S I X P A C K / Q U A D P A C K

Manual Version: 2.43
October 15th, 2004

TRINAMIC
MOTION CONTROL

GmbH & Co KG
Sternstraße 67
D - 20357 Hamburg, Germany
Phone +49-40-51 48 06 - 0
FAX: +49-40-51 48 06 - 60
http://www.trinamic.com

SIXpack 6 Channel 800mA
QUADpack 4 Channel 1500mA

Trinamic Motion Control GmbH & Co KG
Sternstraße 67
D – 20357 Hamburg, Germany
Phone +49-40-51 48 06 - 0
FAX: +49-40-51 48 06 - 60
http://www.trinamic.com

Table of Contents:

1 BRIEF DESCRIPTION

SIXpack / QUADpack are highly integrated stepper motor controllers for six respectively four 2-phase stepper motors with a coil current of 800 mA respectively 1500 mA each. A DSP supported by special hardware allows a powerful function set and a wide stepping frequency range for all motors. Both PACKs are equipped with RS 232, RS 485 and CAN-Interface.

2 TECHNICAL DATA

ramp profile:	automatic 3-phase ramps (32 Bit signed position resolution) with programmable parameters for maximum frequency and acceleration for each channel; alternatively user defined ramps; automatic reference search (reference switch)
stepping frequency:	full step frequencies from 0.3 Hz 12.5 kHz
step type:	microstepping resolution 1/16 with user-programmable motor characteristics or sine-table
current control:	programmable acceleration-dependent motor current; programmable stand-by timer for current reduction
interfaces:	RS 232 or RS 485; CAN
protocols:	barcode-reader interface via RS 232 in CAN-mode possible
I/O-lines:	10 bit analogue input for ratiometric measurements or stop functions; digital input for reference switch; separate analogue input; digital I/O and digital output; LED-„Interface active“; 7-segment display (number of active motors, DP(=decimal point) = reference search); 1 Ready Output (Open Collector)
power supply:	15 to 40 V DC; ca. 6W without load; max. ca. 5A, depending on motor type
motor current SIXpack:	software configurable ca. 50-800 mA per channel (peak coil current); constant current (chopper, ca. 23 kHz), motor driver thermally protected
motor cur. QUADpack:	software and DIP configurable ca. 100-1500 mA per channel (peak coil current); constant current (chopper, ca. 40 kHz), motor driver thermally protected
motor type:	bipolar 2-phase motors
motor connectors:	8-pin single-in-line (motor, reference switch, A/D, 5V supply (15 mA))
temperature range:	up to 85°C with reduced current or forced cooling of board
dimensions:	board: W: 126, D: 180, H: 25 mm housing: W: 152, D: 180, H: 36 mm

Trinamic Motion Control GmbH & Co KG
Sternstraße 67
D – 20357 Hamburg, Germany
Phone +49-40-51 48 06 - 0
FAX: +49-40-51 48 06 - 60
http://www.trinamic.com

3 TEST REPORTS

EMV Services GmbH	Test report	Reference	Date	Page
Emission Immunity	No. 00/9091-5	EMV-00/9091-5	May. 09, 00	2 / 26

Customer: Trinamic
Electronic System Design GmbH
Silemstraße 76a
D-20257 Hamburg

Equipment under test: Quadpack Vers.1.0, S/N: Prototype

Date of test: April 13, and May 09, 2000

Test site: EMV Services GmbH
Harburger Schloßstr. 6-12
D-21079 Hamburg

Test personnel:	**Tel.**	**Fax**	**E-mail**
Dipl.-Ing. H. Meisel	040/766293432	040/76629506	meisel@maz-hh.de
Dipl.-Ing. Z. Wang	040/766293431	040/76629506	wang@maz-hh.de

Applied standards:

EN 50 081-2 (1993): **Generic emission standard; Part 2: industrial environment,**

FCC (1997): **Limit for digital devices, Class A,**

EN 61000-6-2 (2000): **Generic immunity standard, Industrial environment:**

- EN 61000-4-2 (1995): Electrostatic discharge immunity test,
- EN 61000-4-3 (1996): Radiated, radio-frequency electromagnetic field - immunity test,
- EN 61000-4-4 (1995): Electrical fast transient/burst immunity test,
- EN 61000-4-5 (1995): Surge immunity tests,
- EN 61000-4-6 (1996): Immunity to conducted disturbances, induced by radio frequency fields.

Test results:

Emission:
The device complies with the limits for conducted emission.
The device complies with the limits for radiated emission.

Immunity:
The requirements of Immunity:

- To electrostatic discharge are fulfilled,
- To radiated, radio-frequency electromagnetic field are fulfilled,
- To electrical fast transient/burst are fulfilled,
- To surge are fulfilled,
- To conducted disturbances, induced by radio frequency fields are fulfilled.

Dr. E. Sauer
Lab manager

p.p. Dipl.-Ing. Z. Wang

EMV Services GmbH
Ein Unternehmen der TÜV Nord Gruppe
Harburger Schloßstraße 6-12
D-21079 Hamburg

EMV Services GmbH	Test report	Reference	Date	Page
Emission Immunity	No. 00/9091-6	EMV-00/9091-6	Nov. 1, 00	2 / 23

Customer: Trinamic
Electronic System Design GmbH
Silemstraße 76a
D-20257 Hamburg

Equipment under test: Sixpack Rev.1.3, S/N: Prototype

Date of test: October 17, 2000

Test site: EMV Services GmbH
Harburger Schloßstr. 6-12
D-21079 Hamburg

Test personnel:	**Tel.**	**Fax**	**E-mail**
Dipl.-Ing. J. Plambeck	040/766293431	040/76629506	plambeck@emv-services.de

Applied standards:

EN 50 081-2 (1993): **Generic emission standard; Part 2: Industrial environment,**

FCC (1997): **Limit for digital devices, Class A,**

EN 61000-6-2 (2000): **Generic immunity standard, Industrial environment:**

- EN 61000-4-2 (1995): Electrostatic discharge immunity test,
- EN 61000-4-3 (1996): Radiated, radio-frequency electromagnetic field - immunity test,
- EN 61000-4-4 (1995): Electrical fast transient/burst immunity test,
- EN 61000-4-6 (1996): Immunity to conducted disturbances, induced by radio frequency fields.

Test results:

Emission:
The device complies with the limits for conducted emission.
The device complies with the limits for radiated emission.

Immunity:
The requirements of Immunity:

- To electrostatic discharge are fulfilled,
- To radiated, radio-frequency electromagnetic field are fulfilled,
- To electrical fast transient/burst are fulfilled,
- To conducted disturbances, induced by radio frequency fields are fulfilled.

Dr. E. Sauer
Lab manager

p.p. Dipl.-Ing. J. Plambeck

4 CONNECTOR ASSIGNMENT

- Motor Connectors:
  - Attention: wrong pinning leads to damage to the pcb
  - **never pull the connectors during operation!**

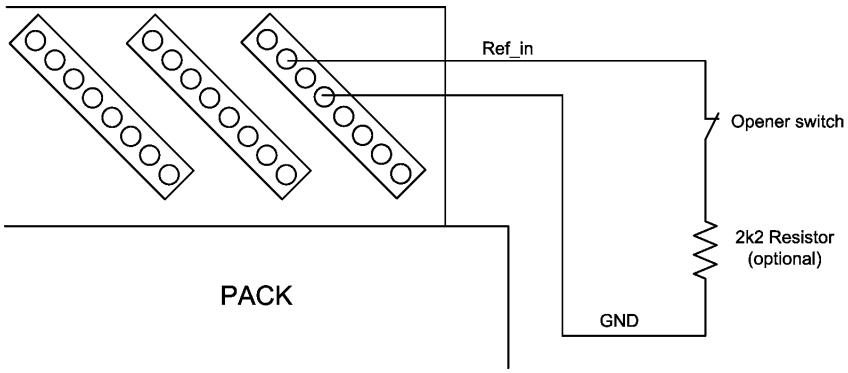
PCB edge

① Analog In (0..5V)
② Referenz In (22K Pullup, TTL)
③ 5V (15mA, 33 OHM)
④ GND
⑤ PHB2
⑥ PHB1
⑦ PHA2
⑧ PHA1

Phase (coil) A of the motor is wired to connectors PHA1 and PHA2, Phase (coil) B to PHB1 and PHB2.

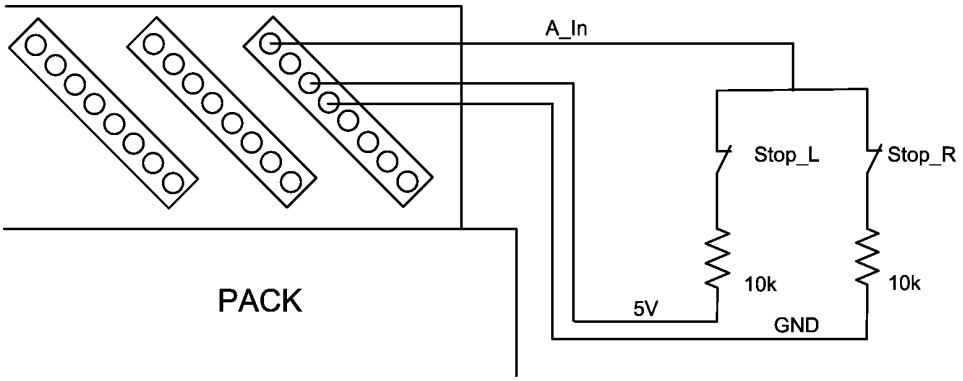
- Electrical Reference Switch

The reference switch is connected to the pins „Ref_In“ and „GND“. Optionally a series resistance of about 2.2kOhms can be inserted to match EMC demands. In general using an opener, i.e. a normally closed switch is advisable. So broken cables can be detected. The reference input is equipped with a Schmitt-trigger.



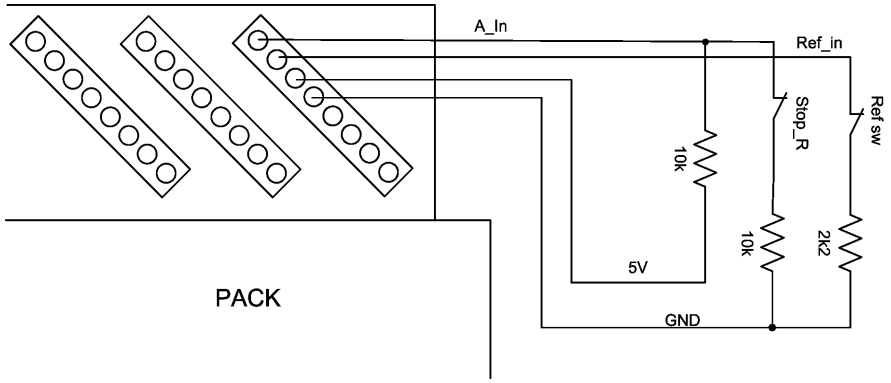
- Wiring with Stop-Switches

To prevent driving beyond the ends of a linear axis stop-switches can be used. They are connected to pin „A_In“ of the motor connector. Again, openers should be used as stop-switches for the reason mentioned above.



- Wiring with combined Stop-/Reference Switch when using **Openers**

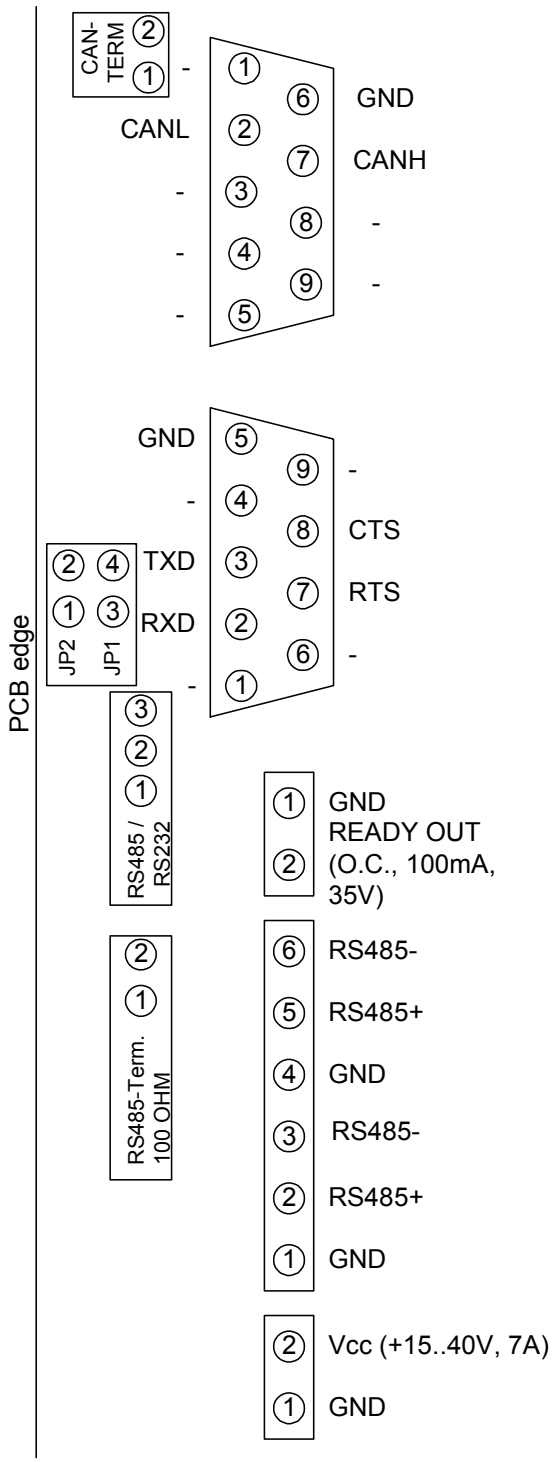
Mounting the reference switch at one of the ends of the axis it can be concurrently used as stop-switch thus saving the respective stop-switch.



Note: In this circuit the reference switch is concurrently used as left stop switch if flag *StopNull* is set. The flag *NullPositive* has to be set, to match the opener. Reference search is done in left direction (flag *NullLeft* set). The insertion of an optional filter combination is shown here. The capacity could for example be 100nF.

Trinamic Motion Control GmbH & Co KG
Sternstraße 67
D – 20357 Hamburg, Germany
Phone +49-40-51 48 06 - 0
FAX: +49-40-51 48 06 - 60
http://www.trinamic.com

- Power Supply Connector, RS485 RS 232-DSUB-9M and CAN-DSUB-9F

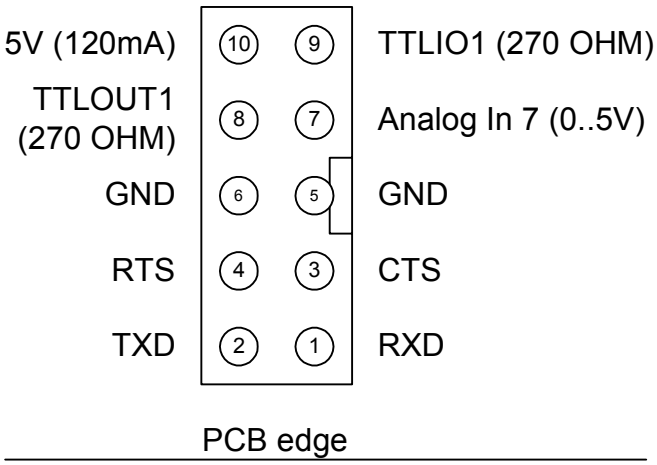


⇒ Current Control for QUADpack

The maximum coil current for the QUADpack can be set in steps of 0mA (0%), 500mA (33%), 1000mA (66%) and 1500mA (100%). A fine adjustment is done by software.

Motor	1		2		3		4	
Dip-SW	1	2	3	4	5	6	7	8
I=100%	ON	ON	ON	ON	ON	ON	ON	ON
I=66%	OFF	ON	OFF	ON	OFF	ON	OFF	ON
I=33%	ON	OFF	ON	OFF	ON	OFF	ON	OFF
I=0	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

⇒ RS 232-Interface on the Motor Connector Side



Type of connector: ribbon cable-connector for 10-pin header with 2,54 mm pitch

⇒ How to obtain the Connectors

AMP Connectors (for motor and reference inputs)
order number:

0-0770602-8 (Case)
0-0770666-1 (Crimp contact)
0-0058517-1 (Crimp tool)

5 SYSTEM START UP

5.1 System Start Up / Notes

When the PACK is supplied with power it runs an internal initialization and a self-test of the internal processor-system starts. If executed successfully a “0“ appears in the LED-display after a second. The PACK is operational now and can receive user commands.
Defective motor drivers can not be detected by the self test. Should the motor turn on and off during operation, a constant high motor current or insufficient cooling of the drivers could be the problem. The motor driver chip turns itself off for a short time when overheated. To allow higher constant motor current the motor drivers of the PACKs are cooled with a heat sink. A forced air cooling can additionally improve the maximum current.
When an application requires detection of temporarily interrupted power supply of the PACK, this can be done for example by signaling via external TTLOUT1 by programming it to a negative level. The RS 232-interface usually receives a 0-byte after hardware reset.

5.2 Selecting Motors

When selecting motors, consider stepper motors with the lowest inductance possible, i.e. low coil resistance, to obtain smoothest movements and the maximum possible rpm. On the other hand low coil resistance lowers the torque. Therefore you should choose the motor with the lowest inductance possible which delivers the needed torque at a coil current of approx. 600 mA respectively 1000mA. Highest possible operating voltage of the PACK results in high rpm also. With higher coil resistance or a too low operating voltage the duty factor of the chopper drivers increases. When exceeding 50% a cheering noise can occur in the coils.

5.3 Length of Wires

motor wires:	must be < 3m (use twisted pair wire)
RS 232:	must be <3m
CAN, RS485:	can be >30m

5.4 Cooling

If the continuous current for drive and stop is higher than 1A with QUADpacks or higher than 500 mA with SIXpacks, the PCBs must be cooled sufficiently.
This can be achieved by mounting the PACK vertically, so that the motor supplies are located on top.
With continuous full load cooling will be required.

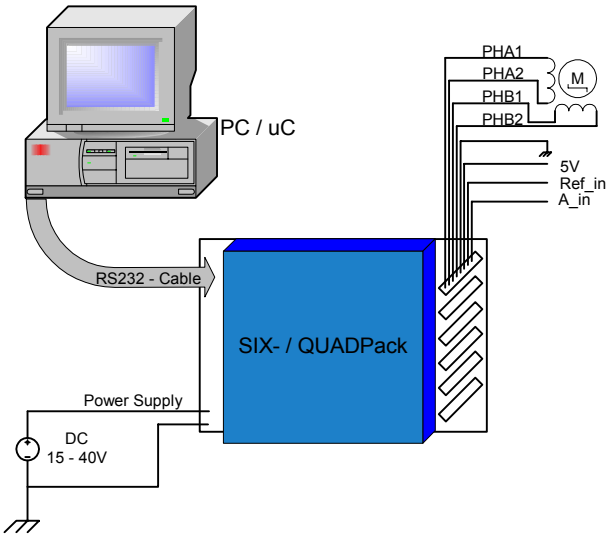
5.5 Grounding

For a good ESD protection the electronics must be connected effectively with ground. Therefore two holes are provided on the PCB with ground contacts.
If the electronics is delivered without housing, these two screws must be connected to ground.
If the electronics are built-in the housing the tape must be taken away of the two fixing drilled holes on the back side. The electronics must now be grounded via these two blank areas.

5.6 Improvement of the EMC-Conduction

To improve the cable-bound conducted emission, a ferrite-clip should be clipped over the supply circuit.

5.7 Example for a TEST - Setup



5.8 Further Information:

For further Information please view our homepage (**www.trinamic.com**). You will find help under “frequently asked questions”. You also have the possibility to send us an e-mail via a contact sheet located on the same site.

6 CONTROL

6.1 Control via RS 485 or RS 232-Interface

The RS 485 interface is a bi-directional 2-wire interface and can handle up to 255 slave devices in half-duplex mode. The RS 232 interface can be used accordingly, however it is not possible to connect multiple transmitters to the receiver input. The baud-rate is pre-configured to 19200 baud. It can be changed via command.
Instructions consist of a 9 byte word, which in turn consist of the address of the unit, a command byte and if required parameters with a length of up to 7 bytes.

Address	command	P0	P1	P2	P3	P4	P5	P6

The command word always has to be completely transmitted during a parameterized timeout (s. CMD $41). It will be aborted and not interpreted, when a break-code is received. If errors occur the interface can be newly synchronized via break-code.
The address of the unit can be set via rotary switches (scanned on reset).
During parameter read out an instruction will be transmitted only after an adjustable transmitter switch-over time (s. CMD $40; pre-set to 6ms) has passed. This allows the transmitter to switch to receiving mode. Ditto for the opposite direction: The PACK continues to drive the line for a pre-set time after transmitting a message. The direction can be checked at the RTS-line of the RS 232-interface (negative = PACK is in sending mode). The CTS-line will be ignored.
When a valid command word is received, the status LED flashes.

6.2 Control via CAN-Interface

The integrated CAN-Controller supports the full-CAN-specifications 2.0A with 11 address bits. Telegrams with a fixed length of 8 bytes are used. The address of the unit (upper 8 address bits) can be set via rotary switches (scanned on reset). The lower 3 address bits are fixed to “000“. Take care: According to the CAN standard 0 is no valid address! Address range: $008 to $7F8 (in increments of 8).
After receiving the first valid instruction via CAN, control via RS 232 or RS 485 will be terminated. The CAN response address is transferred to PACK in a 8 bit format, like at RS 232 / 485. For responding the address is shifted to the left by 3 bits, resulting in the same address range as defined above. If continuous error conditions occur, CAN and RS 232 / 485 will be newly initialized.

⇒ **Setting of the Baudrate via Jumper**

Baudrate	JP1	JP2
(1 Mbit/s (*))	x	x
500 kbit/s (*)	-	x
125 kbit/s (*)	x	-
250 kbit/s (default)(*)	-	-

(*) Note: The PACK has an internal cache for 16 CAN-commands, internal processing time is approx. 2 ms per command.

In order to get the actual jumper-configurations send CMD $30.

6.3 RS 232-Remote Control via CAN-Interface

The RS 232-interface can be controlled via CAN. Therefore the baud-rate is set via command “RS 232-change baud-rate“. Only 8 bit, 1 stop bit and no parity is possible. Of course 7 bit and parity could be

simulated by the user. The response address for bytes received via RS 232 and the packet size for transferring received bytes are configured by a separate instruction.

To forward bytes received via CAN to RS 232, the CAN-address of the PACK is incremented by 1, i.e. the lower 3 bits are “001“. Every byte which is received with this address will be transferred to the RS 232-interface. 1 to 8 bytes can be transferred at once. Please note that the RS 232-interface needs sufficient time before the next block is transmitted. To be sure that the RS 232-cache is empty, it can be checked via command. There is no CTS-handshake, however the CTS-line can be read-out (s. CMD $44).

Bytes received via RS 232 will be sent to the pre-set response address, as soon as the pre-set number of bytes has been received. Incorrect messages will be ignored now. If the configurable RS 232-timeout has expired, remaining bytes will be sent (→ see CMD $41).

7 PROGRAMMING

7.1 Hints for Programming

⇒ **Strategy for Parameter Setting**

The Pack can be parameterized for standard applications with a few commands since it is pre-set with default values. However these default values should not substitute a thorough configuration of all parameters in a given application. Normally the following parameters should be configured for your application:

⇒ **Setting of Motor Current**

Configure maximum current (s. CMD $10) and current control (s. CMD $11) as needed. The minimum current (which provides proper microstepping) selected by current control is 19% (Index 6) for every parameter.

⇒ **Velocity Configuration (global)**

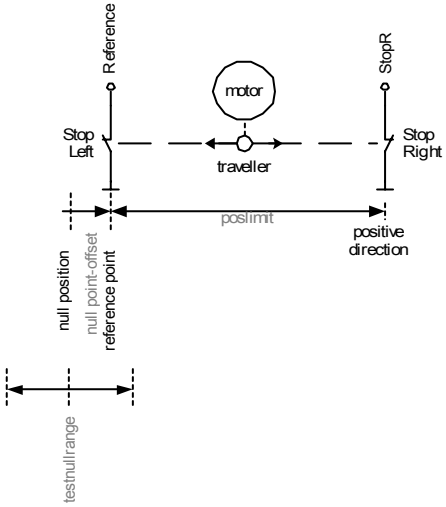
Calculate *clkdiv* (s. CMD $12) with the step frequency formula (s. begin of chapter Instruction Set) so that the required maximum step velocity is achieved with v_i = 511 and small values div_i = 0 or 1.

⇒ **Setting of starting Velocity, max. Velocity and max. Acceleration**

These values (s. CMD $13 and $14) should be adapted to the motor type, mechanical load, and so on. As a reference value div_i should be set so, that the maximum velocity *vmax* could be set between 256 and 511. This way maximum resolution is obtainable. Then the acceleration *amax* can be configured. The velocity *vstart* should not be set too low.

⇒ **Setting of Motor Parameters and Reference Search Parameters**

These settings describe the axis type, the reference search, and so on. For time saving purposes both, fast reference search *FastRef* (s. CMD $15 P6, Bit1) should be activated and the maximum velocity for this reference search should be set. To avoid errors caused by vibrations of the motor during fast reference search, de-bouncing of the reference switch FilterSwitch (s. CMD $15 P5, Bit7) should be activated, too, and the mask for reference point de-bouncing (s. CMD $16) should be programmed with an applicable value. *vmin* (always used with predivider div_i set to 3) will be used while exactly locating the reference switch. The fastest possible *vmin* will be choosen automatically when its value is set to 0.



Graphic assumes null-left Flag is set and null point offset is positive(s. **CMD $15**).

In this configuration the reference switch is reliably closed at position null.

Note: testnullrange >= width of reference switch!

settings: null point-offset, poslimit -> CMD $15
testnullrange -> CMD $18

- The reference switch defines the zero position. The zero position can be moved further into the switch using the *nulloffset* setting. If *testnullbit* is set it must be active at the end of *T0* and the delay time of the filter.
- Activation of the switch is only allowed in the *testnull*range to *testnull* around the zero position. If you reference to the edge of the switch and never exceed the zero position the *testnull* range can be choosen around 1-2 fullsteps * 16. In all other cases you must choose it at least slightly larger than the active area of the reference switch or half of this for *nullcenter* motors.
- The reference search requires proper *poslimit* (0..0x7FFFFFFF) settings! For cyclic axis you must set *poslimit* to the number of microsteps per revolution, for linear axis it should cover at least your whole intended driving range to avoid unintended or interrupted reference drives.

⇒ **Problems with fast Search for Reference**

The fast search for reference will function properly only if CMD $15 and $16 are set correctly, especially those for the reference switch. Also is it sensitive to noise pulses in the wiring of the reference switch – should the fast reference search stop abruptly, anti-noise measures have to be taken for the reference switch input.

⇒ **Interlacing of Requests**

Requests must not be interlaced. Each request should wait for the response of the *PACK* before transmitting a new command. However a delayed response with RS 232 may be outstanding in parallel.

⇒ **Default Values**

For testing purposes here is a list of default values for motor parameters:

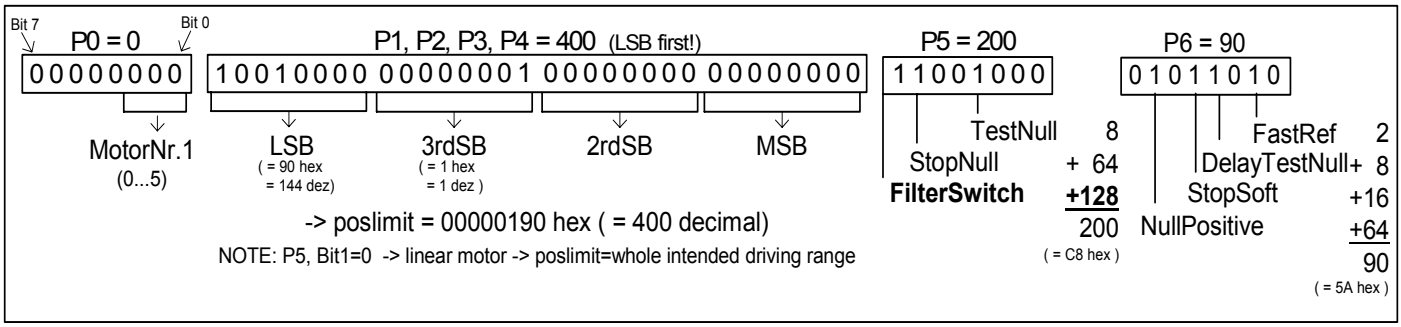
```
clkdiv=5; div=2;                  // 26 kHz microstep-frequency
vstart=5;                         // starting with 254 Hz (should be >=8)
amax=128; vmax=511;               // increments v by 128/16=8 each 2 ms
vmin=4; vrefmax=100;              // 102 Hz /  5086 Hz for reference drive

poslimit=400*16;            // 400 full- = 6400 microsteps/revolution
testnullrange=15*16;        // ignore switch in range  –240 ... 240

Peak current=128;           // define 100% curr. control as 400 mA
T0=500;                     // wait 1000 ms before standby
I0=00%(!);                  // waste no energy for unused motors
I1=50%;                     // power stopped motors with 200 mA
I2=75%;                     // power v const. motors with 300 mA
I3=100%;                    // power accelerat. motors with 400 mA

motortype= Delayedtest0 | NullCenter | Filterswitch | FastRef;

De-bounce mask=$0FFF;       // Filter delay 12-1 cycles = 22 ms
Readymask=$3F;              // check any active motor
Refer.-Readymask=$3F;       // check any referencing motor

propdiv=8;                  // v   = position-difference / 8
intdiv=129;                 // v += pos-difference integral / 129
intclip=129;                // clip pos-difference integrals > 129
intinpclip=1;               // integrate pos-difference of max. 1
```

All other values are set to 0, i.e. the functions are disabled.

7.2 Examples

Attention: All 9 Bytes must be sent to the interface, otherwise the PACK does not recognize the command and waits for the missing bytes.
$ indicates that the value is in hexadecimal notation!

⇒ **Setting motor parameters**
CMD $15 contains information about the motor and settings for the reference drive.
For more details see Hints for Programming and CMD $15 in the Instruction set!



Pseudocode:

```
SendToPack(address);        // Address of the Pack(Six- or Quadpack)
SendToPack($15);            // Command in hexadecimal notation
SendToPack(P0);             // Motor number (0...5)
SendToPack(P1);             // poslimit LSB
SendToPack(P2);
SendToPack(P3);
SendToPack(P4);             // poslimit MSB
SendToPack(P5);             // further settings, for more information read the instruction set
SendToPack(P6);
```

⇒ **Navigating the motor**

CMD $23 prompts the concerned motor to drive to the position, which stands in P1 ... P4.

Pseudocode:

```
sendToPack(address);            // Address of the PACKs
sendToPack($23);                // Command for starting a trapezoidal Ramp
sendToPack(motnr);              // Number of the concerned motor (0...5)
sendToPack(destinationLSB);     // Least significant Byte of the target position
sendToPack(destination3rdSB);
sendToPack(destination2ndSB);
sendToPack(destinationMSB);     // Most significant Byte
sendToPack(0);                  // fill 9 bytes
sendToPack(0);                  // fill 9 bytes
```

⇒ **Inquiring the actual position of a motor**

CMD $20 returns the 4-byte value with the actual position and status of the concerned motor. In addition P6 specifies whether a stop-switch was active. This is e.g. when the motor has lost steps and if during driving back to the real null-point the switch is found too early.

Pseudocode:

```
sendToPack(address);            // Address of the PACKs
sendToPack($20);                // Command for inquiring actual position and action of one motor
sendToPack(motnr);              // Number of the concerned motor (0...5)
sendToPack(receiver);           // address of the receiver
sendToPack(0);                  // fill 9 bytes
sendToPack(0);                  // fill 9 bytes
sendToPack(0);                  // fill 9 bytes
sendToPack(0);                  // fill 9 bytes
sendToPack(0);                  // fill 9 bytes

cmd          = receiveFromPack();   // should be $20
motnr        = receiveFromPack();   // should be the same number as the sent
posakt_byte1 = receiveFromPack();   // LSB of the actual Position
posakt_byte2 = receiveFromPack();
posakt_byte3 = receiveFromPack();
posakt_byte4 = receiveFromPack();   // MSB of the actual Position
act_action   = receiveFromPack();   // Information about the actual action of the motor
stop         = receiveFromPack();   // is 1 when null-switch is active
```

⇒ **Starting a two axis interpolated movement**

Linear motions with multiple axes can be driven. For this, the destinations have to be set via CMD $26 and then the trapezoidal Ramp can be started via CMD $50. In the example the axis 1 is navigated to position 10000 and in parallel the axis 2 to position 2000.

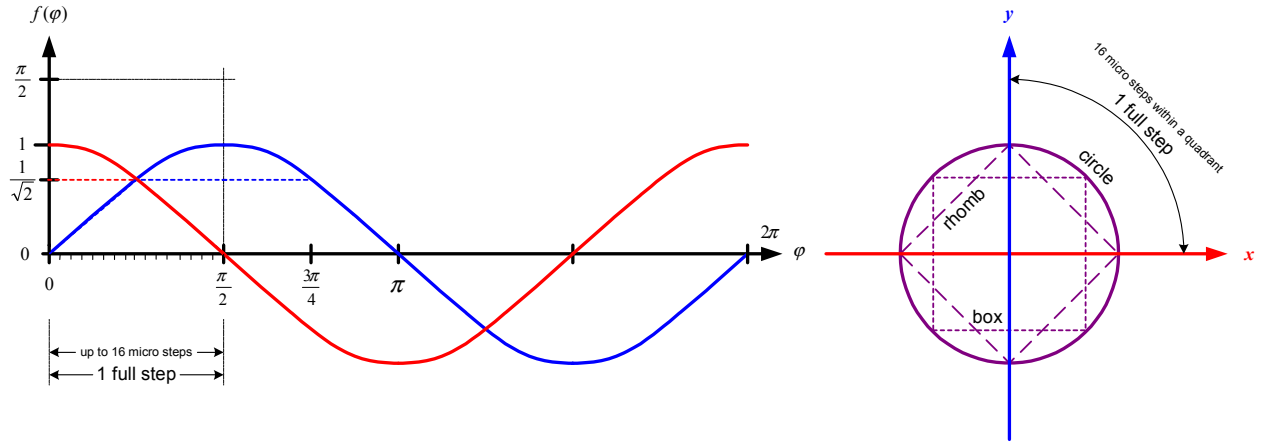
Pseudocode:

```
sendToPack(address);        // Address of the PACKs
sendToPack(0$26);           // Command for setting the destination
sendToPack(0);              // Motor 1
sendToPack($E8);            // 232
sendToPack($03);            // 3*256=768
sendToPack($00);
sendToPack($00);
sendToPack(0);              // fill 9 bytes
sendToPack(0);              // fill 9 bytes

sendToPack(address);        // Address of the PACKs
sendToPack($26);            // Command for setting the destination
sendToPack(1);              // Motor 2
sendToPack($D0);            // 208
sendToPack($07);            // 7*256=1792
sendToPack($00);
sendToPack($00);
sendToPack(0);              // fill 9 bytes
sendToPack(0);              // fill 9 bytes

sendToPack(address);        // Address of the PACKs
sendToPack($50);            // Command for multi-axis Interpolation
sendToPack($03);            // 0000 0011=3, e.g. Mask for motor 1 and 2
sendToPack(0);              // fill 9 bytes
sendToPack(0);              // fill 9 bytes
sendToPack(0);              // fill 9 bytes
sendToPack(0);              // fill 9 bytes
sendToPack(0);              // fill 9 bytes
sendToPack(0);              // fill 9 bytes
```

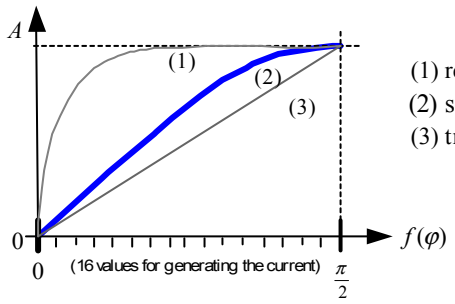
8 INSTRUCTION SET

The instruction code is listed in hexadecimal notation, prefixed with $-sign. *"motnr"* substitutes the number of the motor (0=motornr.1 ... 5=motornr.6). Parameters with more than 1 byte are to be transmitted with the least significant byte (bit 0 – 7) first.

8.1 Adapting the microstep-table to the motor characteristics



alternative motor characteristics (s. CMD $17)



Most motors have varying microstep lengths, due to this the motor would drive discontinuously for a sin -/ cos -current. In order to reach a smoother run, you can drive the motor with an adjusted current, so that the motor's characteristics can be compensated. This current curves are generated with the 16 values in the table set via CMD $17, which describe a quarter period.
(s. left)

8.2 Calculation of the microstep-frequency

$$f_{micro-step} = \frac{f_{clk}}{clkdiv+1} \cdot v_i \Big/ 2^{14+div_i}$$

- full step frequency=1/16 microstep frequency
- f_{clk} is 20MHz
- *clkdiv* is the same for all motors (range 0..31)
- v_i respectively *vakt* is the velocity of each motor (range: –511..+511)
- div_i can be parameterized for each motor (range 0..3)

Note: The microstep frequency must not exceed 200kHz.

8.3 Setting motor parameters

16 bit or 32 bit parameters, marked with "#", are transferred with least significant byte (Bit 0 – 7) first.

⇒ **Peak current**

(settings are done for pairs of motors, i.e. each two motors 0 + 1, 2 + 3 respectively 4 + 5 have the same peak current. Only the values set for motor-numbers 0,2 and 4 are valid.)

CMD	**$10**
P0	*motnr (0...5)*
P1	value (0..255):Sixpack: Imax=0,8A * value/256 Quadpack Imax=1,5A * value/256 * **DIPsetting** (0%, 33%, 66%, 100%)

⇒ **Current control**

Note for **energy saving**: Power consumption can be reduced drastically when power-down current (I0) is set to 0%, i.e. P1=8 for all (unused) motors.

CMD	**$11**
P0	*motnr (0...5)*
P1	*I0*: power down-current (0..8): (0=100%, 75, 50, 38, 25, 19, [13, 9,] 0%)
P2	*I1*: current when motor stands still (s.a.)
P3	*I2*: current for constant velocity (s.a.)
P4	*I3:* current for acceleration (s.a.)
P5,6 #	*T0*(*): (1..65535): power down-delay time in increments of 2 ms (Firmware before v1.40 required even values!)

(*) Note: *T0* is also needed in CMD $15.

⇒ **Velocity setting**

(global setting for all motors, change only with stopped motors!)

CMD	**$12**
P0	*clkdiv* (0..31): s. calculation of step frequency (default: clkdiv=5)

⇒ **Starting velocity**

(change only with stopped motors!)

CMD	**$13**
P0	*motnr (0...5)*
P1,2 #	*vmin* (0..511): velocity used in combination with div_i =3 for searching of the reference-switch. The fastest possible vmin will be calculated automatically when vmin=0 is set. ***vmin* <= 250Hz / fminsteps** ***vmin* <= *vstart*** (fminsteps is the microstep frequency when velocity = 1 and div_i =3 is choosen)
P3,4 #	*vstart* (1..511): starting and ending velocity for acceleration ramp. Limited by firmware to **0 < *vstart* < (512 shr(3-*div*[i]))-1 (e.g. *div*[i]=2 -> 0 < *vstart* < 256)**
P5	div_i (0..3): s. calculation of step frequency (**default: div_i =2**)

⇒ **Velocity, Acceleration**

(*amax* and *vmax* are checked permanentely, the *Pack* promptly reacts to changes)

CMD	**$14**
P0	*motnr (0...5)*
P1,2 #	*amax* (1..32767): max. motor acceleration 1/64 *amax* is accumulated with 500Hz during acceleration to *vact*: Beginning with *vstart* the motor is accelerated until *vmax* has been reached. **0 < *amax* <= *vstart* * 64 (default: *amax* =128)**
P3,4 #	*vmax* (1..511): Maximum value for *vact* ***vmax* <= 511**

Trinamic Motion Control GmbH & Co KG
Sternstraße 67
D – 20357 Hamburg, Germany
Phone +49-40-51 48 06 - 0
FAX: +49-40-51 48 06 - 60
http://www.trinamic.com

⇒ **Motor Parameters**

Note on the reference search algorithm: Usually the reference switch is logically left, i.e. search orientation is in the direction of descending co-ordinates. However if it is defined as logically right (*NullLeft* not set) then the driving range is in the area of negative numbers because zero is the largest number on the position scale.

Note: To exchange left and right physically, only one phase of the motor has to be polarized reversely. The setting *NullCenter* is extremely useful for rotary axes: Here the center of the zero switch is located, i.e. the center between the left and right start of the switch operating point.

For further information read chapter 7.1: Hints for Programming!

CMD	**$15**
P0	*motnr (0...5)*
P1,2,3,4 #	*Poslimit* (0..0x7FFFFFFF): number of µ-steps (note: full-steps *16) per revolution for rotary axes (is used with rotation or way optimization and reference search only), resp. total way for linear axis (with reference search and mechanical reference, then 5/4 of the distance will be driven in direction of the mechanical stop)
P5 #	motor type: • Bit 0: *PIMode*: 0=trapezoidal ramp, 1=*PI-controller* • Bit 1: *Rotary Axis*: 0=linear, 1=*rotary* axis • Bit 2: *AutonullCmd*: flag to start automatic reference search (s. CMD $22) • Bit 3: *TestNull*: set *Stop -bit*(s. CMD $20 P6), (triggering an reference search) if *StopNoRef*(P6, Bit 6)=0) if motor stops at zero point but switch remains inactive after settle time. (Note: Check only when motor is ready!) • Bit 4: *NullLeft*: 1=zero point left of driving range, otherwise right • Bit 5: *NullCenter*: set zero point to the center of the switch's active area • Bit 6: *StopNull*: stops the motor and sets *Stop- bit*, (triggering an automatic reference search, if StopNoRef(P6, Bit 6)=0) if zero switch is active outside the tolerance area around zero point or analogue value exceeds its applying limit (s.b. modifier-bits 12 & 13) • Bit 7: *FilterSwitch*: 1=Zero point switch is de-bounced for 2-30ms (s.b.: mask for switch de-bouncing)
P6 #	• Bit 0: *Way optimization* for *rotary* axis by automatic selection of turning direction • Bit 1: *FastRef*: search for reference with high velocity (s.b.: *vrefmax*) • Bit 2: *MechRef*: use mechanical reference (drives 5/4 * *poslimit* towards zero point) rather than a reference switch • Bit 3: *DelayTestNull*: delay zero point check by *T0*-time (s. option *TestNull*) in order to avoid failures caused by vibration. Initialization T0 s. CMD $11 • Bit 4: *StopSoft*: 1=motor decelerates with *amax* when stop condition (switch or analogue values) is detected rather than stopping abruptly (s. description of *stop-function* via analogue-input s. CMD $31) • Bit 5: *StopNoRef*: 1=no setting of *Autonullcmd* (avoiding automatic reference search) on stop conditions, the Motor just stops instead. • Bit 6: *NullPositive*: 1=zero point switch is high active, i.e. high level at null-point 0=switch is low active • Bit 7: *StopAtFullsteps*: Stop ramps only at the nearest fullstep position

⇒ **Reference Search Parameters**

(change only with motors standing still)

Note: Only relevant for fast reference search!

CMD	**$16**
P0	*motnr (0...5)*
P1,2 #	*Vrefmax* (1..511): fast reference search velocity: **511 >= *vmax* >= *vrefmax* >= *vstart***
P3,4 #	mask for reference switch de-bouncing ($0001=0ms,$0003=2ms,...$FFFF=30ms)
P5 #	Bit0: 1=Stop after reference search, 0=continue actual action after reference search

⇒ **Write motor characteristics Table**

Depending on the position the motors are controlled with discrete analogue current values. The lower 5 bits of the position counter of each motor are used as a pointer into the symmetrical characteristics table and determine the current for coil A of the motor. The current for coil B is determined from the same table by a pointer shifted by 16 steps. For customizing the table can be modified for all motors in common. Therefore only the lower half (16 entries) has to be programmed.

Default-table: 255 * SIN([0.5..15.5]/32*PI)

CMD	**$17**
P0	pointer to the table (start = 0,4,8 or 12)
P1..P4 #	4 table entries (0..255) starting from the given position **(e.g. table=255, 255, ..., 255 -> full step)**

⇒ **Null Point-Offset and -Range**

The null point-offset allows to compensate for the tolerance of the reference switch of linear drives. When used, the reference search will drive further into the null point and the null point is set there. This especially is important, when the zero point check is enabled (*TestNull*, s.a.). If at the same time testing for premature interrupt of reference switch is enabled (*StopNull*, s.o.) a small area around the zero point can be excepted from the test via the parameter *testnullrange*. Outside this area the reference switch triggers an emergency stop and reference search, when the motor is driving into the direction of the reference point. The offset also can be used to shift the null-point farther to the middle of a linear axis.

CMD	**$18**
P0	*motnr (0...5)*
P1,2,3,4 #	*nulloffset* (signed long): distance between zero point and reference switch
P5,6 #	*testnullrange* (0..65535): range where zero switch may be active

⇒ **PI-Parameter**

The PI-controller allows to generate a velocity profile by continuously giving new target positions via the host computer. The factors for the integral and proportional part determine the feedback-control characteristics. The controller works at 500Hz. The proportional part is 64 * position difference / *propdiv*. The integral part integrates the part of the position difference clipped to a maximum limit by *intinpclip*. The influence of the integral part is determined by the divisor *intdiv*.

CMD	**$19**
P0	*motnr (0...5)*
P1	*propdiv* (1..255)
P2,3 #	*intdiv* (1..32767)
P4,5 #	*intclip* (1..32767)
P6	*intinpclip* (0..255)

8.4 Driving Ramps

⇒ **Query Position and Activity**

CMD	**$20**
P0	*motnr (0...5)*
P1	response address

response	
CMD	$20
P0	*motnr (0...5)*
P1,2,3,4 #	*posact* (signed long): current position
P5	Current action (0: inactive, 5: ramp, 10: PI-controller, 15: rotation, 20 – 29: reference switch search, 30: mechanical reference)
P6	bit 0: *stop-status*. 1=Stop-condition has occurred. Flag is cleared after read.

⇒ **Query Velocity and Activity**

CMD	**$21**
P0	*motnr (0...5)*
P1	Response address

Response	
CMD	$21
P0	*motnr (0...5)*
P1,2 #	*Vact* (integer): actual velocity
P3	Current action(0: inactive, 5: ramp, 10: PI-controller, 15: rotation, 20 – 29: reference switch search, 30: mechanical reference)

⇒ **Start search of Reference**

First the motor is stopped. The motor optionally drives fast (vrefmax), searching for the position of the switch. When the switch is found, the motor is driven back to the point, where the switch becomes inactive. Then it is slowly driven with vmin towards the switch to find the exact position. If the switch cannot be found again at slow speed where it had been found before, or if no switch is seen during 125% of the drive limit-range, the whole procedure repeats by first stopping again, which may give another chance to hold grip for an axis out of control. After the reference point has been identified via the reference point switch the position is set to null respectively to null-offset and the motor resumes its previous operation e.g. by driving a to the actual position *posact*, where the reference search started, if CMD $16 P5, Bit0=0 (s. CMD $16).
($22 and CMD $15, P5, Bit2 start the same action!)

CMD	**$22**
P0	*motnr (0...5)*

⇒ **Start trapezoidal Ramp**

The motor drives from its current position to the target position. The command does not influence the motor, if the motor is still active. To change the target position at any time (on-the-fly) use command $26, and follow it by command $23 with the same target position, to ensure that the motor also starts if it stood still before. The motor will use the optimum way to the target, while considering the motion parameters as well as the current velocity.
If rotation has been active, the motor is only stopped by this command. The distance of any ramp must be within 32 bit signed range too– which is no restriction as long as you consider 0 as one limit of your driving range.
note: For circular motors the position cant be both positive and negative.
(null-left-motors (s. CMD $15): 0 <= position < poslimit)
(non null-left-motors : 0 >= position > -poslimit)

CMD	**$23**
P0	*motnr (0...5)*
P1,2,3,4 #	target position (32 bit signed long)

⇒ **Activate PI-Controller on Target Position (on-the-fly target position change)**

The motor position will be controlled by the PI-Controller so that the target position is reached. Switching to PI-controller happens immediately, even if the motor is inactive. When the PI-controller was already active the target position will be reprogrammed immediately. The maximum distance must be less than 32 bit signed range too – which is no restriction as long as you consider 0 as one limit of your driving range (which however is not mandatory).

CMD	**$24**
P0	*motnr (0...5)*
P1,2,3,4 #	Target position (32 bit signed long)

⇒ **Start Rotation / change Rotation Velocity**

With this command it is possible to rotate an axis with the given velocity. The maximum acceleration is obeyed when the velocity is to be changed. The change to rotation happens immediately, even if the motor is still active. A check of the reference switch can not take place with too fast rotation.
Cyclic motors will wrap around at the end of their position ranges when rotating. (s. CMDs $15, $16)

CMD	**$25**
P0	*motnr (0...5)*
P1,2 #	Rotation velocity (-511..511)

⇒ **Set Target Position (on-the-fly target position change)**

Modifies the target position without influencing the operation mode. Can be used to shorten or lengthen a ramp. If a ramp has to overshoot due to late changing its target position the peak will de-celerate with amax instantly and continue driving a reverse ramp on its own.

CMD	**$26**
P0	*motnr (0...5)*
P1,2,3,4 #	target position (32 Bit signed long)

⇒ **Set actual Position**

Forces the internal position counter to any position. Can lead to unintended reference searches at detection of zero switches.

CMD	**$27**
P0	*motnr (0...5)*
P1,2,3,4 #	*posact*: new actual position (32 bit signed long)

⇒ **Query all Motor Activities, Request delayed Response**

Each axis can be queried for activity with this command. When delayed response is requested, the *PACK* will send the response as soon as all concerned motors are inactive. The response contains the actual action of all motors.

Attention! In RS 485 mode with multiple slaves this command can lead to bus collisions! To avoid this, the bus should not be used for other transactions while waiting for response.

CMD	**$28**
P0	response address
P1	mask for delayed response (Bit 0=Motor 0, Bit 5=Motor 5) (0: motor masked, 1: respond only after motor has become inactive)

response	
CMD	$28
P0..P5 #	*current action* (0..5) (0: inactive, 5: ramp, 10: PI-controller, 15: rotation, 20 - 29: reference search, 30: mechanical calibration)

⇒ **Start trapezoidal Ramp in parallel**

This command allows a coordinated start movement, by starting multiple motors at the same time. The target position has to be programmed previously (s. CMD $26).

CMD	**$29**
P0	mask for ramp (bit 0=motor 0, bit 5=motor 5) (0: motor masked, 1: start motor)

⇒ **Stop Motors selectively or synchronously**

Multiple motors can be stopped at the same time via this command. It sets the target position of each motor concerned equal to its actual position. However motors driving beyond vstart will overshoot and return driving another ramp back to the point you set as their new target using this command.

CMD	**$2A**
P0	Mask for deceleration (Bit 0=Motor 0, Bit 5=Motor 5) (0: Motor masked, 1: set target position to actual position)

8.5 Additional Inputs / Outputs

⇒ **Read Motor Input Channels and additional Inputs**

The analogue channels are prepared for ratiometric measurements of resistive dividers. Channel 6 is the external input, channel 7 measures the voltage supply of the *PACK* (1V equals value 22). The reference inputs are inverted.

CMD	**$30**
P0	*channel no (0=channel0 ... 7=channel7).*
P1	response address

Response	
CMD	$30
P0	*channel no (0...7).*
P1,2 #	*analogue value* (unsigned 0..1023)
P3	reference input (Bit 0)
P4	all reference inputs / jumpers (bit 0 = motor 0, ..., bit 6=jumper1, bit 7 = jumper2) (s. setting the baud rate)
P5	bit 0: logic state at TTLIO1

⇒ **Setting the Limits for the Stop Function left/right**

A potentiometer or a resistor network connected to two stop switches at the analogue input of each motor can trigger a hard or soft stop. The voltage of the analogue input should increase in right direction (cw). The measured value is checked dependent on the motor direction. When a stop-condition occurs the motor is stopped immediately. If *StopSoft*-Flag is set, the motor is in decelerated with the pre-set acceleration. If the *StopSoft* Flag isn't set the motor will be stopped abruptly, so that the precise motor position may be lost. Therefore a reference search will be started additionally if the *StopNoRef*-Flag is not set. When the *StopNull*-Flag is set, the zero switch also functions as a limit switch. If the reference switch is defined on the right side, the motor then can only drive in the area of negative co-ordinates (***position* <= 0**) (s. CMD $15)

CMD	**$31**
P0	*channel no (0...7):*
P1,2 #	*analogue value* minimum for stop left (0=no function)
P3,4 #	*analogue value* maximum for stop right (≥1023=no function)

⇒ **Setting of additional Outputs**

CMD	**$32**
P0	bit 0: set logic level at TTLOUT1 (1=READY, 0= active)
P1	bit 0: output enable for TTLIO1 (1=input, 0=output)
P2	bit 0: set logic level at TTLIO1
P3	bit 0: 1=TTLOUT1 works as ready output (1=READY, 0= active)

⇒ **Function of the ready Output**

The ready output can be activated (low, open collector), whenever a motor is active (velocity greater than 0) or search of reference. The ready output will be switched within 2 ms after start/end motor movement. The repeatability (jitter) equals approximately the microstep rate during start respectively stop (s. CMD $13).

CMD	**$33**
P0	mask for active motors (bit 0=motor 0, bit 5=motor 5)
P1	mask for reference search of a motor (bit 0=motor 0, bit 5=motor 5)

8.6 Other Settings

⇒ **Adjust RS 232 / RS 485 Baud rate**

CMD	**$40**
P0,P1 #	baud rate divisor (16 bit): baud rate divisor=20MHz/ (16*baud rate)
P2,P3 #	transmitter switch on/off delay time in increments of 2ms (1..1000) **(default: 6ms)**

⇒ **Setting of Timeout for Abort of Packet (RS 232 / RS 485)**

CMD	**$41**
P0,P1 #	timeout in increments of 2ms (2..65535)

⇒ **Change Address of Unit (RS 232 / RS 485)**

CMD	**$42**
P0	new RS 232 / RS 485 address

⇒ **Read out Information about Unit**

Allows to read out of firmware revision, reset-flag, temperature and serial number.

CMD	**$43**
P0	response address

response	
CMD	$43
P0	*firmware-revision* (e.g. 148=V1.4.8)
P1	*reset flag*: during first read out 1, afterwards 0
P2	temperature of PACK in °C (8 bit signed)
P3,P4 #	serial number

⇒ **Configure /query RS 232-operating Mode via CAN**

CMD	**$44**
P0	response address
P1	response address for RS 232-receiving via CAN (upper 8 bit)
P2	number of bytes which should be forwarded with RS 232-receiving (1..8, 0=disabled)

Response	
CMD	$44
P0	number of bytes in the RS 232-send buffer (0=empty)
P1	bit 0: State of the CTS line (inverted)

⇒ **Power-Down modus**

(Version 1.46 and since Version 1.49)
These versions store the actual motor positions as soon as the power supply goes below 13V, if enabled. The motors are stopped, as soon as undervoltage is detected. Thus the devices should not be operated below 15V under normal conditions.

The power down state is independent of bit 2 as well observed as set back with bit 3. The power down state which is read also is independent of the stop & save-activation set via bit 2.

CMD	**$45**
P0	response address
P1	• Bit 0: 1 = Read positions from EEprom, and copy them to the motor position registers, if they contain valid values • Bit 1: 1 = set positions in EEprom as invalid • Bit 2: 1 = activate Power down (stop motors & autosave position on undervoltage) • Bit 3: 1 = Reset powerdown state and re-enable the motors (only possible while the supply is above the power down level)

Response	
CMD	$45
P0	• Bit 0: 1 = loaded valid position from EEprom • Bit 1: *1 = located valid position in EEprom* • Bit 2: *1 = Power down – status before Command* • Bit 3: *1 = Power down – status after Command*

⇒ **Complete Hardware Reset**

CMD	**$CC**
P0	HW-Reset

Attention: All configured parameters will be replaced by the default parameters
This command needs about 1 second.
Note: The RS232 usually receives a 0-byte during execution.

8.7 Multi-dimensional Movement

⇒ **Start multi-dimensional linear Interpolation**

Coordinated movement with multiple motors to a target position. The motors have to stand still before execution. All motors reach the target position at the same time. The target position has to be set for each motor (with command $26) before. The axis, which, regarding its position distance, is the fastest, will be automatically used as a master. In regard of its distance this axis will be driven with the lowest acceleration. To determine if the target position has been reached, all involved motors have to be queried.

CMD	**$50**
P0	mask for motors (bit 0=motor 0, bit 5=motor 5) (0: motor unused, 1: motor used in multi-dimension motion)

8.8 Service-Functions

These functions are not intended for the user and when used improperly the unit can be damaged permanently.

⇒ **Enable erasing and writing the Flash-Memory**

CMD	**$F2**
P0	address for response
P1,2,3,4 #	magic code

Response	
CMD	$F2
P0	1=erase OK

⇒ **Program Flash Memory:**

This function can only be used after erasing. It should not be interrupted by any other function until the flash memory is fully programmed

CMD	**$F3**
P0-P6 #	7 data bytes

⇒ **Query flash Memory Check-Sum and abort Programming if necessary**

CMD	**$F4**
P0	address for response

response	
CMD	$F4
P0,P1 #	*check-sum* (value depends on SW -version)

⇒ **Read out Flash-Memory**

Query the check sum to set the auto-incrementing address pointer to zero, before using this function the first time.

CMD	**$F5**
P0	address for response

response	
CMD	$F5
P0-P6 #	7 data bytes read from the memory

⇒ **Write EEPROM**

CMD	**$F6**
P0	address for response
P1,2	Magic Code 2
P3,P4	address
P5,P6	value

response	
CMD	$F6
P0	1=writing successful

⇒ **Read EEPROM**

CMD	**$F7**
P0	address for response
P1,2	*Magic Code* 2
P3,P4	address
P5,P6	value

response	
CMD	$F7
P0	1=writing successful

Trinamic Motion Control GmbH & Co KG
Sternstraße 67
D – 20357 Hamburg, Germany
Phone +49-40-51 48 06 - 0
FAX: +49-40-51 48 06 - 60
http://www.trinamic.com

9 Instruction table