V379	UDINT	1	1	32
V380	UDINT	1	1	8000
V381	UDINT	1	1	32767
V382	UDINT	1	0	4294967295
V383	UDINT	1	0	32767
V384	UDINT	1	0	32767
V385	UDINT	1	0	32767
V386	UDINT	1	0	32767
V389	UDINT	1	0	48
V390	UDINT	1	1	8000
V391	UDINT	1	1	32767
V392	UDINT	1	0	4294967295
V393	UDINT	1	0	32767
V394	UDINT	1	0	4294967295
V395	UDINT	1	0	32767

Parameter N°	Type	Resolution	Min value [PU]	Max value [PU]
V396	UDINT	1	0	32767
V399	UDINT	1	0	48
V400	UDINT	1	0	32760*50*V403/(3.2768*V406*V271)
V401	UDINT	1	0	32760*50*V403/(3.2768*V406*V271)
V402	UDINT	1	0	(32760*V403/65.536)-V408*1000*V407
V403	REAL	0.01	133774.75*3276.8*(V271/50)/ ((Ipeak*100)*32760.00)	32760.0
V404	UDINT	1	0	32767
V405	UDINT	1	0	32767
V406	UDINT	1	1	16
V407	UDINT	1	3	4
V408	REAL	0.01	0.0	20.0
V409	UDINT	1	0	700/1.4142135
V410	REAL	0.01	0.0	127.9
V411	REAL	0.01	2*(2*V263)/1000	(2*V263)*32.767/0.99
V412	UDINT	1	2	100
V413	UDINT	1	0	32767
V414	UDINT	1	0	32760*100/(0.3431457*V257*V271)
V415	UDINT	1	0	32760*100/(0.3431457*V257*V271)
V417	UDINT	1	0	32760/(V257*0.21845333*(V263/50))
V425	UDINT	1	1	20
V426	REAL	0.01	0.0	32760*(Ipeak*100)/(7592.3856*100*(V263/50))
V427	REAL	0.01	0.0	13107*(Ipeak*100)/(V412*13107.2*1.4142135)
V428	REAL	0.01	0.0	(Ipeak*100)/141.42
V434	REAL	0.01	0.0	926790*4/(Ipeak*100)
V435	REAL	0.01	2*V271/1000	32767*V271/(1000*0.99)
V448	UDINT	1	0	32767
V450	REAL	0.01	0.0	127.9
V451	REAL	0.01	2*(2*V263)/1000	(2*V263)*32.767/0.99
V452	UDINT	1	0	100
V453	UDINT	1	0	32767
V454	UDINT	1	0	32760*100/(0.3431457*V257*V271)
V455	UDINT	1	0	32760*100/(0.3431457*V257*V271)
V457	UDINT	1	0	32760/(V257*0.21845333*(V263/50))
V465	UDINT	1	1	20
V466	REAL	0.01	0.0	32760*(Ipeak*100)/(7592.3856*100*(V263/50))
V467	REAL	0.01	0.0	13107*(Ipeak*100)/(V452*13107.2*1.4142135)
V468	REAL	0.01	0.0	(Ipeak*100)/141.42
V470	REAL	0.01	0.0	32767.0
V471	REAL	0.01	0.0	32767.0
V472	UDINT	1	0	32767
V473	UDINT	1	0	32767
V474	REAL	0.01	0.0	926790*4/(Ipeak*100)
V475	REAL	0.01	2*V271/1000	32767*V271/(1000*0.99)
V479	REAL	0.01	1.0	2000.0
V480	REAL	0.01	0.0	100.0
V481	DWORD	1	0	7h
V488	UDINT	1	0	32760*((234.5+20.0)*2*133772.76)/ ((Ipeak*100)*65.536*(234.5+120.0))
V489	REAL	0.01	0.0	(32760/((Ipeak*100)*0.01503011016))-V408
V491	REAL	0.01	-(Ipeak*100)/141.42	0.0
V492	REAL	0.01	V491	0.0

Parameter N°	Type	Resolution	Min value [PU]	Max value [PU]
V496	UDINT	1	0	32760/(0.21845333*V257*(V263/50))
V497	REAL	0.01	0.0	32760/(109.22667*V257*(V263/50)/V259)
V498	REAL	0.01	0.0	32760*(Ipeak*100)/(7592.3856*100*(V263/50))
V499	REAL	0.01	0.0	32760*(Ipeak*100)/(7592.3856*100*(V263/50))
V500	DWORD	1	0	FFh
V501	DINT	1	-32767	32767
V502	UDINT	1	0	65535
V503	UDINT	1	0	65535
V504	DINT	1	-32767	32767
V505	DINT	1	-32767	32767
V506	DINT	1	-32767	32767
V507	UDINT	1	0	32767
V508	DINT	1	-32767	32767
V509	DINT	1	-32767	32767
V511	DINT	1	-32767	32767
V512	UDINT	1	0	65535
V513	UDINT	1	0	65535
V514	DINT	1	-32767	32767
V515	DINT	1	-32767	32767
V516	DINT	1	-32767	32767
V517	UDINT	1	0	32767
V518	DINT	1	-32767	32767
V519	DINT	1	-32767	32767
V521	DINT	1	-32767	32767
V522	UDINT	1	0	65535
V523	UDINT	1	0	65535
V524	DINT	1	-32767	32767
V525	DINT	1	-32767	32767
V526	DINT	1	-32767	32767
V527	UDINT	1	0	32767
V528	DINT	1	-32767	32767
V529	DINT	1	-32767	32767
V531	DINT	1	-32767	32767
V532	UDINT	1	0	65535
V533	UDINT	1	0	65535
V534	DINT	1	-32767	32767
V535	DINT	1	-32767	32767
V536	DINT	1	-32767	32767
V537	UDINT	1	0	32767
V538	DINT	1	-32767	32767
V539	DINT	1	-32767	32767
V541	DINT	1	-32767	32767
V542	UDINT	1	0	65535
V543	UDINT	1	0	65535
V544	DINT	1	-32767	32767
V545	DINT	1	-32767	32767
V546	DINT	1	-32767	32767
V547	UDINT	1	0	32767
V548	DINT	1	-32767	32767
V549	DINT	1	-32767	32767
V600	UDINT	1	0	(Ipeak*100)/(V108*1.4142)

Parameter N°	Type	Resolution	Min value [PU]	Max value [PU]
V601	UDINT	1	10	1500
V602	REAL	0.01	100.0	500.0
V603	REAL	0.01	0.0	50.0
V604	UDINT	1	0	32760*133772.76/((Ipeak*100)*65.536*1.4)
V605	UDINT	1	0	V600*91/100
V606	UDINT	1	V601/2	1500
V680	DWORD	1	0	3h
V681	DWORD		1	80000000h
V682	UDINT	1	1	500
V683	UDINT	1	1	500
V684	UDINT	1	1	500
V685	UDINT	50	2*V263	6553500
V686	UDINT	1	2	2048
V687	UDINT	1	1	V686-1
V688	DINT	1	-2147483648	2147483647
V700	DWORD		0	7E7E7E7Eh
V701	DWORD		0	7E7E7E7Eh
V702	DWORD		0	7E7E7E7Eh
V703	DWORD		0	7E7E7E7Eh
V704	DWORD		0	7E7E7E7Eh
V705	DWORD		0	7E7E7E7Eh
V706	DWORD		0	7E7E7E7Eh
V707	DWORD		0	7E7E7E7Eh
V708	DWORD		0	7E7E7E7Eh
V709	DWORD		0	7E7E7E7Eh
V712	DWORD		0	7E7E7E7Eh
V713	DWORD		0	7E7E7E7Eh
V714	DWORD		0	7E7E7E7Eh
V715	DWORD		0	7E7E7E7Eh
V716	DWORD		0	7E7E7E7Eh
V717	DWORD		0	7E7E7E7Eh
V718	DWORD		0	7E7E7E7Eh
V719	DWORD		0	7E7E7E7Eh
V720	DWORD		0	7E7E7E7Eh
V721	DWORD		0	7E7E7E7Eh
V724	DWORD		0	7E7E7E7Eh
V725	DWORD		0	7E7E7E7Eh
V726	DWORD		0	7E7E7E7Eh
V727	DWORD		0	7E7E7E7Eh
V728	DWORD		0	7E7E7E7Eh
V729	DWORD		0	7E7E7E7Eh
V730	DWORD		0	7E7E7E7Eh
V731	DWORD		0	7E7E7E7Eh
V732	DWORD		0	7E7E7E7Eh
V733	DWORD		0	7E7E7E7Eh
V900	DINT	1	-2147483648	2147483647
V901	DINT	1	-2147483648	2147483647
V902	DINT	1	-2147483648	2147483647
V903	DINT	1	-2147483648	2147483647
V904	DINT	1	-2147483648	2147483647
V905	DINT	1	-2147483648	2147483647

Parameter N°	Type	Resolution	Min value [PU]	Max value [PU]
V906	DINT	1	-2147483648	2147483647
V907	DINT	1	-2147483648	2147483647
V908	DINT	1	-2147483648	2147483647
V909	DINT	1	-2147483648	2147483647
V910	DINT	1	-2147483648	2147483647
V911	DINT	1	-2147483648	2147483647
V912	DINT	1	-2147483648	2147483647
V913	DINT	1	-2147483648	2147483647
V914	DINT	1	-2147483648	2147483647
V915	DINT	1	-2147483648	2147483647
V920	DINT	1	-2147483648	2147483647
V921	DINT	1	-2147483648	2147483647
V922	DINT	1	-2147483648	2147483647
V923	DINT	1	-2147483648	2147483647
V924	DINT	1	-2147483648	2147483647
V925	DINT	1	-2147483648	2147483647
V926	DINT	1	-2147483648	2147483647
V927	DINT	1	-2147483648	2147483647
V928	DINT	1	-2147483648	2147483647
V929	DINT	1	-2147483648	2147483647
V930	DINT	1	-2147483648	2147483647
V931	DINT	1	-2147483648	2147483647
V932	DINT	1	-2147483648	2147483647
V933	DINT	1	-2147483648	2147483647
V934	DINT	1	-2147483648	2147483647
V935	DINT	1	-2147483648	2147483647
V1045	REAL	0.01	0.0	20000.0
V1046	REAL	0.01	0.0	2000.0
V1050	REAL	0.01	0.0	Ipeak/141.42
V1051	REAL	0.01	0.0	16777200*(Ipeak*100)/(90509.668*(50/V263))
V1052	REAL	0.01	2*(2*V263)/1000	32767*(2*V263)/(0.99*1000)
V1057	REAL	0.01	0.0	(Ipeak*100)/141.42
V1058	REAL	0.01	0.0	16777200*(Ipeak*100)/(90509.668*(50/V263))
V1059	REAL	0.01	2*(2*V263)/1000	32767*(2*V263)/(0.99*1000)
V1060	REAL	0.01	0.0	16777200*(Ipeak*100)/(741455.2*(50/V263))
V1091	UDINT	1	4	1000000
V1092	UDINT	1	4	1000000
V1093	UDINT	1	5	100000
V1094	UDINT	1	0	200
V1095	UDINT	1	0	1000
V1096	UDINT	1	0	1000
V1097	UDINT	1	0	200
V1098	UDINT	1	0	140000*50*(1+0.25*((V271/V263)-1))/(V257*V271)
V1099	UDINT	1	0	140000*50*(1+0.25*((V271/V263)-1))/(V257*V271)
V1301	UDINT	1	0	50000
V1305	UDINT	1	0	2147483647
V1306	DINT	1	-67108863	67108863
V1307	UDINT	1	1	65535
V1310	DINT	1	-2147483648	2147483647
V1311	DINT	1	-67108863	67108863
V1312	UDINT	1	1	65535

Parameter N°	Type	Resolution	Min value [PU]	Max value [PU]
V1313	DINT	1	-2147483648	2147483647
V1353	REAL	0.01	1.0	100000.0
V1359	UDINT	1	0	32767
V1362	UDINT	1	0	32767
V1372	UDINT	1	0	32767
V1376	UDINT	1	0	32767
V1383	UDINT	1	0	32767
V1393	UDINT	1	0	32767

A.4 NCK parameters

A.4.1 NCK Parameters for Flexium+

The following parameters can be updated on the fly :

P11	Measurement conversion coefficient for analog axes
P13	Handwheel measurement conversion coefficien
P15	Homing direction
P16	Machine home position
P17	Axes travel limits
P18	Backlash error compensation
P19	High Speed Cutting: Feed-forward acceleration time constant - Dry friction correction time constant
P20	Dry friction correction amplitude
P23	Maximum following error
P24.. P25	Axes synchronization and duplication
P30	Maximum axes traverse rate
P32	Maximum permissible acceleration
P33	Approach speed
P40	Number of sensor spindle pulses by revolution
P41	Spindle reference reversal (according to the range selected)
P42	Spindle origin
P43	Spindle indexing speed and spindle speed limits
P44	Spindle indexing in-position window
P45	Spindle indexing gain
P46	Speed range and max speed for spindles @0 ... @7
P47	Speed range and max speed for spindles @8 ... @15
P48	Speed range and max speed for spindles @16 ... @23
P49	Speed range and max speed for spindles @24 ... @31
P55 N0 ÷ N7	Speed anticipation coefficient - High Speed Cutting: number of filter terms
P56	Channel specific time constant
P57	Dynamic lag control
P66	Time constant for axis
P70 N1, N5	Analog ports configuration
P73	Servo-system loop coefficient for analog axes
P85	Drives coupling assignment
P116	Electronic Gear Box function (EGB)
P120 N2, N5	Machine configuration

A.4.2 NCK Parameters for Flexium

The following parameters can be updated on the fly :

P3	Servo-controlled and interpolated axes
P6	Spindle declaration and assignment to channels
P10	Axis measurement direction
P11	Axis measurement conversion coefficient
P12	Handwheel measurement direction
P13	Handwheel measurement conversion coefficient
P14	Handwheel declaration
P15	Homing Direction
P16	Home position in machine dimensions
P17	Axis travel limit
P18	Backlash error compensation
P19	High Speed Cutting machining
P20	Direction of axis speed reference
P21	Servo-System loop gain coefficient for analog axes
P23	Maximum following error
P24	Synchronised axes characteristics
P25	Enable monitoring of analog axes signals
P30	Maximum axes traverse rates
P32	Maximum permissible acceleration
P33	Approach speed
P40	Spindle measurement conversion
P41	Spindle reference reversal
P42	Spindle origins
P43	Spindle indexing speed and speed limit by PLC
P44	Spindel indexing inposition window
P45	Spindle indexing gain
P46	Spindle 1 speed range
P47	Spindle 2 speed range
P48	Spindle 3 speed range
P49	Spindle 4 speed range
P55	Speed anticipation coefficient - Speed reference filter
P56	Channel specific time constant
P57	Dynamic lag control

P70	Axis/Spindle switching. Sensorless DISC NT spindles and Analog port
P77	Filtering SSI encoder faults
P85	Tandem Function: master/slave coupled axes
P86	Tandem Function: Defines master axes with torque synchronisation

Page deliberately empty.

Page deliberately empty.

Published by NUM Spa

Via F. Somma 62
20012 Cuggiono (MI) (Italy)
Tel. : +39 02 979 691
website: www.num.com

Printed November 2016

CNC HighEnd Applications