CANBus

Interface

TRANSLATION OF THE GERMAN ORIGINAL MANUAL

Phytron GmbH

Industriestraße 12

82194 Gröbenzell, Germany

Tel.: +49(0)8142/503-0

Fax: +49(0)8142/503-190

In this manual you will find the feature descriptions and programming a controller for positioning of a stepper motor.

This manual is supplementary to the "*phy***LOGIC**™-Command References" for the *phy***MOTION***TM* controller.

In the manual *phy***MOTION**TM *Modular Multi-axis Controller for Stepper Motors* (http://www.phytron.eu/phyMOTION) are the descriptions of the features and specifications for the *phy***MOTION**TM stepper motor controller.

Every possible care has been taken to ensure the accuracy of this technical manual. All information contained in this manual is correct to the best of our knowledge and belief but cannot be guaranteed. Furthermore we reserve the right to make improvements and enhancements to the manual and / or the devices described herein without prior notification.

We appreciate suggestions and criticisms for further improvement.

Email address: doku@phytron.de

Questions about the use of the product described in the manual that you cannot find answered here, please contact your representative of phytron (http://www.phytron.eu/) in your local agencies.

1 Legal information

This manual:

Read this manual very carefully before mounting, installing and operating the device and if necessary further manuals related to this manual.

- Please pay special attention to instructions that are marked as follows:

	**DANGER – Serious injury!**	*Indicates a high risk of serious injury or death!*
	**DANGER – Serious injury from electric shock!**	*Indicates a high risk of serious injury or death from electric shock!*
	**WARNING – Serious injury possible!**	*Indicates a possible risk of serious injury or death!*
	**WARNING – Serious injury from electric shock!**	*Indicates a possible risk of serious injury or death from electric shock!*
	**CAUTION – Possible injury!**	*Indicates a possible risk of personal injury.*
	**CAUTION – Possible damage!**	*Indicates a possible risk of damage to equipment.*
	**CAUTION – Possible damage due to ESD!**	*Refers to a possible risk of equipment damage from electrostatic discharge.*
	**"Any heading"**	*Refers to an important paragraph in the manual.*

Safety Instructions

CAUTION – Possible damage!

Malfunctions are possible while programming the instruction codes – e.g. sudden running of a connected motor, braking etc.

- Please test the program flow step by step.

CAUTION – Possible damage!

For each application, the functional reliability of software products by external factors such as voltage differences or hardware failure, etc. is affected.

- To prevent damage due to system error, the user should take appropriate safety measures. These include back-up and shut-down mechanisms.

CAUTION – Possible damage!

Each end user system is customised and differs from the testing platform. Therefore the user or application designer is responsible for verifying and validating the suitability of the application.

- The suitability of the device's use must be tested and validated.

CAUTION – Possible damage!

Some modules are set to a default value on delivery. So, e.g., the motor current must be set to the corresponding value (see the motor data from the motor manufacturer). Connected components like motors can be damaged by incorrectly set values.

- Please check before starting, if the parameters are correct.

2 Contents

3 Introduction

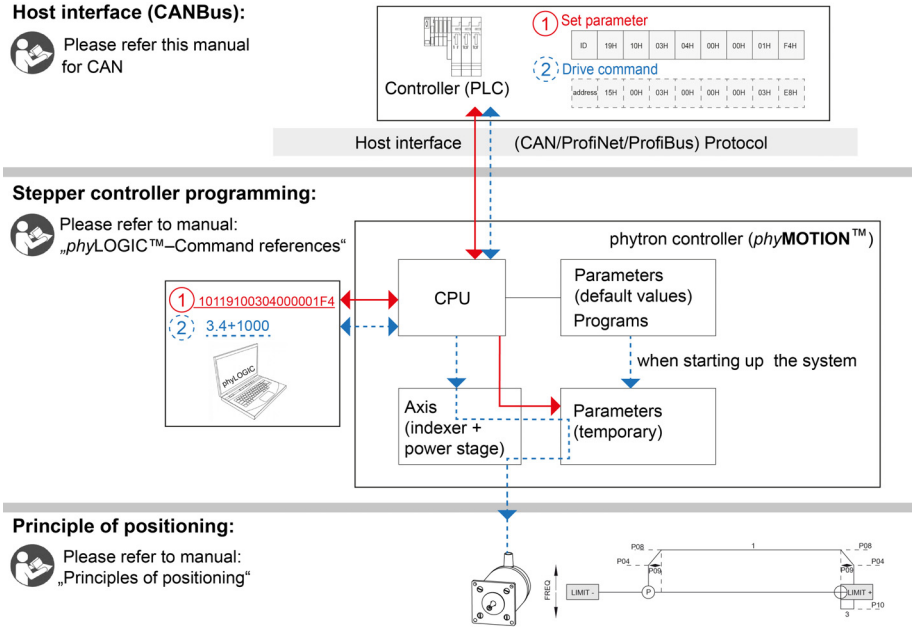
*phy***LOGIC**™ is the programming language to communicate with phytron programmable logic controllers like the MCC-Series or our *phy***MOTION**™.

*phy***LOGIC**™ commands can easily be sent to the controller with phytron's programming software (*phy***LOGIC**™ Toolbox) via USB, embedded into other protocols like Ethernet or into interface protocols like ProfiBus / ProfiNet.

You can parameterise your commands (e.g. a driving command) per axis either just the first time you set up your system, or adjust the parameters temporarily before sending a driving command.

Example: For "relative run" you can set: step resolution (P45), run current (P41), run frequency (P14), start stop frequency (P04), ramp (P15), recovery time (P16), boost (P17), boost current (P42), current hold time (P43), etc.

Use this illustration to find the adequate manual for your programming task:



Each of our programmable controllers comes along with pre set parameters (default values), which are automatically loaded into the temporary memory of each axis while starting the device. These parameters can be changed during your program is executed to optimise your motion tasks at any time.

If you want your controller to wake up with a new set of parameters, you have to explicitly store them in the non volatile storage of the main CPU unit by using a certain command.

4 General

This manual describes the firmware program (version 1.0.1) for the CANS01 interface module as implemented for the MCM01 module of the *phy***MOTION**™. The CANS01 module uses an asynchronous transfer protocol and operates as slave on the CAN bus, it will only send data on request by the host.

4.1 Specification

- CAN bus operation at 250 kbps (default):
  configurable in 250/500/1000 kbps
- Register oriented access to controller settings.

4.2 Operating

The CANS01 module uses dedicated identifiers in the range of $0x100_{Hex}$ to $0x254_{Hex}$.

The CANS01 sub module may be accessed by sending a CAN protocol with its receive identifier (from 100 in steps of 10). CANS01 acknowledges the message with a response that uses the transmission identifier (receive ID + 1).

The ID is set by the command Befehl IC3IDBSx.

See table below:

	ID x	Example 1	Example 2
Receive data	from 100 in steps of 10	100	150
Transmission data	from 101 in steps of 10	101	151

The length of a CAN message is eight bytes. For double precision, the CAN protocol has 2 x 8 bytes when writing.

Receive

Received message from a control unit to the MCM01 module.

ID	Byte 1 to 8 data

Transmit

MCM01 sends data as 32 bit integers.

Send message from the MCM01 module to a control unit.

ID + 1	Byte 1 to 8 data

Data types

00_H	unsigned long	32 Bit value
10_H	signed long	31 Bit value plus sign
20_H	double low bytes	double precision: lower 4 bytes
30_H	double high bytes	double precision: higher 4 bytes

Data sizes

Type	Size	Value
Parameter	Byte	value 01_H to 40_H
Axis no.	Byte	higher half byte module no. lower half byte axis no.
Module no..	Byte	01_H to 10_H
Program no.	Byte	01_H to FF_H
Data no.	Byte	
Distance	See data type	signed long or double
Position	See data type	signed long or double
Parameter value	See data type	
Register value	See data type	
Register no.	2 bytes	01_H to $3E8_H$
Output	See data type	
Response data	See data type	

5 Register functions

	Function	Byte								According to *phy*LOGIC™ command (refer to the command reference manual)	
		1	2	3	4	5	6	7	8	Command	Chap.
Send	Read register	40_H	Data type	Register no.		00_H	00_H	00_H	00_H	**R**nnnn**R**	6.15
	Write register	41_H	Data type	Register no.		Register value				**R**nnnn**S**zz	6.15
	Add register	42_H	Data type	Register no.		Register value				**R**nn**+**value **R[R**nn**]+**value	6.16
	Subtract register	43_H	Data type	Register no.		Register value				**R**nn**–**valueFehler! Textmarke nicht definiert. **R[R**nn**]–**value	6.16
	Multiply register	44_H	Data type	Register no.		Register value				**R**nn*****valueFehler! Textmarke nicht definiert. **R[R**nn**]***value	6.16
	Divide register	45_H	Data type	Register no.		Register value				**R**nn**:**value**Fehler! Textmarke nicht definiert.** **R[R**nn**]:**value **R**nn/valueFehler! Textmarke nicht definiert. **R[R**nn**]/**value	6.16
	Raise register to a higher power	46_H	Data type	Register no.		Register value				**R**nn**E**value	6.16
	Register Square root	47_H	Data type	Register no.		Register value				**R**nn**QW**Fehler! Textmarke nicht definiert.	6.16
	Register Sinus	48_H	Data type	Register no.		Register value				**R**nn**SIN**	6.16
	Register Cosines	49_H	Data type	Register no.		Register value				**R**nn**COS**Fehler! Textmarke nicht definiert.	6.16
	Register Tangent	$4A_H$	Data type	Register no.		Register value				**R**nn**TAN**	6.16
	Register Timer Ticker Value (100 µs)	$4B_H$	Data type	Register no.		00_H	00_H	00_H	00_H	**R**nn**STT**	6.16

Function		Byte							
		1	2	3	4	5	6	7	8
Response	Command is valid	Send string (Byte 1 to 4)				Response data type insigned long			
	Command is invalid	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H

6 Interface assignment

*phy***LOGIC**™ commands usually include the device address (e.g. if you use the *phy***LOGIC**™ Toolbox).
In the following chapter the instruction set for each module type is defined with reference to the *phy***LOGIC**™ command reference.

6.1 Master (CPU)

6.1.1 Data set (commands) (8 Byte)

	Function of the commands (8 bytes)	Byte								According to *phy*LOGIC™ command (refer to the command reference manual)	
		1	2	3	4	5	6	7	8	Command	Chap
Send	Read CPU status	00_H	00_H	00_H	00_H	00_H	00_H	00_H	00_H	ST	6.17
	Status and Error Reset	00_H	01_H	00_H	00_H	00_H	00_H	00_H	00_H	STC	6.17
	Controller Reset	01_H	00_H	00_H	00_H	00_H	00_H	00_H	00_H	CR	6.3
	Save controller's parameters	02_H	01_H	00_H	00_H	00_H	00_H	00_H	00_H	SA	6.19
	Start program script	03_H	01_H	00_H	00_H	00_H	00_H	00_H	00_H	QPname A	6.14
	Stop program script	03_H	00_H	00_H	00_H	00_H	00_H	00_H	00_H	QPE	6.14
	Emergency stop of all axes and outputs are set to zero	04_H	00_H	00_H	00_H	00_H	00_H	00_H	00_H	–	

Function		Byte							
		1	2	3	4	5	6	7	8
Response	Command is valid	Send string (Byte 1 to 4)				Response data type insigned long			
	Command is invalid	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H

6.1.2 Status Code (CPU)

Status (2 bytes)	Meaning
0001_H	Command error
0002_H	Range value error
0004_H	Checksum error (CRC)
0008_H	ADDR error (module could not be addressed)
0010_H	Timeout error on bus
0020_H	Bad value error
0040_H	Interface error (frame)
0080_H	Software error
0100_H	Internal program is running
0200_H	Forced switch over to remote via PC
1000_H	Programming error: internal program
4000_H	Input inquiry active (wait for input status)
8000_H	Remote/Local switch on Remote

6.2 Axes Modules I1AM and I4XM

Important: Byte '3' (axis no.) is further subdivided into:

Axis module number: higher half byte (1 to F)

Axis number: lower half byte (1 to 4)

Examples:

I1AM01: axis = 31_H (module 3, 1 axis)
I4XM01: axis = 21_H to 24_H (module 2, axis 1 to 4)

6.2.1 Send Protocol

	Function	Byte 1	2	3	4	5	6	7	8	According to *phy*LOGIC™ command (refer to the command reference manual) Command	Chap
Send	Read axis status	10_H	00_H	Axis no.	00_H	00_H	00_H	00_H	00_H	SEm.a	6.17
	Axis status error reset	10_H	01_H	Axis no.	00_H	00_H	00_H	00_H	00_H	n/a	
	Axis stop normal	12_H	00_H	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aS	6.25
	with emergency stop ramp	12_H	01_H	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aSN	6.25
	Reference run – direction	13_H	00_H	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aR–	6.25
	+ direction	13_H	01_H	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aR+	6.25
	Center via – direction	13_H	02_H	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aR–C	6.25
	Center via + direction	13_H	03_H	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aR+C	6.25
	– direction with encoder	13_H	04_H	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aR–^I	6.25
	+ direction with encoder	13_H	05_H	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aR+^I	6.25

	Function	Byte 1	2	3	4	5	6	7	8	According to *phy*LOGIC™ command (refer to the command reference manual) Command	Chap
Send	Center via – direction with encoder	13_H	06_H	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aR–C^I	6.25
	Center via + direction with encoder	13_H	07_H	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aR+C^I	6.25
	only pulse zero in – direction	13_H	08_H	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aR–I	6.25
	only pulse zero in + direction	13_H	09_H	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aR+I	6.25
	Center on limit switch OFF	13_H	$0A_H$	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aRC-	6.25
	Center on limit switch ON	13_H	$0B_H$	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aRC+	6.25
	Center on limit switch OFF with Encoder track zero	13_H	$0C_H$	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aRC-^I	6.25
	Center on limit switch On with Encoder track zero	13_H	$0D_H$	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aRC+^I	6.25
	Center only in + dir.	13_H	$0E_H$	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aRCW	6.25
	Center only in – dir.	13_H	$0F_H$	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aRCCW	6.25
	Free running – direction	14_H	00_H	Axis no.	00_H	00_H	00_H	00_H	00_H	m.a**L**r	6.25
	Free running + direction	14_H	01_H	Axis no.	00_H	00_H				00_H	00_H

	Function	Byte 1	2	3	4	5	6	7	8	According to *phy*LOGIC™ command (refer to the command reference manual) Command	Chap
Send	Relative positioning with parameter										
	P14 signed long	15_H	10_H	Axis no.	00_H	distance				m.a*rvalue*	6.25
	P14 double low bytes	15_H	20_H	Axis no.	00_H	distance low bytes				m.a*rvalue*	6.25
	P14 double high bytes	15_H	30_H	Axis no.	00_H	distance high bytes				m.a*rvalue*	6.25
	P14 float	15_H	40_H	Axis no.	00_H	distance high bytes				m.a*rvalue*	6.25
	P4 signed long	15_H	11_H	Axis no.	00_H	distance				m.a*rvalue*	6.25
	P4 double low Bytes	15_H	21_H	Axis no.	00_H	distance low bytes				m.a*rvalue*	6.25
	P4 double high Bytes	15_H	31_H	Axis no.	00_H	distance high bytes				m.a*rvalue*	6.25
	P4 float	15_H	41_H	Axis no.	00_H	distance high bytes				m.a*rvalue*	6.25
	Absolute positioning with parameter										
	P14 signed long	16_H	10_H	Axis no.	00_H	position				m.aArvalue	6.25
	P14 double low Bytes	16_H	20_H	Axis no.	00_H	position low bytes				m.aArvalue	6.25
	P14 double high Bytes	16_H	30_H	Axis no.	00_H	position high bytes				m.aArvalue	6.25
	P14 float	16_H	40_H	Axis no.	00_H	position high bytes				m.aArvalue	6.25
	P4 signed long	16_H	11_H	Axis no.	00_H	position				m.aArvalue	6.25

	Function	Byte 1	2	3	4	5	6	7	8	According to *phy*LOGIC™ command (refer to the command reference manual) Command	Chap
Send	P4 double low Bytes	16_H	21_H	Axis no.	00_H	position low bytes				m.aArvalue	6.25
	P4 double high Bytes	16_H	31_H	Axis no.	00_H	position high bytes				m.aArvalue	6.25
	P4 float	16_H	41_H	Axis no.	00_H	position high bytes				m.aArvalue	6.25
	Deactivate axis	17_H	00_H	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aMD	6.25
	Activate axis	17_H	01_H	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aMA	6.25
	Reset power stage	18_H	00_H	Axis no.	00_H	00_H	00_H	00_H	00_H	m.aC	6.25
	Write parameters										
	signed long	19_H	10_H	Axis no.	Para-meter no.	Parameter value				m.aPmm Svalue	6.25
	double low bytes	19_H	20_H	Axis no.	Para-meter no.	Parameter value				m.aPmm Svalue	6.25
	double high bytes	19_H	30_H	Axis no.	Para-meter no.	Parameter value				m.aPmm Svalue	6.25
	Read parameters										
	signed long	$1A_H$	10_H	Axis no.	Para-meter no.	00_H	00_H	00_H	00_H	m.aPmmR	6.25
	double low bytes	$1A_H$	20_H	Axis no.	Para-meter no.	00_H	00_H	00_H	00_H	m.aPmmR	6.25
	double high bytes	$1A_H$	30_H	Axis no.	Para-meter no.	00_H	00_H	00_H	00_H	m.aPmmR	6.25

6.2.2 Response Protocol

Function		Byte							
		1	2	3	4	5	6	7	8
Response	Command is valid	Send string (Byte 1 to 4)				Response data type insigned long			
	Command is invalid	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H

6.2.3 Status Code (Axes)

Status (32 bit)	Meaning
0001_H	busy
0002_H	NOT NOW (command can't be executed because the motor is running)
0004_H	wait for SYNC (only ProfiNet):
0008_H	reference run successful
0010_H	limit switch "+" active
0020_H	limit switch "–" active
0040_H	limit switch "center" active
0080_H	software switch plus has been activated
0100_H	software switch minus has been activated
0200_H	Power stage is ready
0400_H	Axis is in the ramp
0800_H	internal error
1000_H	limit switch error while positioning
2000_H	APS power stage error (short circuit, under voltage)
4000_H	SFI error
8000_H	Encoder error

6.3 Digital IO-Module (DIOM)

6.3.1 Outputs

Function		Byte								According to *phy*LOGIC™ command (refer to the command reference manual)	
		1	2	3	4	5	6	7	8	Command	Chap
Send	Read output status	20_H	00_H	Module no.	00_H	00_H	00_H	00_H	00_H	**AG**1R	6.3
	Read module status	20_H	01_H	Module no.	00_H	00_H	00_H	00_H	00_H	-	-
	Module status reset	20_H	02_H	Module no.	00_H	00_H	00_H	00_H	00_H	-	-
	Set output status	21_H	00_H	Module no..	00_H	Output				**AG**1S	6.3

Function		Byte							
		1	2	3	4	5	6	7	8
Response	Command is valid	Send string (Byte 1 to 4)				Response data type insigned long			
	Command is invalid	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H

6.3.2 Inputs

Function		Byte								According to *phy*LOGIC™ command (refer to the command reference manual)	
		1	2	3	4	5	6	7	8	Command	Chap
Send	Read input status	30_H	00_H	Module no.	00_H	00_H	00_H	00_H	00_H	**EZ**n.a **EG**1R	6.5
	Read module status	30_H	01_H	Module no.	00_H	00_H	00_H	00_H	00_H	-	-
	Reset module status	30_H	02_H	Module no.	00_H	00_H	00_H	00_H	00_H	-	-

Function		Byte							
		1	2	3	4	5	6	7	8
Response	Command is valid	Send string (Byte 1 to 4)				Response data type insigned long			
	Command is invalid	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H

6.4 Analogue IO-Modules (AIOM/AIM/AOM)

6.4.1 Output Modules (AIOM or AOM)

Function		Byte								According to *phy*LOGIC™ command (refer to the command reference manual)	
		1	2	3	4	5	6	7	8	Command	chap.
Send	Read analogue output channel	50_H	00_H	Module No.	Channel no.	00_H	00_H	00_H	00_H	**AZ**n.a **AG**1R **EZ**n.a **EG**1R	6.2 6.2 6.5 6.5
	Read module status	50_H	01_H	Module No	00_H	00_H	00_H	00_H	00_H	-	-
	Analogue I/O error Reset	50_H	02_H	Module No	00_H	00_H	00_H	00_H	00_H	**SI0**mC	6.15
	Set output channel	51_H	00_H	Module No	Channel no.	00_H	00_H	Analogue value		-	-
	Read output channel configuration	52_H	00_H	Module No	Channel no.	00_H	00_H	00_H	00_H	-	-
	Write output channel configuration	53_H	00_H	Module No	Channel no.	00_H	00_H	00_H	Function	-	-

Function		Byte							
		1	2	3	4	5	6	7	8
Response	Command is valid	Send string (Byte 1 to 4)				Response data type insigned long			
	Command is invalid	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H

6.4.1 Input-Modules (AIOM or AIM)

	Function	Byte								According to *phy*LOGIC™ command (refer to the command reference manual)	
		1	2	3	4	5	6	7	8	Command	chap.
Send	Read analogue input channel	60_H	00_H	Module No.	channel no.	00_H	00_H	00_H	00_H	**AZ**n.a **AG**1R **EZ**n.a **EG**1R	6.2 6.2 6.5 6.5
	Read module status	60_H	01_H	Module No.	00_H	00_H	00_H	00_H	00_H	-	-
	Analogue I/O error Reset	60_H	02_H	Module No	00_H	00_H	00_H	00_H	00_H	**SI0**m**C**	6.15
	Read input channel configuration	62_H	00_H	Module No	channel no.	00_H	00_H	00_H	00_H		
	Write input channel configuration	63_H	00_H	Module No.	channel no.	00_H	00_H	00_H	Function		

Function		Byte							
		1	2	3	4	5	6	7	8
Response	Command is valid	Send string (Byte 1 to 4)				Response data type insigned long			
	Command is invalid	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H	FF_H

7 List of Parameters

No.	Meaning	Default
P01	Type of movement (free run, relative / absolute, reference run) 0 = Rotational movement (ignoring limit switches) 1 = Hardware limit switches are monitored for XY tables or other linear systems, 2 limit switches: Mechanical zero and limit direction – Limit direction + 2 = Software limit switches are monitored 3 = Hardware and software limit switches are monitored	0
P02	Measuring units of movement: only used for displaying 1 = step 2 = mm 3 = inch 4 = degree	1
P03	Conversion factor for the thread 1 step corresponds to ... If P03 = 1 (steps) the conversion factor is 1. Computing the conversion factor: $Conversion\,factor = \frac{Thread}{Number\,of\,steps\,per\,revolution}$ Example: 4 mm thread pitch 200-step motor = 400 steps/rev. in the half step mode $Conversion\,factor = \frac{4}{400} = 0.01$	1
P04	Start/stop frequency The start/stop frequency is the maximum frequency to start or stop the motor without ramp. At higher frequencies, step losses or motor stop would be the result of a start or stop without ramp. The start/stop frequency depends on various factors: type of motor, load, mechanical system, power stage. The frequency is programmed in Hz.	400
P05 **P06**	not used	

No.	Meaning	Default
P07	Emergency stop ramp Input for I1AM0x: in 4000 Hz/s steps I4XM01: in 1 Hz/s steps	100 000
P08	f_{max} MØP (mechanical zero point) Run frequency during initializing (referencing) Enter in Hz (integer value) I1AM0x: 40 000 maximum I4XM01: 400 000 maximum	4000
P09	Ramp MØP Ramp during initializing, associated to parameter P08 Input for I1AM0x: in 4000 Hz/s steps I4XM01: in 1 Hz/s steps	4000
P10	f_{min} MØP Run frequency for leaving the limit switch range Enter in Hz	400
P11	MØP offset for limit switch direction + (away from “LIMIT+” switch, towards “LIMIT–” switch) Distance between reference point MØP and limit switch activation Unit: is defined in parameter P02 P11>=0	0
P12	MØP offset for limit switch direction – (away from “LIMIT–” switch, towards “LIMIT+” switch) Distance between reference point MØP and limit switch activation Unit: is defined in parameter P02 P12>=0	0
P13	Recovery time MØP Time lapse during initialization Enter in msec	20

No.	Meaning	Default
P14	f_{max} Run frequency during program operation Enter in Hz (integer value) I1AM0x: 40 000 maximum I4XM01: 400 000 maximum	4000
P15	Ramp for run frequency (P14) Input for I1AM0x: in 4000 Hz/s steps I4XM01: in 1 Hz/s steps	4000
P16	Recovery time position Time lapse after positioning Input in msec	20
P17	Boost (current is defined in P42) 0 = off 1 = on during motor run 2 = on during acceleration and deceleration ramp Remarks: The boost current is set in parameter P42 for internal power stages. You can select with parameter P17 in which situation the controller switches to boost current. P17 = 1 means, the boost current always is switched on during motor run. During motor standstill the controller switches to stop current.	0
P18	Internally used for linear interpolation	
P19	Encoder deviation MØP counter	
P20	Mechanical zero counter This counter contains the number of steps referred to the mechanical zero (MØP). If the axis reaches the MØP, P20 will be set to zero.	0

No.	Meaning	Default
P21	Absolute counter Encoder, multi turn and also for single turn. The value of P22 is extended to P21 by software. The encoder counters have a fixed resolution, e.g. 10 bit (for single-turn encoders: the resolution is bits per turn), then the read value repeats. A saw tooth profile of the the numerical values is produced during a continuous motor running. This course is "straightened" by software. P20 and P21 will be scaled to the same value per revolution by P3 and P39 and are therefore directly comparable, see P36.	0
P22	Encoder counter Indicates the true absolute encoder position. Is only set for A/B encoders to zero (after reset), the absolute encoder remains the value.	0
P23	Software Limit Switch (Axial limitation pos. direction +) If the distance is reached, the run in + direction is aborted. 0 = no limitation	0
P24	Software Limit Switch (Axial limitation neg. direction –) If the distance is reached, the run in – direction is aborted. 0 = no limitation	0
P25	Compensation for play Indicates the distance, the target position in the selected direction is passed over and afterwards is started in reverse direction. 0 = no compensation for play	0

<table>
<tr><th>No.</th><th>Meaning</th><th>Default</th></tr>
<tr><td>P26</td><td>The data transfer rate is set by P26 (ONLY for SSI encoder), by which the encoder is read. The transfer rate is dependent on the length of the cable by which the encoder is connected to the device. The shorter the cable, the encoder can more quickly be read.
Data transfer rate 1 to 10 (= 100 to 1000 kHz)
1 = 100 kHz
2 = 200 kHz
3 = 300 kHz
4 = 400 kHz
5 = 500 kHz
6 = 600 kHz
7 = 700 kHz
8 = 800 kHz
9 = 900 kHz
10 = 1000 kHz</td><td>1</td></tr>
<tr><td>P27</td><td>Limit switch type
NCC: normally closed contact
NOC: normally open contact
<table><tr><th></th><th>LIMIT–</th><th>Center/Ref</th><th>LIMIT+</th></tr><tr><td>0</td><td>NCC</td><td>NCC</td><td>NCC</td></tr><tr><td>1</td><td>NCC</td><td>NCC</td><td>NOC</td></tr><tr><td>2</td><td>NOC</td><td>NCC</td><td>NCC</td></tr><tr><td>3</td><td>NOC</td><td>NCC</td><td>NOC</td></tr><tr><td>4</td><td>NCC</td><td>NOC</td><td>NCC</td></tr><tr><td>5</td><td>NCC</td><td>NOC</td><td>NOC</td></tr><tr><td>6</td><td>NOC</td><td>NOC</td><td>NCC</td></tr><tr><td>7</td><td>NOC</td><td>NOC</td><td>NOC</td></tr></table></td><td>0</td></tr>
<tr><td>P28</td><td>Axis options
0 = Power stage is deactivated after power on
1 = Power stage is activated after power on</td><td>0</td></tr>
<tr><td colspan="3">P29 not used</td></tr>
</table>

<table>
<tr><th>No.</th><th>Meaning</th><th>Default</th></tr>
<tr><td>P30</td><td>For I4XM01 only!
Frequency band setting
0 = manual
1 = automatic
<u>Remark</u>:
It is recommended to work with the automatic setting mode.
For each run frequency (P14) and ramp (P15) the controller automatically selects suitable settings.</td><td>1</td></tr>
<tr><td>P31</td><td>For I4XM01 only!
Frequency and ramp predivider (only if P30 = 0, manual)
This parameter changes the predivider which supplies the hardware (frequency generated) with a clock of 20 MHz derived.</td><td>3</td></tr>
<tr><td></td><td>
<table>
<tr><th>P31</th><th>Run frequency</th><th>resolution</th><th>predivider</th></tr>
<tr><td>0</td><td>1 Hz … 8 kHz</td><td>⅛ Hz</td><td>2440</td></tr>
<tr><td>1</td><td>1 Hz … 16 kHz</td><td>¼ Hz</td><td>1220</td></tr>
<tr><td>2</td><td>1 Hz … 32 kHz</td><td>½ Hz</td><td>609</td></tr>
<tr><td>3</td><td>1 Hz … 65 kHz</td><td>1 Hz</td><td>304</td></tr>
<tr><td>4</td><td>2 Hz … 130 kHz</td><td>2 Hz</td><td>152</td></tr>
<tr><td>5</td><td>4 Hz … 260 kHz</td><td>4 Hz</td><td>75</td></tr>
<tr><td>6</td><td>8 Hz … 520 kHz</td><td>8 Hz</td><td>37</td></tr>
<tr><td>7</td><td>16 Hz … 1 MHz</td><td>16 Hz</td><td>18</td></tr>
<tr><td>8</td><td>32 Hz … 2 MHz</td><td>32 Hz</td><td>9</td></tr>
<tr><td>9</td><td>64 Hz … 4 MHz</td><td>64 Hz</td><td>4</td></tr>
</table>
The parameter can be used for individual settings when automatic frequency band setting for the specific application is not appropriate.
</td><td></td></tr>
<tr><td>P32</td><td>TBD</td><td>1</td></tr>
</table>

No.	Meaning	Default
P33	TBD	1
P34	Encoder type 0 = no encoder 1 = incremental 5.0 V 2 = incremental 5.5 V 3 = serial interface SSI binary Code 5.0 V 4 = serial interface SSI binary Code 5.5 V 5 = serial interface SSI Gray Code 5.0 V 6 = serial interface SSI Gray Code 5.5 V 7 = EnDat 5 V 8 = EnDat 5.5 V	0
P35	Encoder resolution for SSI and EnDat encoder Enter max. encoder resolution in Bit (max. 48 Bit) Special feature EnDat: if the parameter is set to zero, the controller uses the resolution which is read from the connected instrument.	10
P36	Encoder function This parameter specifies the use of P21 as a pure counter or whether its value is continuously compared with the value of the P20 counter, if the counter values vary too much, the motion is aborted with an error message. 0 = counter 1 = counter+SFI	0
P37	Encoder tolerance for SFI Enter tolerance value for SFI evaluation Input: tolerance value for SFI-evaluation in the selected resolution (P3 * P20). If P21 is used for step failure indication the scale of the counter P20 * P3 must be equal to the scale of the counter P21 * P39 and P21 must be set to zero after initialization of the scaling (or can be set to the same value as P20). e.g. scaling to 360°/rev.: Motor 200 steps per revolution, 1/20 step, →P3 = 360 / 200 / 20 = 0.09, encoder 10 bit / rev. → P39 = 360 / 2^{10} = 0.3515625	0

No.	Meaning	Default
P38	Encoder preferential direction of rotation 0 = + (positive) 1 = – (negative)	0
P39	Encoder conversion factor 1 increment corresponds to ... Computing the conversion factor: $Conversion\,factor = \frac{Thread}{Encoder\,steps\,per\,revolution}$	1
P40	Stop current in 0.01 A_{eff} steps depending on the power stage I1AM01: 0 to 250 (0 to 2.5 A_{eff}) I1AM02: 0 to 350 (0 to 3.5 A_{eff}) ZMX^{+}: 0 to 630 (0 to 6.3 A_{eff}) MCD^{+}: 0 to 63 (0 to 6.3 A_{eff}) APS: 0 to 350 (0 to 3.5 A_{eff})	2
P41	Run current in 0.01 A_{eff} steps I1AM01: 0 to 250 (0 to 2.5 A_{eff}) I1AM02: 0 to 350 (0 to 3.5 A_{eff}) ZMX^{+}: 0 to 630 (0 to 6.3 A_{eff}) MCD^{+}: 0 to 63 (0 to 6.3 A_{eff}) APS: 0 to 350 (0 to 3.5 A_{eff})	6
P42	Boost current in 0.01 A_{eff} steps I1AM01: 0 to 250 (0 to 2.5 A_{eff}) I1AM02: 0 to 350 (0 to 3.5 A_{eff}) ZMX^{+}: 0 to 630 (0 to 6.3 A_{eff}) MCD^{+}: 0 to 63 (0 to 6.3 A_{eff}) APS: 0 to 350 (0 to 3.5 A_{eff})	10
P43	Current hold time in msec	20

No.	Meaning	Default
P44	For I4XM01 only! Origin of the Control pulses for the axis 0 = 1:1 (Input=Output) 1 = from X 2 = from Y 3 = from Z 4 = from U 5 = from external	0
P45	Step resolution 1 to 512 0 = 1/1 step 7 = 1/16 step 1 = 1/2 step 8 = 1/20 step 2 = 1/2.5 step 9 = 1/32 step 3 = 1/4 step 10 = 1/64 step 4 = 1/5 step 11 = 1/128 step 5 = 1/8 step 12 = 1/256 step 6 = 1/10 step 13 = 1/512 step (e.g. APS01) Important: for I1AM: step resolution from 1/1 to 1/128 step	3
P46	not used	
P47	not used	
P48	not used	
P49	Power stage temperature in 1/10 °C	(read only)
P50	Divider for Control pulses only for I4XM01 Control pulses $_{Output}$=1/(n+1) * Control pulses $_{Input}$ 0 : 1/(0+1)=1 1: 1/(1+1)= 1/2 2: 1/(2+1) =1/3 3: 1/(3+1)=1/4 4: 1/(4+1)=1/5 5: 1/(5+1)=1/6	n=0
P51	Pulse width: (n+1)*100 ns only for I4XM01 n: 0….31 e.g. n=19: (19+1)*100 ns=2000 ns= 2µs -> F_{max}=1/(2*2 µs)=250 kHz	n=19
P52	Internally used for trigger position.	

No.	Meaning	Default
P53	Power stage monitoring 0 = off 1 = on	1
P54	Motor temperature in 1/10 °C -999999: Temperature module not existent -9999: negative overflow or temperature lower -220 °C at PT100 9999: positive overflow or temperature higher +390 °C at PT100	-999999 (read only)
P55	Motor temperature **warning** in 1/10 °C If the motor warmed up to a defined temperature value, a warning will occur. We recommend operating the motor until it is cooled again.	0
P56	Motor temperature **shut-off** in 1/10 °C If the motor warmed up to a defined temperature value, the controller switches off and the power stage must be reset.	0

8 Warranty, Disclaimer and Registered Trademarks

8.1 Disclaimer

Phytron GmbH has verified the contents of the manual to match with the hardware and software. However, errors and omissions are exempt and Phytron GmbH assumes no responsibility for complete compliance. The information contained in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

8.2 Warranty

The *phy***MOTION**™ modules are subject to **legal warranty**. Phytron will repair or exchange devices which show a failure due to defects in material or caused by the production process. This warranty does not include damage caused by the customer, for example, not intended use, unauthorised modifications, incorrect handling or wiring.

8.3 Registered Trademarks

In this manual several trademarks are used which are no longer explicitly marked as trademarks within the text. The lack of these signs may not be used to draw the conclusion that these products are free of rights of third parties. Some product names used herein are for instance.

- *phy***MOTION**™ is a trademark of the Phytron GmbH.
- *phy***LOGIC**™ is a trademark of the Phytron GmbH.
- Microsoft is a registered trade mark and WINDOWS® is a trade mark of the Microsoft Corporation in the USA and other countries.

9 Index