SIXpack 2

Manual

6-Axis stepper motor controller / driver module
1.4A RMS (2A peak) / 48V
with CAN, RS485 and RS232 interface

TRINAMIC
MOTION CONTROL

GmbH & Co KG
Sternstraße 67
D - 20357 Hamburg, Germany
http://www.trinamic.com

Table of Contents:

1 Life support policy

TRINAMIC Motion Control GmbH & Co. KG does not authorize or warrant any of its products for use in life support systems, without the specific written consent of TRINAMIC Motion Control GmbH & Co. KG.

Life support systems are equipment intended to support or sustain life, and whose failure to perform, when properly used in accordance with instructions provided, can be reasonably expected to result in personal injury or death.

2 Introduction

2.1 Brief Description

The SIXpack 2 is a highly integrated stepper motor controller for six 2-phase stepper motors with a coil current of up to 2000 mA each. It is the fully compatible advancement of the QUADpack and SIXpack with enhanced ratings and functions.
The SIXpack 2 comes in a rugged box package with tested electromagnetic compatibility. It is easy to use and the ideal solution for stepper motor control in an industrial environment.
A DSP supported by special hardware allows a powerful function set and a wide stepping frequency range for all motors. The SIXpack 2 is equipped with RS 232, RS 485 and CAN-Interface.

2.2 What do I have to know about my application

Before the SIXpack 2 is set up the demands of the application to the stepper motors should be clearly defined.
What are the right motors? The SIXpack 2 supports 2-phase stepper motors. The peak motor power is defined by the modules supply voltage and the coil current of the driver ICs (max. 2000mA).
The actual motor position is not known after power on. Thus a reference search has to be made in order to find the absolute position. Most applications use switches or photo interrupters to detect the reference position. In some applications a mechanical limit point without any detector switch is appropriate. The SIXpack 2 does not support encoders.
To control the SIXpack 2 there are different interfaces (RS232, RS485, CAN). It is not relevant to the SIXpack 2 if it is controlled by a desktop-PC, another machine operating unit or a set up with microcontroller. A stand alone operation is not possible.

2.2.1 The SIXpack 2 does not support

Stand alone operation:	The SIXpack 2 can not operate without initialization and control by a host connected to one of its serial interfaces.
Closed loop operation:	The SIXpack 2 has no encoder interface and there is no other possibility for nominal/actual value comparison for motor position. It is possible to compare the internal position counter with the status of the reference switch. If an invalid value is detected an automatic reference search can be started automatically.
External output stages:	There is no possibility to connect external power stages to the SIXpack 2 to increase the maximum motor rating above 2A / 48V.

2.3 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:	9 byte control, 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, Decimal point indicates reference search); 1 Ready Output (Open Collector)
power supply:	15 to 48 V DC (absolute max. 58V) max. ca. 12A, depending on motor type
motor current SIXpack2:	software configurable ca. 100 - 2000 mA per channel (peak coil current); constant current (chopper, ca. 36 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 For detailed housing dimensions see "SIXPACK_mechanical" document on TechLibCD or www.trinamic.com
weight:	927g
material:	housing: steel with red powder coating circuit board: UL certified, leadfree and RoHS compliant

Feature	SIXpack 2 data	Advantages of new feature
Supply voltage (nominal)	15 .. 48 VDC	High motor dynamics / higher torque (up to 3x compared to SIXpack)
Supply voltage (maximum limit)	13 .. 58 VDC	
Motor current	Software setting 0.2A to 1.4A RMS	
Numbers of motors	6	
Chopper scheme	Slow Decay / Mixed Decay	Reduced motor resonances at medium velocities and improved microstep exactness when using Mixed Decay
Motor drivers	TMC239A-LA	Very low power dissipation and high reliability
Software	Fully compatible to SIX/QUADpack	
Control LEDs	Data, Reset, +5V	
Reset button	Yes	
Protection	Varistor protection for all motors	Life plug protection up to some degree
Hot plugging / emergency off	Additional motor off-jumper	Not visible by software – connect to TTLIO1 if software readout is desired

2.4 Electrical data

The operational ratings show the intended / the characteristic range for the values and should be used as design values. In no case shall the maximum values be exceeded.

Symbol	Parameter	Min	Typ	Max	Unit
V_S	Power supply voltage	13	15 … 48	58	V
I_S	Supply current		<<Sum(Imot)	10	A
I_{COIL}	Motor coil current for sine wave peak (chopper regulated, adjustable via software)	0	0.3 … 2.0	2.0	A
I_{MC}	Nominal RMS motor current	0	0.2 … 1.4	1.4	A
f_{CHOP}	Motor chopper frequency		36.8		kHz
t_{SLP}	Coil output slope		300		ns
I_S	Power supply current		<< I_{COIL}	1.4 * I_{COIL}	A
I_{MC}	Continuous motor current (**RMS**)	0	0.3… 1.1	1.1	A
U_{+5V}	Internally generated +5V voltage	4.8	5	5.2	V
I_{+5V}	Load on +5V output (sum of all outputs)			100	mA
R_{+5VMC}	Internal protection resistor on +5V for motor outputs		33		Ohm
U_{AIN}	Analog input voltage range (ratiometric measurement referred to +5V output)		0 … U_{+5V}		V
$U_{REFINLO}$	Reference input low voltage	0		1.5	V
$U_{REFINHI}$	Reference input high voltage (internal 10K pullup to +5V)	3.5		5	V
U_{INPROT}	Input protection range for REFIN and AIN (not for normal use)	-24		24	V
I_{READY}	Output current ready output (Open collector with internal 1K pullup to +5V)			-100	mA
T_{ENV}	Environment temperature (operating / full current)	-25		60	°C
T_{ENV}	Environment temperature with reduced current or forced cooling of board			85	°C

Table 2.1: Operational Ratings

3 System Start Up

3.1 System Start Up / Notes

When the SIXpack 2 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 SIXpack 2 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. This condition should not occur in a normal operation condition. If the SIXpack 2 reports a board temperature above or near 85°C, a forced air cooling or other means to reduce heat dissipation, like reduction of stand by motor current is proposed to ensure a long product life time.
When an application requires detection of temporarily interrupted power supply of the SIXpack 2, this can be done for example by signaling via external TTLOUT1 by programming it to a negative level. It will return to a high level after a reset. The RS 232-interface usually receives a 0-byte after hardware reset.

3.2 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 increases the required motor current. Therefore you should choose the motor with the lowest inductance possible which delivers the required torque at a coil current of approx. 1000 to 1400 mA. Highest possible operating voltage of the SIXpack 2 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 chirping noise can occur in the coils.

3.3 Length of Wires

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

3.4 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 in the housing the protection 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.

3.5 Improvement of the EMC-Conduction

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

3.6 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.

4 Replacing QUADpack or SIXpack

The SIXpack 2 is fully compatible to the QUADpack and SIXpack. Adaptation is provided by DIP-switches.

4.1 DIP-switch marking comments

Mx_I0, Mx_I1:	Inputs for motor current setting. The index x specifies the motor number
2A_135:	Enable of 2A motor current for motor 1, 3 and 5.
2A_246:	Enable of 2A motor current for motor 2, 4 and 6.
MD_135:	Disable of mixed decay for motor 1, 3 and 5.
MD_246:	Disable of mixed decay for motor 2, 4 and 6.

4.2 SIXpack compatible setting

Motor current: 0.8 A

Chopper scheme: "Slow decay"
To use "mixed decay" for all motors switch 'MD_135' and 'MD_246' off.

4.3 QUADpack compatible setting

Motor current: 0.5 A

Chopper scheme: "Slow decay"
To use "mixed decay" for all motors switch 'MD_135' and 'MD_246' off.

Motor current: 1.0 A

Chopper scheme: "Slow decay"
To use "mixed decay" for all motors switch 'MD_135' and 'MD_246' off.

Motor current: 1.5 A

Chopper scheme: "Slow decay"
To use "mixed decay" for all motors switch 'MD_135' and 'MD_246' off.

5 Fundamental Functions – First Steps

This part of the documentation describes the use of SIXpack 2 via an example with limited functionality. For additional functionality please refer to next chapter. In addition to the required steps in hardware set up you should take into the operation the delivered Windows™-Software.

5.1 Warning Advice

To avoid damage to the SIXpack 2 please be aware that:

- Wrong connector pin assignment may destroy the SIXpack 2. Be extremely accurate at the installation and when confectioning cables.
- Disconnecting the Motor while operational may destroy the SIXpack 2. Disable the motor current by pulling the jumper, or better power down the device before connecting / disconnecting motors.

5.2 Basic Device Settings

5.2.1 SIXpack 2 Address

Before the SIXpack 2 is put into operation some adjustments of the device have to be checked, eminently the address of the serial interface. It is set via the CANHI and CANLO switches and should be set to zero by default. Refer to Figure 5.1.

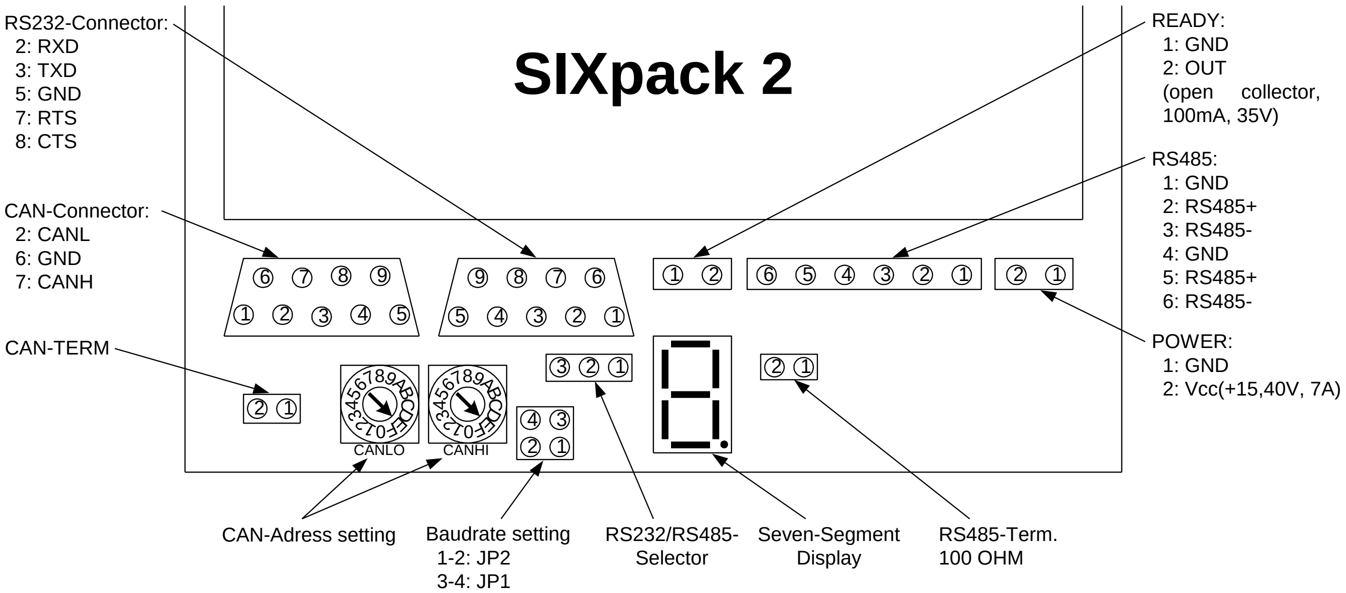


Figure 5.1: Jumper and connectors

5.2.2 Option RS232/RS485

To operate the SIXpack 2 via its RS232 interface, place the jumper RS232/RS485 at position RS232 (Refer Figure 5.1). The use of RS485 is described in chapter 6.1.1.

5.2.3 Baudrate of serial interface

The baud rate of the serial interface is set via jumper JP1 and JP2 (refer Figure 5.1).

JP1	JP2	Baud rate RS232/RS485	Baud rate CAN
X	X	9600	1 Mbit/s (*)
-	X	57600	500 kbit/s (*)
X	-	38400	125 kbit/s (*)
-	-	19200	250 kbit/s (default)(*)

Table 5.1: Adjustment of baudrate with jumpers

(*): The SIXpack 2 has an internal buffer for 16 CAN commands which need about 2ms execution time each. This might limit the maximum data throughput.

The command *GetInputValues* (SQPack-Tab I/O, $30) provides the actual jumper configuration in the variable `AllInputs`.
Bit 6 = Jumper1, Bit 7 = Jumper2

The baud rate for RS232/485 can be modified by software also. This setting is not stored permanently.
In order to get the actual jumper-configurations send CMD $30.

5.2.4 Termination of CAN/RS485

Each interface can be terminated by setting the jumpers "TERM CAN" and "TERM RS485" (refer Figure 5.1).

5.2.5 Seven-segment display

The seven segment display shows the number of active motors. With an appropriate power supply a "0" is shown at start. The decimal point indicates that a reference search at any motor is accomplished.
If a malfunction occurs the display shows "8", "C" or "F". Try to restart the SIXpack 2.

5.2.6 Driver enable

The jumper "Driver enable" (close to the motor connectors) enables (jumper set) or disables (jumper open) the drivers for all motors. Pin2 of the jumper is GND, and Pin1 has a 10k pullup to +5V with a CMOS logic gate level. It can be used as an interlock/motor enable switched by opto coupler or relay. If the drivers are disabled, i.e. jumper open, it is safe to disconnect the motors while power on and retain the actual settings of the SIXpack 2.
Hint: It is possible to use a switch to enable/disable the drivers.

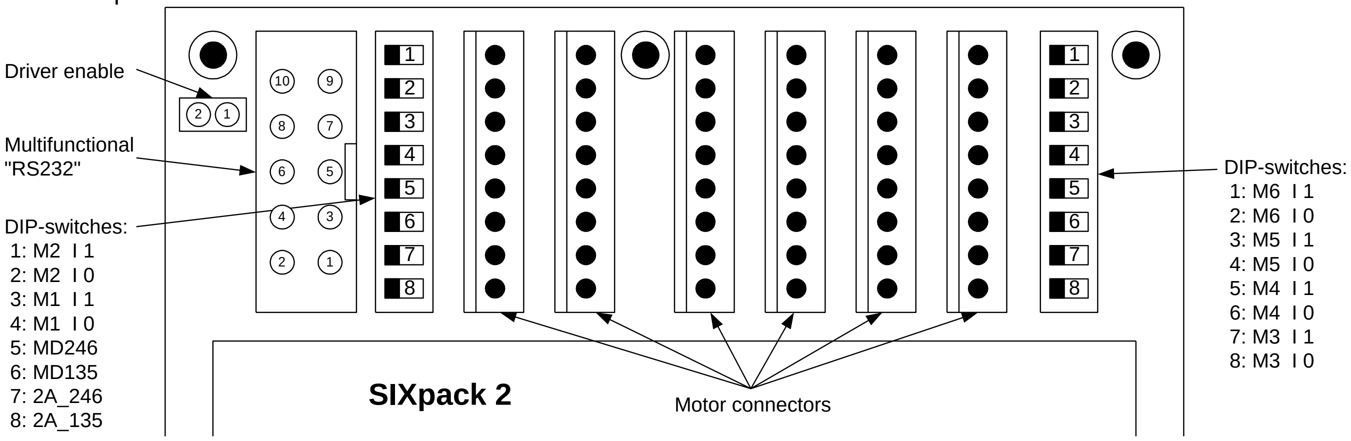


Figure 5.2: Driver enable, DIP switches and motor connectors

5.2.7 Adjusting the maximum current

The maximum current of the SIXpack 2 can be set via DIP-switches. The switches are close to the motor connectors. This adjustment allows a coarse adaptation of the current setting to the motors' maximum current. Additionally it provides a compatibility setting for the former QUADpack and SIXpack, which had different maximum currents. With the appropriate DIP-switch configuration the SIXpack 2 is fully software compatible. For fine current adjustment please see 5.6.1 and 6.2.1.2.

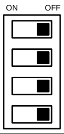
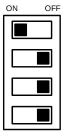
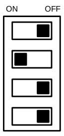
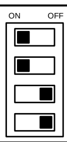
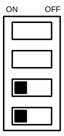
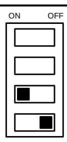
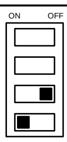
Current	DIP-switch position	description
0.5 A	Mx_I0 Mx_I1 2A_135 2A_246 	Current setting for each motor (index x specifies the motor number)
0.8 A	Mx_I0 Mx_I1 2A_135 2A_246 	Current setting for each motor (index x specifies the motor number)
1.0 A	Mx_I0 Mx_I1 2A_135 2A_246 	Current setting for each motor (index x specifies the motor number)
1.5 A	Mx_I0 Mx_I1 2A_135 2A_246 	Current setting for each motor (index x specifies the motor number)
2.0 A	Mx_I0 Mx_I1 2A_135 2A_246 	2.0 A maximum current for all motors
	Mx_I0 Mx_I1 2A_135 2A_246 	2.0 A maximum current for motor 1, 3 and 5. Current setting for motor 2, 4 and 6 via 'Mx_I0' and 'Mx_I1'.
	Mx_I0 Mx_I1 2A_135 2A_246 	2.0 A maximum current for motor 2, 4 and 6. Current setting for motor 1, 3 and 5 via 'Mx_I0' and 'Mx_I1'.

Table 5.2: Adjusting maximum current

5.2.8 Setting the chopper mode

The SIXpack 2 supports "Mixed Decay" which provides reduced motor resonance at medium velocities and improved microstep exactness. This mode can be switched off by the DIP-switches if desired.

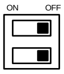
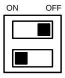
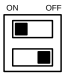
Chopper Scheme	DIP-switch position	description
Mixed decay (all motors)	 ON OFF MD_135 MD_246	Mixed decay for all motors
Mixed decay (motor 1, 3 and 5)	 ON OFF MD_135 MD_246	Mixed decay for motor 1, 3 and 5
Mixed decay (motor 2, 4 and 6)	 ON OFF MD_135 MD_246	Mixed decay for motor 2, 4 and 6

Table 5.3: Adjusting chopper mode

5.3 Connections

Initial operation of the SIXpack 2 is possible after installing current supply and serial interface.

5.3.1 Current supply

Any power supply unit with an output voltage of 15-48V may be used. The required current is conform to the usage and quantity of motors. The connector is labeled POWER. Be aware that the polarity is correct and make sure that your supply voltage never exceeds the absolute maximum rating.
With an appropriate current supply the LEDs "+5V" and "+24V" light up and the 7-segment-display shows zero. If the display shows "8", "C" or "F", or the unit resets continuously, an internal defect has been detected. When continuous resets occur, the Flash memory could be erased. You can try to swap flash with an other device.

5.3.2 Serial interface

The RS232 interface is connected via a null modem cable with one serial interface of the PC.

5.3.3 Motor connectors

CAUTION: Connecting or disconnecting while power on may damage the motor drivers of the SIXpack 2.

The function "Driver enable" (refer to 5.2.6) provides the possibility to disconnect motors while the modules power is on. Thereby all actual settings are retained.

The motors are connected with the 8 pin connectors. The pinning of the connectors are as follows:

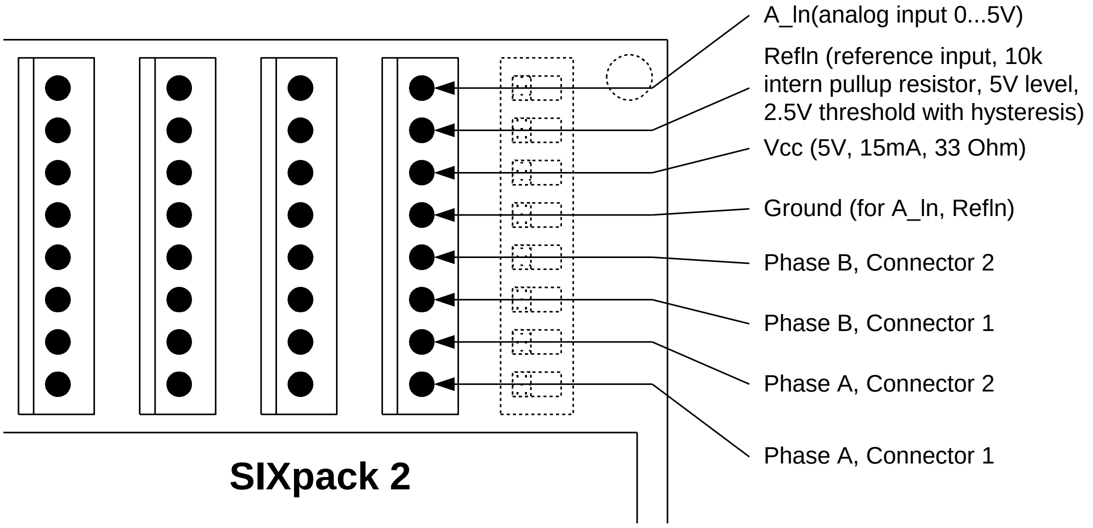


Figure 5.3: Pinning of motor connector

For fundamental functions the pin connections for the motor phases are important. Connect the motor coils indicated in Figure 5.4 with the connector of the pack.

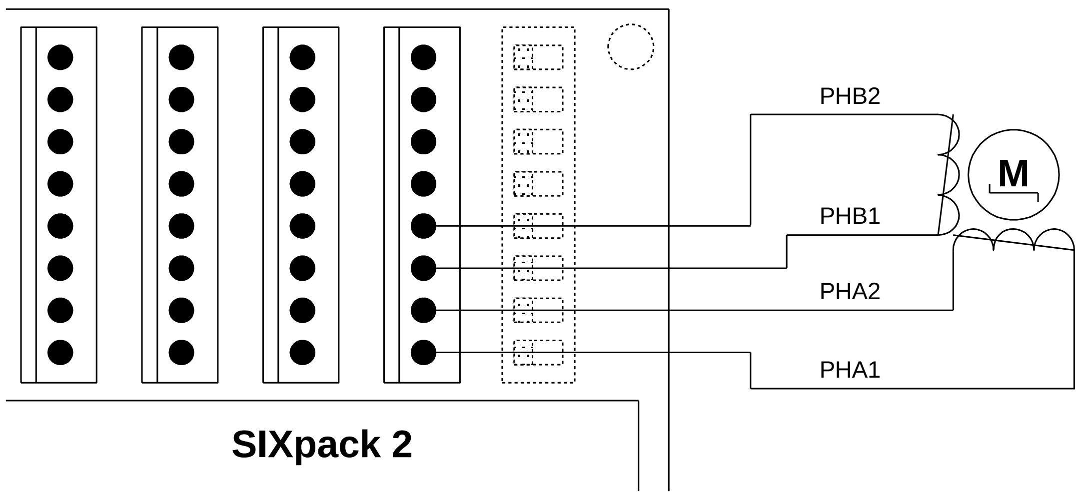


Figure 5.4: Connecting motor coils

5.3.4 Connector specifications

Motor connector:	Producer:	AMP Connectors (www.amp.com)
	Item number:	0-0770602-8 (case)
		0-0770666-1 (crimp contacts)
		0-0058517-1 (crimping tool)
Power supply/RS485:	Producer:	Weidmüller (www.weidmueller.com)
	Item number:	1716320000 (2-pole, VDC, Ready)
		1716360000 (6-pole, RS485)

5.4 Start-up with software SQPack

After adjusting the necessary hardware of the SIXpack 2 the first function tests with the Windows™ program SQPack can be done.

5.4.1 Installation

The installation of the program SQPack is accomplished by copying the file "SQPack.exe" to any location on your PCs hard disc. At first start the file "SQPack.ini" is created to save actual settings. To reset the settings of the software simply remove this file.

5.4.2 Initiation

To initiate the program double click "SQPack.exe". A window with nine tabs is opened. In front is the tab "Connection". To establish a connection to the SIXpack 2 choose the correct COM-Port. The other settings should be at necessary values.

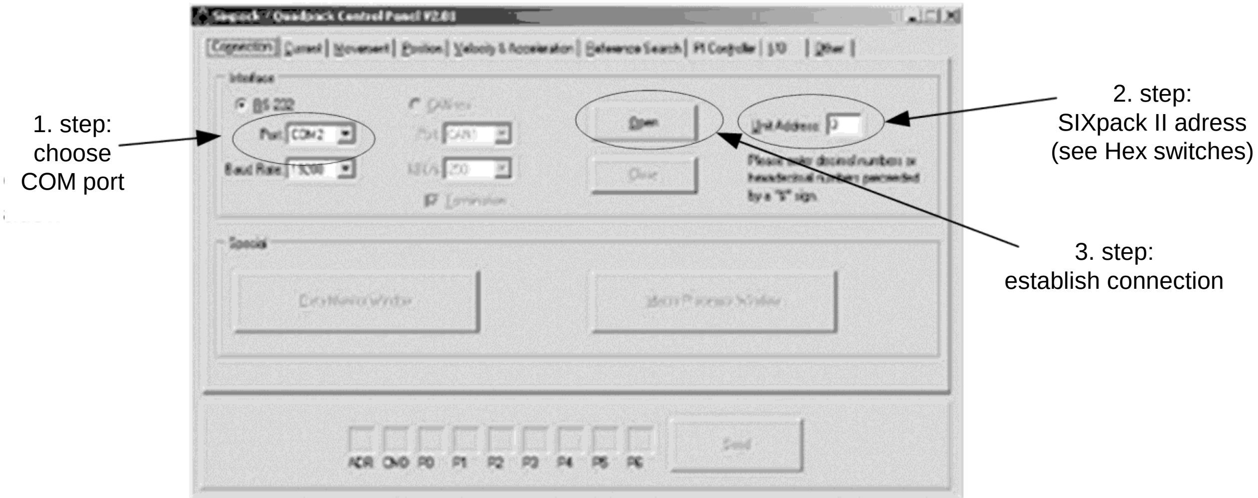


Figure 5.5: Setup of SQPack

5.4.3 Functional test: Get system information of SIXpack 2

To check the connection the command *GetUnitInformation* is useful. Choose the tab "Others" and activate in box "Commands" the line "$43: Get Unit Information". If the connection settings are done correctly it is possible to click the button "Send" at the bottom of the window to transfer the command to the SIXpack 2. If a connection is established the values for "Firmware Revision", "Reset Flag", "Temperature" and "S/N" are displayed in the box "Parameters".

5.4.4 "First steps": Movement of motor

The significant advantage of stepper motors is to turn to a specific position. The motor is accelerated by the SIXpack 2 to an assigned velocity and decelerated in time to reach the specified position.
Therefore choose on the tab "Movement" the command "Ramp". The variable "Position" represents the exact target position. By clicking the "Send" button the motor turns to the specified position.

5.4.5 Concept of SIXpack 2 interface protocol

The SIXpack 2 is controlled by frames of 9 byte generally. The software SQPack displays this frames in the lower part of the window next to the "Send" button. The nine bytes are shown hexadecimal.
At start it is easiest to create the frames by software and add these eventually to an own program.

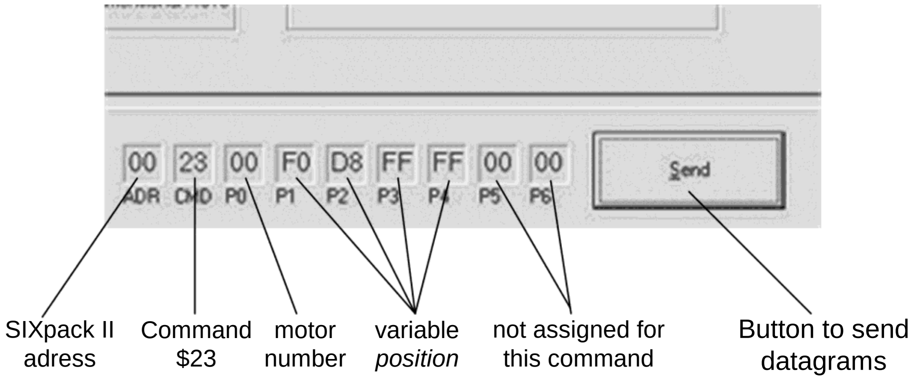


Figure 5.6: Command frames of "SQPack"

5.4.6 Macro functions of SQPack

At the tab "Connection" the software provides the function "Macro Processor Window" to store command sequences to a file. This file can be opened by an ASCII-Editor and the command sequence is formatted as follows:

SendToPack (adr, cmd, p0, p1, p2, p3, p4, p5, p6)

"SendToPack" stands for the function which sends nine bytes to the SIXpack 2 via the serial interface. The parameters are hexadecimal.

A Macro is generated by clicking the button "Record Macro" on the tab "Macro Processing Window". A dialog box is opened to name the file in which the macro is saved. Afterwards any commands can be saved to the file until the recording is ended by clicking "EndRecording".
CAUTION: Some commands are excepted at stopped motor only. According to this refer to the command description.

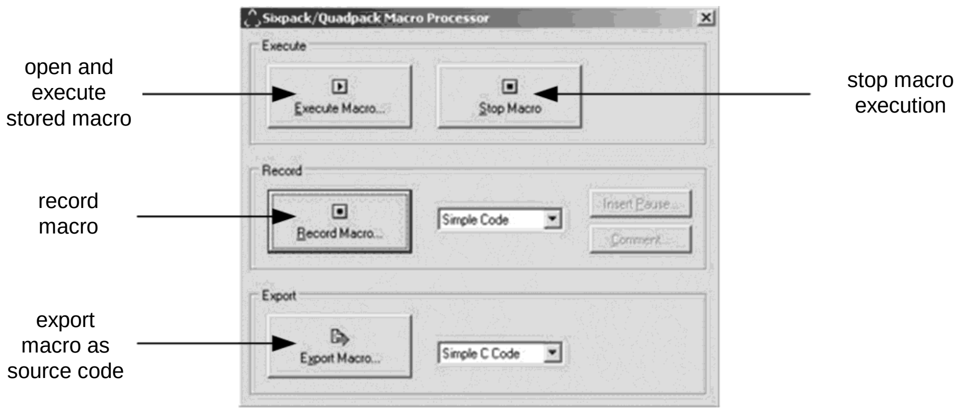


Figure 5.7: Macro function of SQPack

5.5 Operation with reference/ending points

Most applications with stepper motors need a reference point to acquire the actual position of the motor. The reference point of each axis is detected by the SIXpack 2 with a reference search. The positions scaling determined by type of the application. It is a **linear axis** if the motor moves a mechanic with defined end and start point. Every point on the axis is assigned to a position between start and end point.
If there is no end or start point it is a **rotating axis**. The position values for the axis are defined for one rotation. The value starts at zero again after full revolution.

5.5.1 Types of reference point definitions

The SIXpack 2 provides two completely different types to define the reference point:

- **Mechanical stopper for reference point (linear axis only)**
  At start point or end point a stopper for the mechanic of the axis is build in. At reference search the SIXpack 2 moves the respective axis in direction of the stopper (defined by the flag MT_NULLLEFT, SQPack-tab: "Reference search", CMD $15) for a distance of `Poslimit` * 5/4. The multiplication with 5/4 causes the SIXpack 2 to reach the stopper before end of the reference search. The motor is mechanically stopped, reference point is reached.

- **Electrical reference point**
  At any point of the positioning scale of the axis an electrical switch is implemented which is activated by the passing motor. The SIXpack 2 catches this for reference point.
  A reference switch attached one end of a linear axis can be used as end switch. At activation of the switch the motor stops.

5.5.2 Hardware installation

5.5.2.1 Connection of 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.

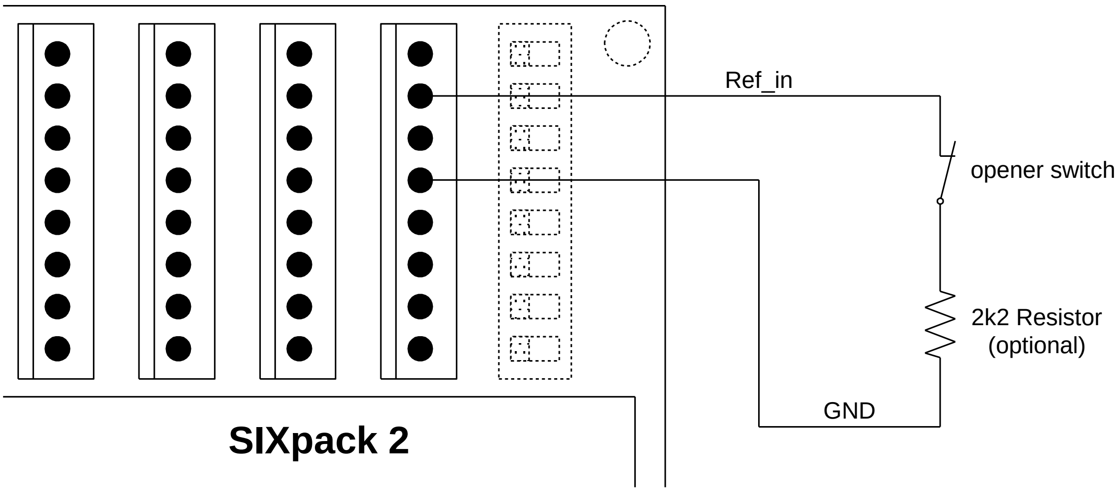


Figure 5.8: Connecting reference switch

5.5.2.2 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.

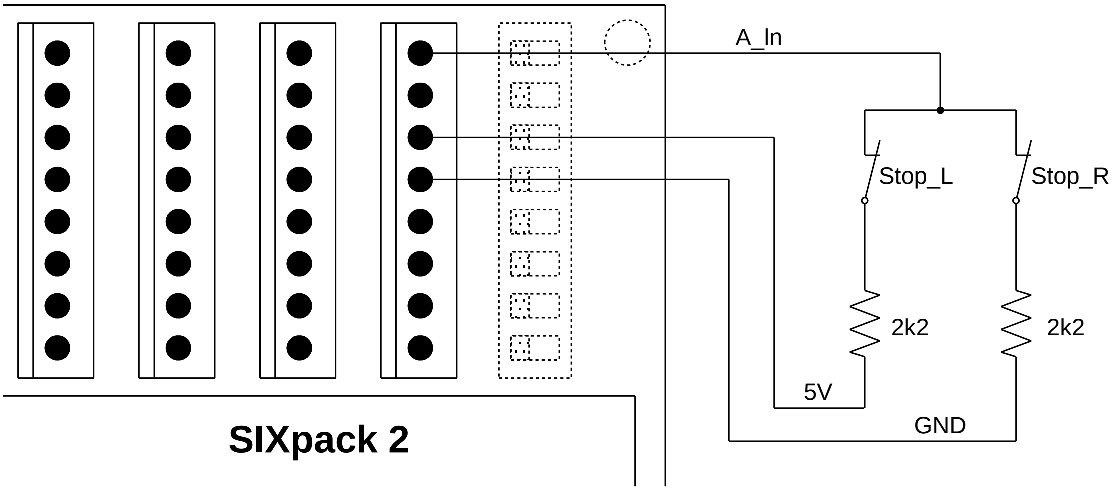


Figure 5.9: Connecting two end switches

5.5.2.3 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.

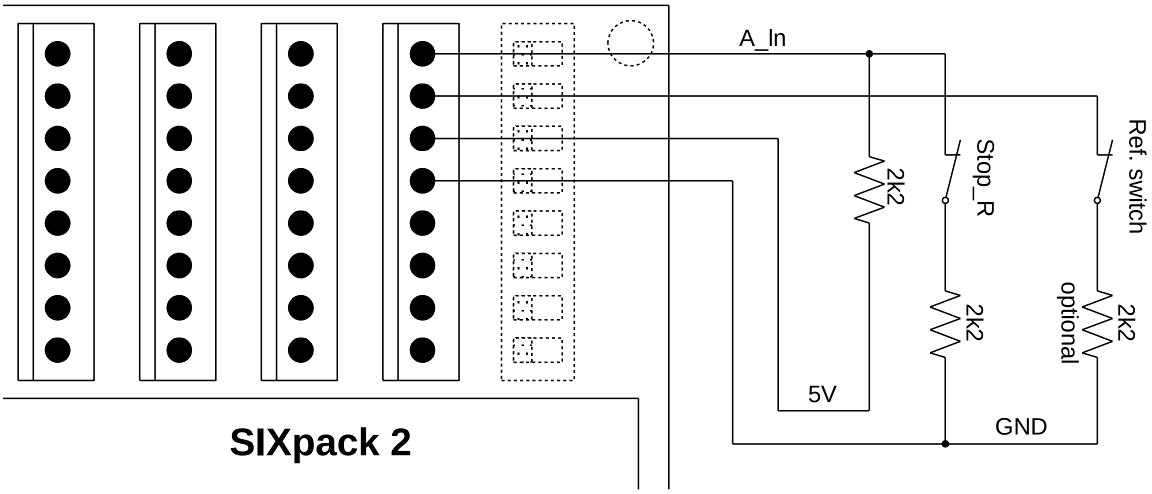


Figure 5.10: Combination of reference and end switches

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). Refer to CMD $15.

5.5.3 Reference search software configuration

Most settings for the reference search are adjusted with the command *SetMotorParameters* (CMD $15, SQPack-tab: "Reference search").

5.5.3.1 Calculation of value range

The positioning is defined by the motors count of micro steps. The pack has 16 microsteps per full step. The angle the motor moves for each full step depends on the motor and can be seen in the motors' manual.

Calculation of the value range of a linear axis is easiest experimental. Therefore set the axis mechanics manually to the start position and move it with use of SQPack to the end of the axis. Use the command "Ramp" (CMD $23, SQPack-tab: "Movement") and set value of *Position* higher until the end of the axis is reached. *Position* represents now the length of the axis.

5.5.3.2 Linear Axis, mechanical reference point

Important parameters for configurations are:

- variable `Poslimit`,
- Flag MT_NULLLEFT,
- Flag MT_MECHREF.

The value range of the axis is specified in `Poslimit`. The SIXpack 2 evaluates with this value the length of the reference search (5/4 * `Poslimit`). MT_NULLLEFT defines the direction of the reference stopper. It is on the left side of the axis for NullLeft = 1.
For `Poslimit` = 10000 the produced dataset by SQPack is:

SendToPack($00, $15, $00, $10, $27, $00, $00, $10, $04).

The reference search is started with command *StartReferenceSearch* (CMD $23), from the SQPack-tab "Reference Search". The dataset is as follows:

SendToPack($00, $22, $00, $00, $00, $00, $00, $00, $00).

5.5.3.3 Linear Axis, reference switch at beginning of axis

The hardware configuration described above is the starting point. For the basic configuration only the variables `Poslimit`, MT_NULLLEFT and MT_NULLPOSITIVE are relevant. MT_NULLLEFT and MT_NULLPOSITIVE are set to 1. The resulting dataset evaluated by SQPack for an axis of the length of `Poslimit` = 33333 is:

SendToPack($00, $15, $00, $20, $A1, $07, $00, $10, $40).

Start of reference search as described in 5.5.3.3.

5.5.3.4 Linear axis, combined end/reference switch left, end switch right

Starting point is the same. Additionally to `Poslimit`, the Flags MT_NULLLEFT and MT_STOPNULL have to be set. The resulting dataset for `Poslimit` = 500000 is:

SendToPack($00, $15, $00, $20, $A1, $07, $00, $50, $40).

Start of reference search as described in 5.5.3.3.

5.5.4 Known issues

First you have to be sure the reference switch is on the left (ROL moves toward reference switch). If this is different, select or deselect the flag NullLeft, depending to your actual setting.

When slow homing movement and then a jitter around the reference point occurs, the Sixpack 2 interprets the reference switch inverted. Either use an inverted reference switch (closer/opener) or change the state of both "Nullpositive" and "NullLeft". Now the homing still goes into the same direction, but with the correct fast algorithm.
When now the motor after homing still keeps moving forth and back, please increase the NullTestRange.

5.6 Basic configurations for operation

5.6.1 Adjusting motor current

Additionally to the possibility to adjust the maximum motor current manually by the DIP switches in 5 stages, (refer to 5.2.7) the motor current can be adjusted more precisely using a software setting (refer to 6.2.1.2).
First, the (maximum) current requirement of the motors has to be determined. Please refer to the motors' datasheet. Commonly the motor data sheet specifies the full step current setting. To match this with the SIXpack 2, a current of 1.4 times the motor's peak current is possible. The command *PeakCurrent* (CMD $10, SQPack-tab „Current") configures the maximum current.

CAUTION: The maximum current setting is always in common for two adjacent motor channels (1 and 2, 3 and 4, 5 and 6). The current setting concerns two motors!!!

The peak current setting issued by CMD $10 has a value range of 0…255. 255 refers to the full current as set by the switch settings. Additionally, CMD $11 specifies a situation dependant current reduction. Typically, the acceleration current is set to 100%, the constant velocity current is set to 100% or less, the standby current is set equal to the constant velocity current or less (e.g. 50%), the power down current is set even lower, down to 0%, depending on the torque required in these situations (refer to 6.7).

The resulting current setting is:

Current = (Peak Current / 256) * (Current specified by DIP switches) * CMD$11 percentage

5.6.2 Configuration of acceleration and velocity

The exact and calculation and configuration of micro steps and the description of parameters concerning calculation of micro steps follow in "6 Full Functionality". In this chapter the detection and configuration of maximum acceleration and velocity is described, only.

5.6.2.1 Configuration of initial velocity

CAUTION: This value may be changed at motor standstill, only !!!
The initial velocity characterizes velocity at start of motion ramp. Value range is 1…256. The command *StartVelocity* (CMD $13, SQPack-Tab „Velocity & Acceleration") configures the initial velocity in the variable `VStart`.
Attention: This command changes the two other values `VMin` and `ClkDiv`, also. These functions are explained in the Instruction Set. The default values `VMin` = 5 and `Div` = 2 are sufficient here.

Example dataset: `VStart` = 100, `VMin` = 5, `Div` = 2 for Motor 3:
SendToPack($00, $13, $02, $0A, $00, $0A, $00, $02, $00)

5.6.2.2 Acceleration and maximum velocity

Acceleration and maximum velocity are related closely. Starting from `VStart` a velocity with fixed frequency is raised by `AMax` until `VMax` is reached. A detailed instruction is provided in the Instruction Set.
Both values are set with command *SetAMaxVMax* (CMD $14, SQPack-Tab „Velocity & Acceleration"). `AMax` has to be ranged from 1 to (`VStart`*64), `VMax` from 1 to 511.

Example dataset: `AMax` = 200, `VMax` = 400 for Motor 4:
SendToPack($00, $14, $03, $C8, $00, $90, $01, $00, $00).

5.6.3 Motion control

Divers commands for the movement of the axis are combined in chapter 6.6 "Commands for axis movements".

6 Full Functionality

6.1 Inputs and Outputs

6.1.1 RS232 or RS485 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 = SIXpack 2 is in sending mode). The CTS-line will be ignored.
When a valid command word is received, the status LED flashes.

6.1.2 CAN interface

The integrated CAN-Controller supports the full CAN-specifications 2.0B. Nevertheless, it is recommended to use 11 address bits, only. 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 SIXpack 2 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. For setting the Baud rate please refer to chapter 5.2.3.

6.1.3 Ready output

The Ready output can be activated (low, open collector, with pullup to 4.3V internally), whenever a motor is active (velocity above 0) or at reference search. The ready output will be switched within 2ms after start/end of motor action. The repeatability (Jitter) matches approximately the micro step rate at start or stop (see command *SetVMinVStart*, CMD $13, SQPack-Tab „Velocity & Acceleration").

6.1.4 Multifunctional connector "RS232"

The 10 pin connector "RS232" next to the motor connectors offers divers inputs and outputs for additional functions of the SIXpack 2.

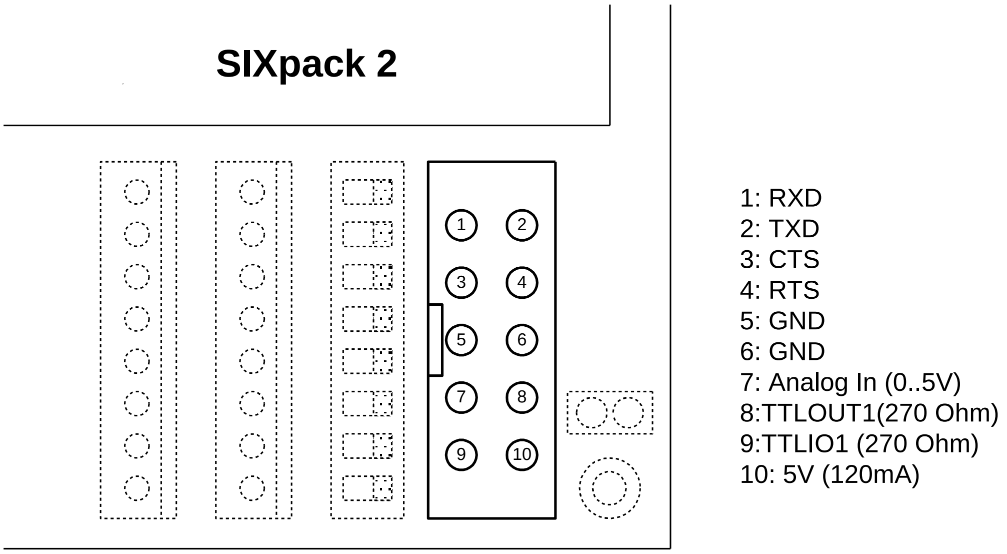


Figure 6.1: Pinning of multifunctional connector "RS232"

Pin	Use	Description
1, 2, 3, 4	connected directly to the sub-D-connector of the RS232 interface	electrically identical with the interface at the sub-D connector and can not operate independent
5, 6	ground	
7	analog input (0..5V)	value readout with command *GetInputValues* (CMD $30)
8	digital output (TTL)	TTL levels, internal resistor 270Ω. Value readout with command *SetOutputs* (CMD $32) in variable TTLOUT1. Function "Ready output" is activated if TTLOUT1_READY is set.
9	digital I/O port (TTL)	TTl levels, internal resistor 270Ω. Command *SetOutputs* (CMD $32) : TTLIO_INPUT = 0: Output, set with variable TTLIO1 TTLIO_INPUT = 1: Input, value readout with variable TTLIO1 of *GetInputValues* (CMD $30).
10	output, max 5V, 120 mA	

Table 6.1: Pin description of multifunctional connector "RS232"

6.1.5 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).

6.2 Programming

The concept programming the SIXpack 2 is based on dataset of a fixed length. To allow networks the datasets have to contain the SIXpack 2 address.

The commands dataset itself contains a command byte and seven bytes to transfer parameters. The bytes have to be transmitted even if they contain no data.

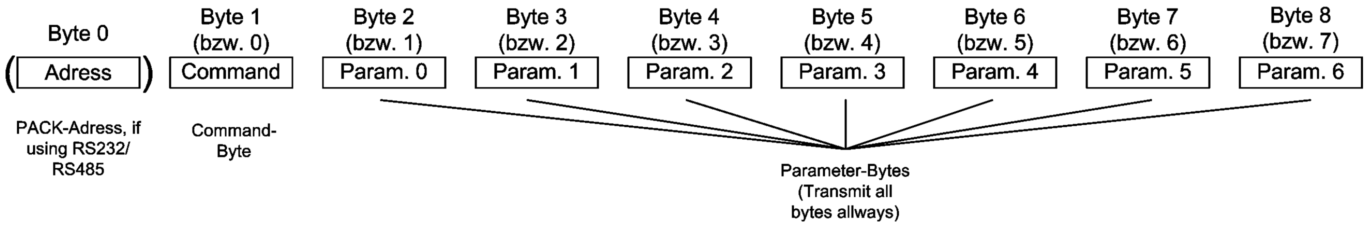


Figure 6.2: Command dataset

6.2.1 Hints for Programming

6.2.1.1 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:

6.2.1.2 Setting of Motor Current, refer to 5.6.1

Configure maximum current (s. CMD $10) and current control (s. CMD $11) as needed. The resulting current depends on the maximum current defined by the DIP-switches. It is recommended to use motors with the same maximum current for motors 1 & 2, 3 & 4 and 5 & 6, since their peak current setting (s. CMD $10) is linked together. The peak current multiplies the maximum current defined by the DIP switches with value / 256. This provides a very exact peak current setting. The minimum current (which provides proper microstepping) selected by current control is 19% (Index 6) for every parameter.
The current control (s. CMD $11) provides a coil current in percent of the peak current.

Example: Connecting a 1A motor to channel 1 and a 600mA motor to channel 2
Motor 1 is specified by the manufacturer for 1A RMS current. The required peak current then is 1.4A in microstep operation. The nearest DIP switch setting is 1.5A. Now, set the peak current setting to 1.4/1.5*255 = 238. For motor 2, peak current requirement is 600mA*1.4=840mA. We now can try with a DIP switch setting of 1.0A. Using the setting for motor 1, This gives us a peak current of 1A*238/255=0.93A. This is too much, but now we have the following possibilities: 1. Scale down the situation dependant setting. 2. Reduce DIP current setting to 800mA, resulting in 0.74A peak, which is slightly lower than desired. Or 3., reduce motor peak current setting for motor 1 and 2. Be careful not to exceed the motor's maximum specified current! It is easier to pair same motors at neighboring channels.

6.2.1.3 Velocity Configuration (global)

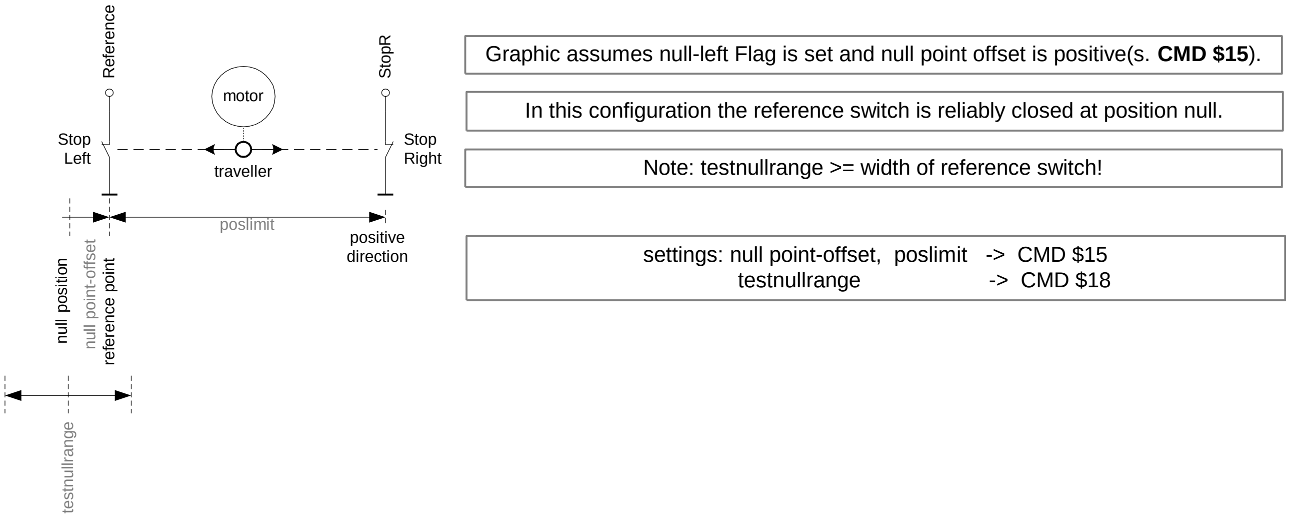
Calculate *clkdiv* (s. CMD $12, SQPack-Tab "Velocity & Acceleration") with the step frequency formula (s. begin of chapter Instruction Set) so that the required maximum step velocity is achieved with v_i = 511 and small values div_i = 0 or 1.

6.2.1.4 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.

6.2.1.5 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.



- 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 *To* 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.

6.2.1.6 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.

6.2.1.7 Interlacing of Requests

Requests must not be interlaced. Each request should wait for the response of the SIXpack 2 before transmitting a new command. However a delayed response with RS 232 may be outstanding in parallel.

6.2.1.8 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= *Delayedtesto | 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

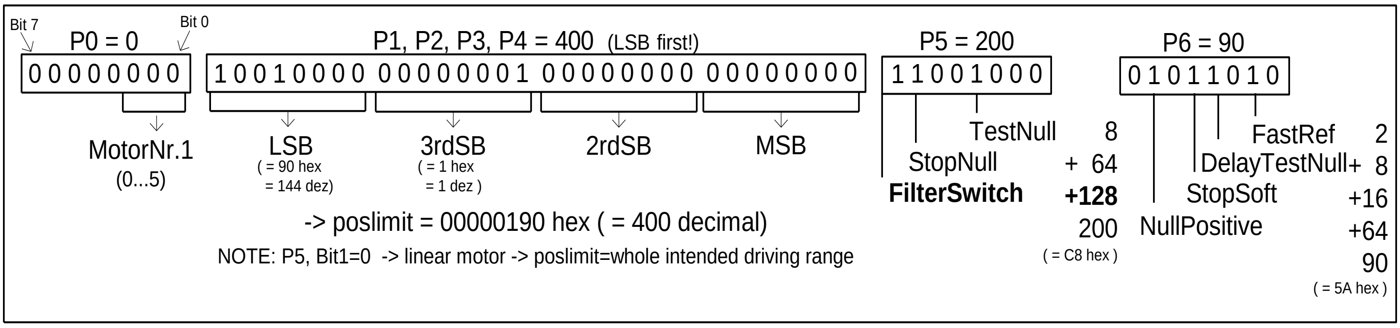
All other values are set to 0, i.e. the functions are disabled.

6.2.2 Examples

<u>Attention:</u> All 9 Bytes must be sent to the interface, otherwise the SIXpack 2 does not recognize the command and waits for the missing bytes.
$ indicates that the value is in hexadecimal notation!

6.2.2.1 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!



<u>Pseudocode:</u>
SendToPack(address); // Address of the *Pack*(Sixpack2)
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);

6.2.2.2 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 SIXpack2
sendToPack($23);                         // Command for starting a trapezoidal Ramp
sendToPack(motno);                       // 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
```

6.2.2.3 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 SIXpack
sendToPack($20);                         // Command for inquiring actual position and action of one motor
sendToPack(motno);                       // 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
motno        = 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
```

6.2.2.4 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 Sixpack2
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 Sixpack2
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 SIXpack2
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
```

6.3 Adjusting SIXpack 2 to motors micro step characteristics

6.3.1 Calculation of micro step frequency

The SIXpack 2 operates with a fixed micro step frequency of 16 micro steps every full step. The programming positions are specified in micro steps in relation to zero. Following formula calculates the given motors micro steps for a whole rotation:

Number of micro steps / rotation = 360° / Full step angle * 16

The full step angle is specified in the motors datasheet.

The absolute micro step frequency, the number of steps per second, in relation of the motors actual velocity settings is described with following formula:

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

- Full step frequency = 1/16micro step frequency
- f_{clk} is 20MHz
- *Clkdiv* (ClockDivider, $12, SQPack-Tab "Velocity & Acceleration") is the same for all motors (value range 0…31)
- v_i respectively *vakt* is the velocity of each motor (range: –511..+511)
- div_i (Div, SQPack-Tab "Velocity & Acceleration") can be parameterized for each motor (range 0…3)

Note: The micro step frequency must not exceed 200 kHz.

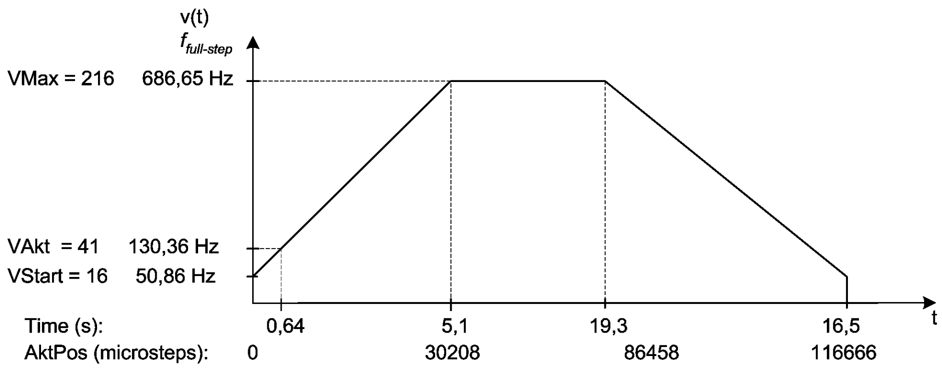
6.3.1.1 Example

The starting position (get with *GetPositionAndActivity*, CMD $20, set with *SetActualPosition*, CMD $27) is zero, target position is 116666 (refer to command *StartRamp*, CMD $23).
Actual full step frequency ($f_{full-step}$) is calculated with given formula for $f_{micro-step}$ by multiplication with 16. VAkt for accelerations is:

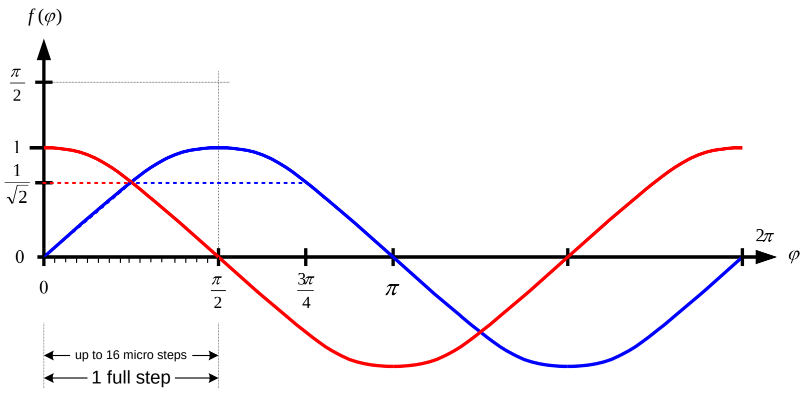
$$\mathrm{VAkt} = \mathrm{VStart} + \frac{\mathrm{AMax}/64}{\mathrm{t}/2\,\mathrm{ms}}$$

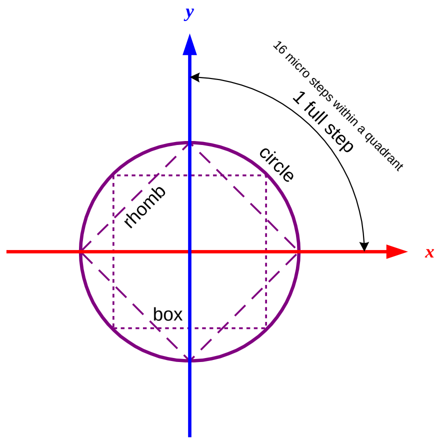
This formula is not valid for decelerations. To realize the actual function the SIXpack 2 decelerates more slowly than it accelerates.

Parameter:
ClockDivider = 5,
Div = 2,
VStart = 16,
VMax = 128,
AMax = 5

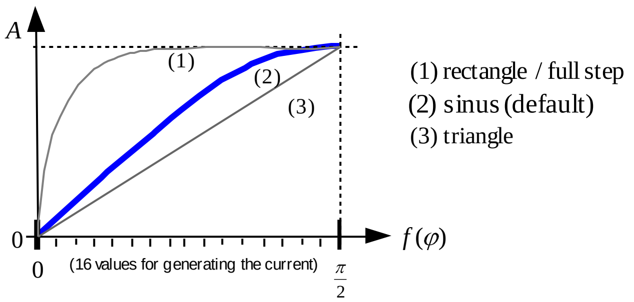


6.3.2 Adapting the microstep-table to the motor characteristics





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)

6.4 Reference point adjustments

Additionally to the standard configurations for end and reference switches described in chapter 5.5 there are many possibilities to adept these settings to an individual configuration.

6.4.1 Coordinate plane of an axis

In standard configuration the position counter for each axis is zero. The target position value for a movement is the absolute number of micro steps from zero to target position. The maximum value of micro steps per axis is adjusted by variable `Poslimit` in command *SetMotorParameters* (CMD $15, SQPack-Tab "Reference search".

6.4.2 Reference point / reference switch

The reference point is defined by a mechanic stopper or a reference switch. If an active reference switch is used, in opposition to the recommended passive switch (refer to chapter 5.5.2), the Flag MT_NULLPOSITIVE (command *SetMotorParameters*, CMD $15) has to be set to zero (MT_NULLPOSITIVE = 0).

6.4.3 Moving zero-point

In basic configuration the reference point is the zero-point, also. With the command *SetNullPointOffset* (CMD $18)an offset of the zero-point is possible. This can be important if for example the reference switch is at the center of the axis but the programming allows positive values, only.

6.4.4 Automatic reference search

The SIXpack 2 provides the possibility to start an automatic reference search at identified lost of steps. To activate this function set the Flag MT_TESTNULL (command *SetMotorParameters*, CMD $15). The intern position counter will be compared with the status of the reference switch. If the reference switch reacts at wrong position or does not react at internally logic position, a reference search is started and the flag STOPFLAG (command *GetPositionAndActivity*, CMD $20) set.

6.4.5 Adjusting activity zone of reference switch

Mechanical inaccuracy may lead to reaction of the reference switch a few steps too early or too late. If an automatic reference search (MT_TESTNULL) is used this may start unintended reference searches. It is possible to define a zone around the zero-point in which the status of the reference switch is not checked using the commands *SetNullPointOffset*, (CMD $18) variable `TestNullRange`.

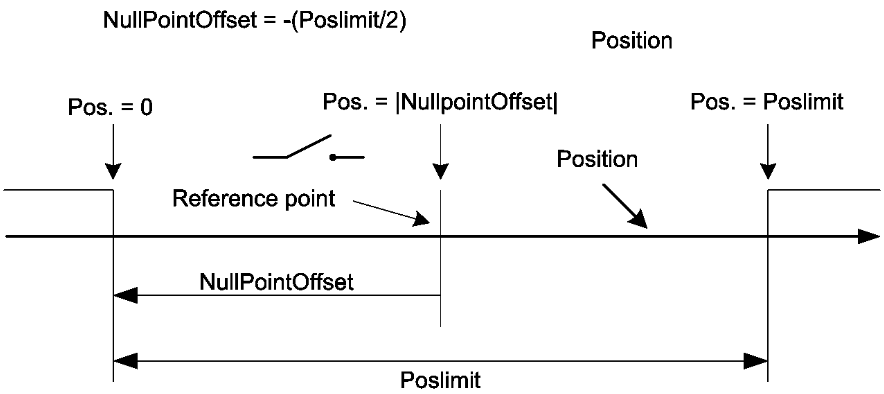


Figure 6.3: Coordinate transformation using „NullPointOffset"

6.4.6 Compensation of reference switch delay

At use of mechanic reference switches the problem of a delay time problem between mechanic and electronic status change occurs often. To avoid dysfunction it is possible to delay the request of the reference switch signal with `PowerDownDelay`. It is set and used in *SetCurrentControl*, CMD $11. MT_DELAYTESTNULL is set and `PowerDownDelay` is 500 in basic configuration.

NOTE: `PowerDownDelay` is used for "power down"- delay (command *SetCurrentControl*, CMD $11), also.

6.4.7 Elimination of glitches

In standard configuration the input signal at RefIn is debounced for 22 ms. The change of levels has to be constant in this time to be interpreted. This value is set with variable `DebouncingTime` (command *SetRefSearchParameters*, CMD $16).

6.4.8 Adjusting reference search velocity

Fast reference search is activated in standard configuration (Flag MT_FASTREF of command *SetMotorParameters*, CMD $15). At this configuration the axis is moved at reference search with the velocity set in variable *VRefMax* (default *VrefMax*=100).

CAUTION: If MT_FASTREF is used the length of the axis has to be defined in the variable `Poslimit`. Otherwise no correct reference search is possible. If MT_FASTREF is not set `Poslimit` is not imperative. The reference search will be with a fixed velocity of 1.

6.4.9 Aborting reference search

A reference search is aborted with CMD $2B, SQPack-Tab "Reference search".

6.5 End switch configurations

To recognize a lost of steps and to avoid a possible mechanic damage it is reasonable to limit linear axis as described in 5.5.

6.5.1 A_In as end switch input

End switches to limit the axis can be connected to the input A_In. Additionally to the in 5.5 described setup with two opening end switches other configurations are possible:
A_In is an analog input. In use as end switch voltage levels below a defined level are interpreted as the left and above an other level as right end switch. These levels are defined with the command *SetStopSwitchLimits*, CMD $30, SQPack-tab "I/O" in the variables `MinLeft` bzw. `MaxRight`.

6.5.2 Combination of end and reference switches

If the reference switch is at the end of the axis it can be used as en switch also. This function is activated by setting the flag MT_STOPNULL (command *SetMotorParameters*, CMD $15). The flag MT_NULLLEFT (command *SetMotorParameters*) defines, on which side of the axis the combined end/reference switch is located.

MT_NULLLEFT = 0 → end/reference switch is on the right.

6.5.3 "Security Margin" for combined end/reference switch

As described in chapter 6.4.5 errors can occur if a switch reacts too late or too early. It is recommended to provide the combined end/reference switch with a "security margin".
The value set with the command *SetMargin*, CMD $1A, SQPack-tab "Reference search", defines the number of steps the SIXpack 2 runs into the reference switch (see Figure 6.3 also).

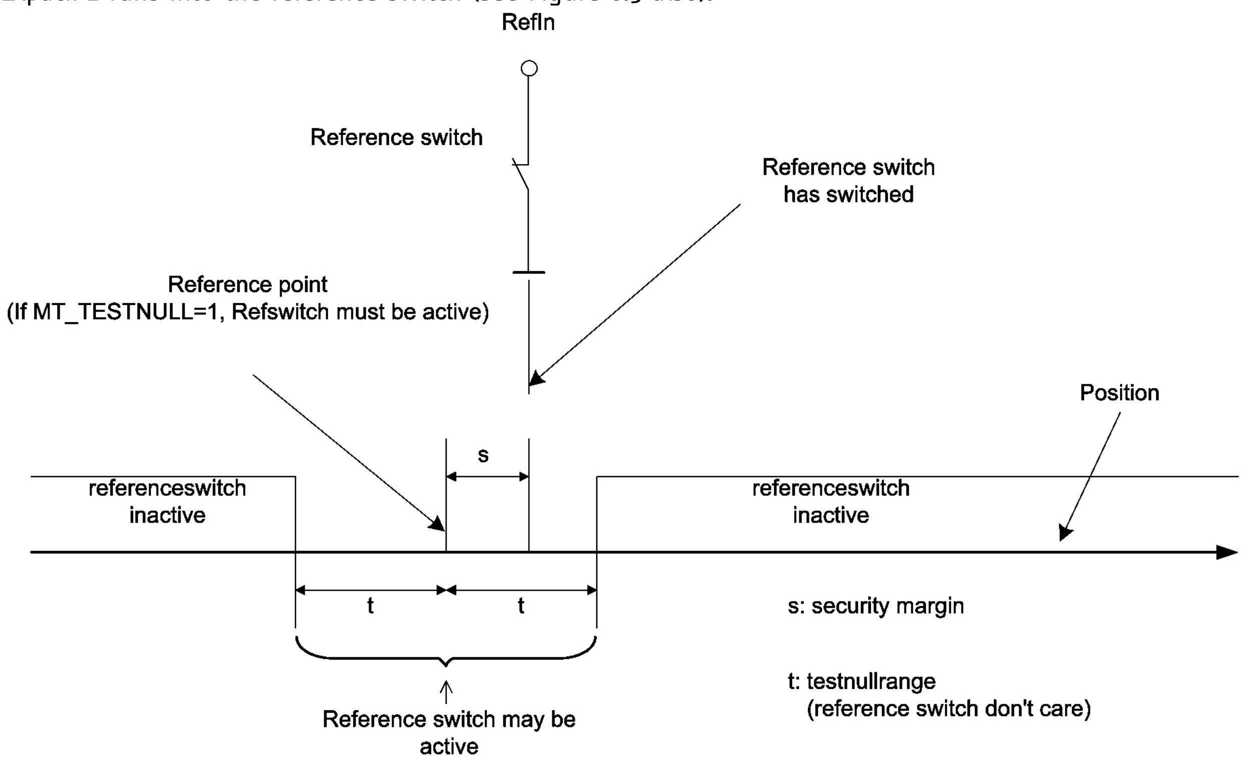


Figure 6.4: Adjustments for reference switch function

6.6 Commands for axis movements

The SIXpack 2 provides different possibilities to move the axis. Which one to choose is related to the demands of the application.
Basically there is the possibility to move the motor on the axis from a to b (ramps run) or with a fixed velocity (rotation) until the next velocity command.

6.6.1 Basic ramp run

The ramp run requires the configuration of a reference run with the values for `Poslimit`, acceleration and velocity (see CMD $15).
Through the command *StartRamp*, CMD $23 every position, defined by `Poslimit` can be achieved. The target is defined by the variable `TargetPosition` (CMD $24, SQPack-tab "PI Controller"). The motor is decelerated before it reaches the target position to stop at the exact target.

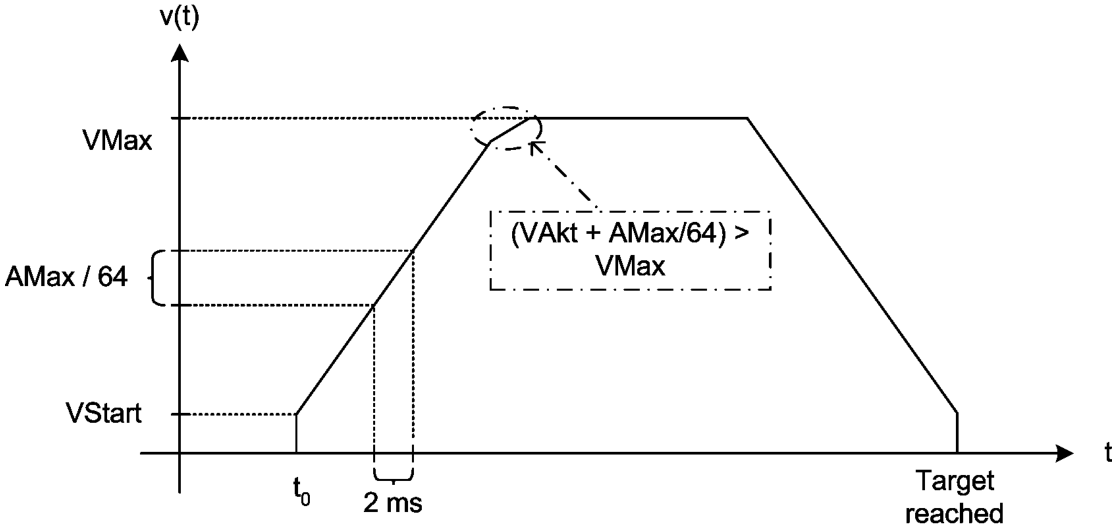


Figure 6.5: Schema of ramp generation

6.6.2 Start of constant rotation

For a rotation with a constant velocity the command *ConstantRotation* (CMD $25) transmits the variable `Velocity` which defines the velocity for the constant move (calculation in 6.3).
After receiving the command the motor is accelerated with the maximum acceleration `AMax` until the speed defined in `Velocity` is reached. To stop the axis the command *ConstantRotation* is used again and `Velocity` is set to 0.

6.6.3 Change target position for ramp run

To change the target position of an active ramp run use the command *SetTargetPosition* (CMD $24, SQPack-tab "PI Controller") to transmit the variable `TargetPosition` with the new target of the movement.
If the motor already reached its defined target *SetTargetPosition* takes no effect. To avoid failures and to be sure the desired position is reached it is recommended to send the command *StartRamp* (CMD $23) with the actual target position in addition afterwards. If the mode *ConstantRotation* (CMD $25) was active, before, the motor is stopped and than moved to the target position.

6.6.4 Starting different motors synchronous

The command *StartRampParallel* (CMD $29) is used to start multiple ramp runs. Before this command is sent each target position should be set with the command *SetTargetPosition* (CMD $24). When all motors are inactive the parallel ramp run can be started.

Example: In Figure 6.6 (left) a parallel started ramp run of Motor 1 and Motor 2 is demonstrated. The parameters for velocity and acceleration (refer to 6.6.1) are identical for both motors but the distance for motor 1 is longer. Both motors start at the same time, accelerate synchronous and have the same maximum velocity. Motor 2 reaches its target position earlier and therefore finishes the ramp earlier than motor 1.

6.6.5 Starting linear interpolation of multiple axis

A linear interpolated movement of different axis is possible with the command *StartInterpolation* (CMD $50). All motors reach the target at the same time. The target has to be set previously at motor standstill with the command *SetTargetPosition* (CMD $24).
The axis with the longest distance, i.e. the longest movement time at maximum velocity, is the leading axis. The velocities of the other axis rely on this leading axis. A small inhomogeneity may occur, nevertheless. Therefore the status of all moved motors should be checked before the next command.

Example: In Figure 6.6 (right) a linear interpolated ramp run is demonstrated. Motor 1 has a longer distance to move than motor 2. Therefore Motor 1 is accelerated to the velocity defined in `VMax`. For motor 2 a lower maximum velocity has been calculated. Both motors reach their target simultaneously. The longer the distance the lower is the acceleration compared to the other motors. Please refer to chapter 7.5.

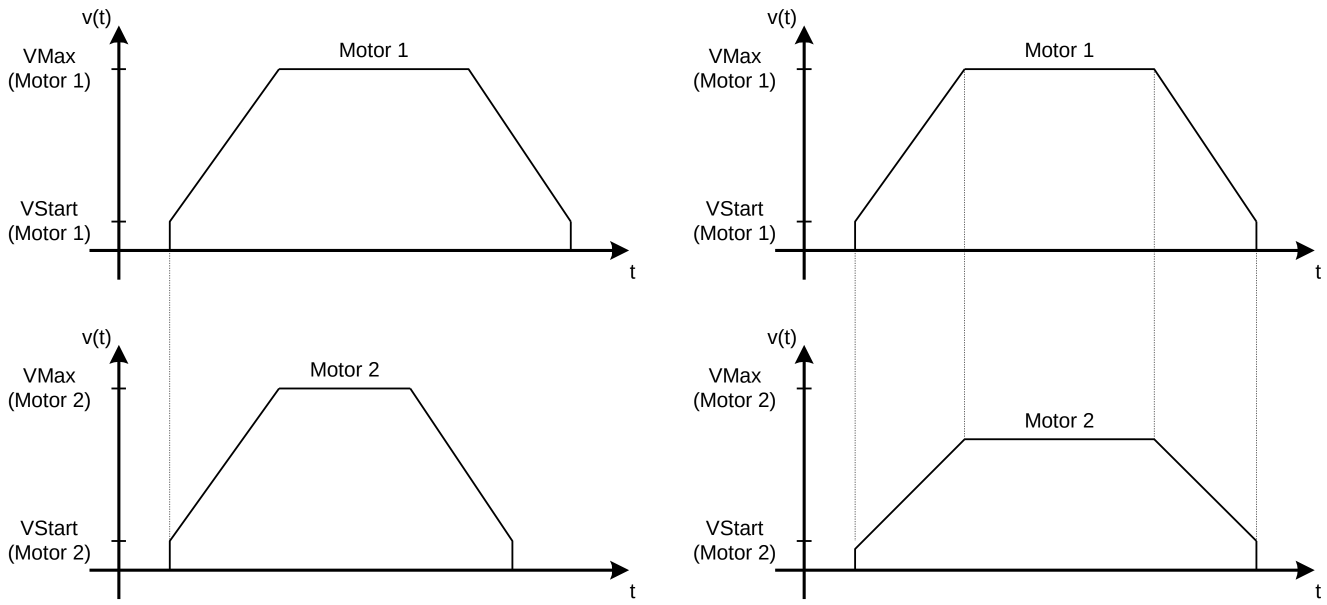


Figure 6.6: Synchronous start of motors (left) and Linear interpolated ramp (right)

6.6.6 Configuration for rotating movements

The SIXpack 2 differentiates between linear and rotating movements. The active mode is defined by the flag MT_ROTARY (command *SetMotorParameters*, CMD $24).
Most important is the interpretation and definition of the positioning values. In linear mode the difference between actual and target position is interpreted as absolute value for the micro steps to run. Target positions may be out of the value range defined by `Poslimit`.
The value range is strictly limited by the value of `Poslimit`. Positions above this value are adjusted with a modulo operator to a multiple value. The flag MT_NULLLEFT (command *SetMotorParameters*) defines in rotating mode if the positioning values are in positive or negative area:

MT_NULLLEFT = 0: 0 ≤ position value < `Poslimit`
MT_NULLLEFT = 1: - (`Poslimit`) < position value ≤ 0

The direction of the motors rotation is in standard configuration identical to the linear mode. Movements with rising coordinates lead to clockwise rotation and with falling coordinates to counter clockwise rotation.
If the flag MT_OPTIMIZEWAY = 1 (command *SetMotorParameters*) is set the SIXpack 2 chooses the shortest way to get to the target position from the starting position.

6.7 Control of motor current

To provide an always sufficient torque or holding torque with the lowest use of power an individual configuration of the motor currents is possible.
The maximum coil current `IMax` is set with the command *SetMaxCurrent* (CMD $10). Please check the DIP-switches of the SIXpack 2, also (refer to 5.2.7).
Also the command *SetCurrentControl* (CMD $11) provides the possibility to adapt to the actual movement. In the variables a percentage grading can be defined.

EXAMPLE:

Current	% of max current	Variable setting
while acceleration	100	`AccelerationCurrent` = 0
while at constant velocity	75	`RunningCurrent` = 1
at target position	38	`StandigCurrent` = 3
after expiration of `PowerDownDelay`	9	`PowerDownCurrent` = 7

To lower the current consumption unused motors should be set to the value of a power down current of zero (`PowerDownCurrent` = 8).

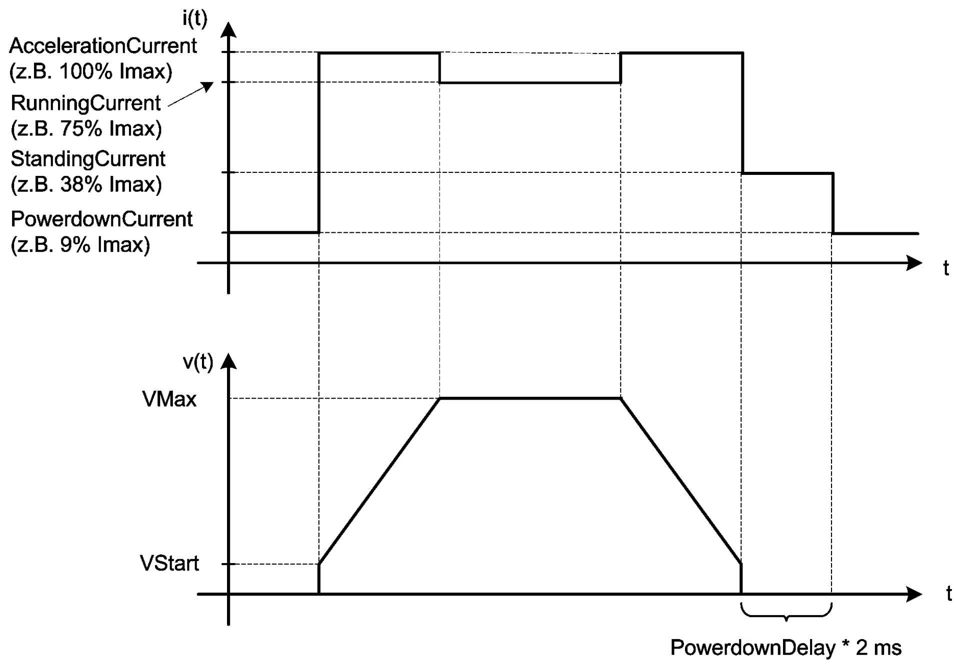


Figure 6.7: Ramp depending current setting

6.8 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= *Delayedtesto | 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 Instruction Set

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

7.1 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	*motno (0…5)*
P1	value (0..255): SIXpack: $I_{max,DIP}$ * value/256 (for $I_{max,DIP}$ refer to 5.2.7 Adjusting the maximum current)

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	*motno (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	*motno (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	*motno (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

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 6.1: Hints for Programming!

CMD	$15
P0	*motno (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	*motno (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	*motno (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	*motno (0...5)*
P1	*propdiv* (1..255)
P2,3 #	*intdiv* (1..32767)
P4,5 #	*intclip* (1..32767)
P6	*intinpclip* (0..255)

7.2 Driving Ramps

Query Position and Activity

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

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

Response	
CMD	$21
P0	*motno (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	*motno (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	*motno (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	*motno (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	*motno (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 decelerate with amax instantly and continue driving a reverse ramp on its own.

CMD	$26
P0	*motno (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	*motno (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 <u>all concerned</u> motors are inactive. The response contains the actual action of <u>all</u> 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)

Abort reference search

The reference search for the specified motor is aborted.

CMD	$2B
P0	*motno (0...5)*

7.3 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)

7.4 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)

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.

7.5 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
Po	mask for motors (bit 0=motor 0, bit 5=motor 5) (0: motor unused, 1: motor used in multi-dimension motion)

7.6 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
Po	address for response
P1,2,3,4 #	magic code

Response	
CMD	$F2
Po	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
Po-P6 #	7 data bytes

Query flash Memory Check-Sum and abort Programming if necessary

CMD	$F4
Po	address for response

response	
CMD	$F4
Po,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
Po	address for response

response	
CMD	$F5
Po-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

8 Instruction table

9 Declaration of Conformity

Declaration of Conformity

for

SIXpack 2
6-Axis stepper motor controller / driver module

Manufacturer

TRINAMIC Motion Control GmbH & Co. KG
Sternstraße 67
D – 20357 Hamburg
Germany
www.trinamic.com

Statement of Conformity

Based on test results using applicable standards, the product is CE compliant and meets the European and national guidelines.

Sample tests have been accomplished by an accredited test labor (EMV Services GmbH & Co. KG, Schloßstraße 6-12, D-21079 Hamburg, Germany).

Applied Standards:

EN 61000-6-4 (2002)	Generic emission standard; part 2: industrial environment
EN 61000-6-2 (2002)	Generic immunity standard; part 2: industrial environment
EN 61000-4-2 (2001)	Immunity of electrostatic discharge
EN 61000-4-3 (2003)	Immunity of radiated, radio frequency electromagnetic field
EN 61000-4-6 (2003)	Immunity of conducted disturbances, induced by radio frequency fields

This conformity is indicated by the symbol CE, i.e. "Conformité Européenne".

10 Test Reports

EMV Services	Test report	Reference	Date	Page
Emission Immunity	No. 05/5183-1-1	EMV-05/5183-1-1	Nov. 29, 05	2 / 2

Customer: TRINAMIC Motion Control GmbH & Co. KG
Sternstraße 67
20357 Hamburg

Equipment under test: SIXpack 2 V1.0, S/N: S02000005/11A000001

Date of receipt: Nov. 24, 2005

Date of test: Nov. 24 and 25, 2005

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

Test personnel:	Tel.	Fax	E-mail
Dipl.-Ing. Z. Wang	040/766293433	040/76629506	wang@emv-services.de

Applied standards:

Emission:

EN 61000-6-4 (2002): **Generic emission standard; Part 2: Industrial environment**

Immunity:

EN 61000-6-2 (2002): **Generic immunity standard; Part 2: Industrial environment:**

- EN 61000-4-2 (2001): Immunity to electrostatic discharge,
- EN 61000-4-3 (2003): Immunity to Radiated, radio-frequency electromagnetic field,
- EN 61000-4-6 (2003): Immunity to conducted disturbances, induced by radio frequency fields.

Test results:

Emission:

The device complies with the limits for radiated emission.

Immunity:

The device complies with the requirements of immunity to:

- electrostatic discharge with criterion A,
- radiated, radio-frequency electromagnetic field with criterion A,
- conducted disturbances, induced by radio frequency fields with criterion A.

The test results only apply to the Equipment under test.

Dr. E. Sauer
Lab manager

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

EMV Services GmbH & Co. KG
Ein Unternehmen der TÜV Nord Gruppe
Harburger Schlossstraße 6-12
D-21079 Hamburg

11 Revision History

Version	Comment	Author	Description
1.06	21-Feb-2007	HC	Revision History added, Min supply voltage corrected to 13V, max. current depends on DIP switches (in 7.1 – Peak current)
1.07	20-Jun-2007	HC	Figure 5.9, Figure 5.10 in line resistor not optional. Figure 6.6 acceleration corrected.
1.08	24-October-2007	HC	5.2.6 Driver enable electrical information updated
1.09	04-March-2008	GE	CAN 2.0B support corrected
1.10	29-Jan-10	OK	Declaration of Conformity added

Table 11.1: Documentation Revisions