PZ281E GCS Commands Manual

E-727 Digital Multi-Channel Piezo Controller

Release: 1.3.0 Date: 29 June 2022

This document describes GCS commands for the following product:

- **E-727**
  Digital Multi-Channel Piezo Controller

Physik Instrumente (PI) GmbH & Co. KG, Auf der Roemerstrasse 1, 76228 Karlsruhe, Germany
Phone +49 721 4846-0, Fax +49 721 4846-1019, Email info@pi.ws, www.pi.ws

First printing 29 June 2022
Document Number PZ281E BRo, Release 1.3.0
E-727-GCS-Commands-PZ281E130.doc

Contents

1 Introduction

This manual describes the commands of the PI General Command Set (GCS), version 2.0, that are valid for E-727 digital multi-channel piezo controllers.

Read the E727T0005 user manual of the E-727 before you operate the system using the GCS commands. The E727T0005 user manual describes the following:

- Safety precautions and intended use
- Specifications and operating elements
- Functionality and parameters
- How to install and operate the E-727
- How to use special features, e.g. data recorder, wave generator, DDL, digital and analog input and output, macros
- SPI interface of the E-727
- EtherCAT interface of the E-727

Other applicable documents:

Description	Document
E-727.AS Digital Multi-Channel Piezo Controller With Area Scan Routines for Fast Optical Alignment	E727T0012 User Manual
PI GCS 2 DLL	SM151E Software Manual
PIMikroMove	SM148E Software Manual
PIFRF-Analyzer	SM159E Software Manual
PI GCS2 driver library for use with NI LabVIEW software	SM158E Software Manual
PIPython	SM157E User Manual
PI MATLAB Driver GCS 2.0	SM155E Software Manual
GCS array data format description	SM146E Software Manual
PI Update Finder	A000T0028 User Manual

The latest versions of the relevant manuals are available for download on our website (www.pi.ws).

For inquiries and orders, contact your PI sales engineer or send us an e-mail (info@pi.ws).

2 GCS Commands

The PI General Command Set (GCS), version 2.0, is supported by a wide range of PI systems. This command set is well-suited for positioning tasks with one or more axes. The command set itself is independent of the specific hardware (controller or attached stages).

GCS commands are used to set operating modes, initiate axis motion and to query system and motion values. Because of the variety of functions and parameters, a sequence of GCS commands must often be transferred in order to achieve a desired system action.

You can type GCS commands, for example, in the *Command Entry* window of PIMikroMove, or in the PITerminal.

You can send GCS commands also via data segment 2 of the SPI interface, see the E727T0005 user manual for more information.

2.1 Format

2.1.1 Notation

The following notation is used to define the GCS syntax and to describe the commands:

<...>	Angle brackets indicate an argument of a command, can be an item identifier (see “Axes, Channels, Functional Elements” in the E727T0005 user manual) or a command-specific parameter
[…]	Square brackets indicate an optional entry
{…}	Braces indicate a repetition of entries, i.e. that it is possible to access more than one item (e.g. several axes) in one command line.
LF	LineFeed (ASCII char #10), is the default termination character (character at the end of a command line)
SP	Space (ASCII char #32), indicates a space character
"..."	Quotation marks indicate that the characters enclosed are returned or to be entered.

2.1.2 GCS Syntax

A GCS command consists of 3 characters, e.g. CMD. The corresponding query command has a question mark added to the end, e. g. CMD?.

Command mnemonic:

CMD ::= character1 character2 character3 [?]

Exceptions:

- Single-character commands, e. g. fast query commands, consist only of one ASCII character. The ASCII character is written as combination of # and the character code in decimal format, e. g. as `#24`.
- `*IDN?` (for GPIB compatibility).

The command mnemonic is not case-sensitive. The command mnemonic and all arguments (e. g. axis identifiers, channel identifiers, parameters, etc.) must be separated from each other by a space (SP). The command line ends with the termination character (LF).

CMD[{SP}<Argument>}]LF

CMD?[{SP}<Argument>}]LF

Exception:

- Single-character commands are not followed by a termination character. The response to a single-character commands is followed by a termination character, however.

The argument <AxisID> is used for the logical axes of the controller. Depending on the controller, an axis identifier can consist of up to 16 characters. All alphanumeric characters and the underscore are allowed. See "Axes, Channels, Functional Elements" in the E727T0005 user manual for the identifiers supported by the E-727.

Example 1:

Axis 1 is to be moved to position 10.0. The unit depends on the controller (e. g. µm or mm).

Send: `MOV SP 1 SP 10.0 LF`

More than one command mnemonic per line is not allowed. Several groups of arguments following a command mnemonic are allowed.

Example 2:

Two axes which are connected to the same controller are to be moved:

Send: `MOV SP 1 SP 17.3 SP 2 SP 2.05 LF`

When a part of a command line cannot be executed, the line is not executed at all.

When all arguments are optional and are omitted, the command is executed for all possible argument values.

Example 3:

All parameters in the volatile memory are to be reset.

Send: `RPA`LF

Example 4:

The position of all axes is to be queried.

Send: `POS?`LF

The response syntax is as follows:

[<Argument>[{SP<Argument>}]"="]<Value>LF

With multi-line replies, the space preceding the termination character is omitted in the last line:

{[<Argument>[{SP<Argument>}]"="]<Value>SPLF}

[<Argument>[{SP<Argument>}]"="]<Value>LF for the last line!

In the response, the arguments are listed in the same order as in the query command.

Query command:

CMD?SP<Arg3>SP<Arg1>SP<Arg2>LF

Response to this command:

<Arg3>"="<Val3>SPLF

<Arg1>"="<Val1>SPLF

<Arg2>"="<Val2>LF

Example 5:

Send: `TSP?`SP`2`SP`1`LF

Receive: `2=-1158.4405`SPLF

`1=+0000.0000`LF

2.1.3 Limitations for GCS Commands

More than one command mnemonic per line is not allowed.

The number of characters per line is limited to 256 byte (1 character = 1 byte). This means that the number of arguments following a command mnemonic is limited to 32.

Example:

If you send

```
TWS 1 100 1 1 200 1 1 300 1 1 400 1 1 500 1 1 600 1 1 700 1 1 800 1 1 900 1 1 1000 1 1 1100 1
```

the controller will return error 24 ("Incorrect number of parameters") when you ask with the ERR? command afterwards because the number of arguments is 33.

2.2 GCS Command Survey

Command	Format	Short Description	Details see
#5	#5	Request Motion Status	p. 13
#7	#7	Request Controller Ready Status	p. 14
#8	#8	Query If Macro Is Running	p. 14
#9	#9	Get Wave Generator Status	p. 15
#24	#24	Stop All Axes	p. 15
*IDN?	*IDN?	Get Device Identification	p. 16
ADD	ADD <Variable> <FLOAT1> <FLOAT2>	Add and Save To Variable	p. 17
AOS	AOS {<AxisID> <Offset>}	Set Analog Input Offset	p. 19
AOS?	AOS? [{<AxisID>}]	Get Analog Input Offset	p. 22
ATZ	ATZ [{<AxisID> <LowValue>}]	Set Automatic Zero Point Calibration	p. 22
ATZ?	ATZ? [{<AxisID>}]	Get State Of Automatic Zero Point Calibration	p. 25
CCL	CCL <Level> [<PSWD>]	Set Command Level	p. 26
CCL?	CCL?	Get Command Level	p. 27
CPY	CPY <Variable> <CMD?>	Copy Into Variable	p. 27
CST?	CST? [{<AxisID>}]	Get Assignment Of Stages To Axes	p. 28
CSV?	CSV?	Get Current Syntax Version	p. 28
CTO	CTO {<TrigOutID> <CTOPam> <Value>}	Set Configuration Of Trigger Output	p. 29
CTO?	CTO? [{<TrigOutID> <CTOPam>}]	Get Configuration Of Trigger Output	p. 34
DDL	DDL <DDLtableID> <StartPoint> {<ValueN>}	Set DDL Table Value(s)	p. 34
DDL?	DDL? [<StartPoint> [<NumberOfPoints> [{<DDLtableID>}]]]	Get DDL Table Value(s)	p. 35
DEL	DEL <uint>	Delay The Command Interpreter	p. 37
DIA?	DIA? [{<MeasureID>}]	Get Diagnosis Information	p. 37
DIO?	DIO? [{<DIOID>}]	Get Digital Input Lines	p. 39
DPO	DPO [{<AxisID>}]	DDL Parameter Optimization	p. 40
DRC	DRC {<RecTableID> <Source> <RecOption>}	Set Data Recorder Configuration	p. 41
DRC?	DRC? [{<RecTableID>}]	Get Data Recorder Configuration	p. 44
DRL?	DRL? [{<RecTableID>}]	Get Number Of Recorded Points	p. 45
DRR?	DRR? [<StartPoint> <NumberOfPoints> [{<RecTableID>}]]	Get Recorded Data Values	p. 45
DRT	DRT {<RecTableID> <TriggerSource> <Value>}	Set Data Recorder Trigger Source	p. 47
DRT?	DRT? [{<RecTableID>}]	Get Data Recorder Trigger Source	p. 48
DTC	DTC {<DDLtableID>}	Clear DDL Table Data	p. 49
DTL?	DTL? [{<DDLtableID>}]	Get DDL Table Length	p. 49
ERR?	ERR?	Get Error Number	p. 50

Command	Format	Short Description	Details see
FDR	FDR <routine name> <scan axis> <scan axis range> <step axis> <step axis range> [L <threshold level>] [A <alignment signal input channel>] [F <frequency>] [V <velocity>] [MP1 <scan axis middle position>] [MP2 <step axis middle position>] [TT <target type>] [CM <estimation method>] [ST <stop position option>]	Defines a fast alignment area scan routine The current valid definition can be queried with FRR?	Only supported by E-727.AS. Descriptions see E727T0012 user manual.
FRC	FRC <routine name> {<routine name coupled>}	Couples fast alignment routines to each other	
FRC?	FRC? [{<routine name>}]	Gets coupled fast alignment routines	
FRS	FRS {<routine name>}	Starts a fast alignment routine	
FRP	FRP {<routine name> <routine action>}	Stops, pauses or resumes a fast alignment routine	
FRP?	FRP? [{<routine name>}]	Gets the current state of a fast alignment routine	
FRR?	FRR? [<routine name> [<result ID>]]	Gets the results of a fast alignment routine	
FRH?	FRH?	Lists descriptions and physical units for the routine results that can be queried with the FRR? command	
GWD?	GWD? [<StartPoint> <NumberOfPoints> [{<WaveTableID>}]]	Get Wave Table Data	p. 51
HDI?	HDI?	Get Help For Interpretation Of DIA? Response	p. 53
HDR?	HDR?	Get All Data Recorder Options	p. 54
HLP?	HLP?	Get List Of Available Commands	p. 54
HLT	HLT [{<AxisID>}]	Halt Motion Smoothly	p. 55
HPA?	HPA?	Get List Of Available Parameters	p. 56
HPV?	HPV?	Get Parameter Value Description	p. 57
IDN?	IDN?	Get Device Identification	p. 59
IFC	IFC {<InterfacePam> <PamValue>}	Set Interface Parameters Temporarily	p. 59
IFC?	IFC? [{<InterfacePam>}]	Get Current Interface Parameters	p. 60
IFS	IFS <Pswd> {<InterfacePam> <PamValue>}	Set Interface Parameters As Default Values	p. 62
IFS?	IFS? [{<InterfacePam>}]	Get Interface Parameters As Default Values	p. 64
IMP	IMP <AxisID> <Amplitude>	Start Impulse And Response Measurement	p. 65
JOG	JOG {<AxisID> <Velocity>}	Start Motion With Given Velocity	p. 66
JOG?	JOG? [{<AxisID>}]	Get Velocity For Motion Caused By JOG	p. 69
JRC	JRC <Jump> <CMD?> <OP> <Value>	Jump Relatively Depending On Condition	p. 70

Command	Format	Short Description	Details see
MAC	MAC <keyword> {<parameter>} Especially: MAC BEG <macroname> MAC DEF <macroname> MAC DEF? MAC DEL <macroname> MAC END MAC ERR? MAC FREE? MAC NSTART <macroname> <uint> [<String1> [<String2>]] MAC START <macroname> [<String1> [<String2>]]	Call Macro Function	p. 71
MAC?	MAC? [<macroname>]	List Macros	p. 74
MEX	MEX <CMD?> <OP> <Value>	Stop Macro Execution Due To Condition	p. 74
MOV	MOV {<AxisID> <Position>}	Set Target Position	p. 75
MOV?	MOV? [{<AxisID>}]	Get Target Position	p. 77
MVR	MVR {<AxisID> <Distance>}	Set Target Relative To Current Position	p. 77
ONT?	ONT? [{<AxisID>}]	Get On-Target State	p. 79
OVF?	OVF? [{<AxisID>}]	Get Overflow State	p. 80
POS?	POS? [{<AxisID>}]	Get Real Position	p. 81
PUN?	PUN? [{<AxisID>}]	Get Axis Unit	p. 81
RBT	RBT	Reboot System	p. 82
RMC?	RMC?	List Running Macros	p. 82
RPA	RPA [{<ItemID> <PamID>}]	Reset Volatile Memory Parameters	p. 83
RTR	RTR <RecordTableRate>	Set Record Table Rate	p. 84
RTR?	RTR?	Get Record Table Rate	p. 85
SAI?	SAI? [ALL]	Get List Of Current Axis Identifiers	p. 86
SEP	SEP <Pswd> {<ItemID> <PamID> <PamValue>}	Set Nonvolatile Memory Parameters	p. 87
SEP?	SEP? [{<ItemID> <PamID>}]	Get Nonvolatile Memory Parameters	p. 89
SIC	SIC <FA input channel ID> <calculation type> [{<calculation parameter>}]	Defines calculation settings for an analog input channel	Only supported by E-727.AS. Descriptions see E727T0012 user manual.
SIC?	SIC? [{<FA input channel ID>}]	Gets the calculation settings for an analog input channel	
SPA	SPA {<ItemID> <PamID> <PamValue>}	Set Volatile Memory Parameters	p. 90
SPA?	SPA? [{<ItemID> <PamID>}]	Get Volatile Memory Parameters	p. 94
SSN?	SSN?	Get Device Serial Number	p. 95
STE	STE <AxisID> <Amplitude>	Start Step And Response Measurement	p. 95
STP	STP	Stop All Axes	p. 96
SVA	SVA {<AxisID> <Amplitude>}	Set Open-Loop Axis Value	p. 97

Command	Format	Short Description	Details see
SVA?	SVA? [{<AxisID>}]	Get Open-Loop Axis Value	p. 99
SVO	SVO {<AxisID> <ServoState>}	Set Servo Mode	p. 100
SVO?	SVO? [{<AxisID>}]	Get Servo Mode	p. 101
SVR	SVR {<AxisID> <Difference>}	Set Relative Open-Loop Axis Value	p. 102
TAD?	TAD? [{<InputSignalID>}]	Get ADC Value Of Input Signal	p. 103
TAV?	TAV? [{<FA input channel ID>}]	Gets voltage value of an analog input channel	Only supported by E-727.AS. Descriptions see E727T0012 user manual.
TCI?	TCI? [{<FA input channel ID>}]	Gets calculated value of an analog input channel	
TIO?	TIO?	Tell Digital I/O Lines	p. 103
TLT?	TLT?	Get Number of DDL Tables	p. 104
TMN?	TMN? [{<AxisID>}]	Get Minimum Commandable Position	p. 104
TMX?	TMX? [{<AxisID>}]	Get Maximum Commandable Position	p. 105
TNR?	TNR?	Get Number Of Record Tables	p. 105
TNS?	TNS? [{<InputSignalID>}]	Get Normalized Input Signal Value	p. 106
TPC?	TPC?	Get Number of Output Signal Channels	p. 107
TSC?	TSC?	Get Number of Input Signal Channels	p. 107
TSP?	TSP? [{<InputSignalID>}]	Get Input Signal Value	p. 108
TWC	TWC	Clear All Wave Related Triggers	p. 108
TWG?	TWG?	Get Number of Wave Generators	p. 109
TWS	TWS {<TrigOutID> <PointNumber> <Switch>}	Set Trigger Line Action To Waveform Point	p. 109
TWS?	TWS? [<StartPoint> [<NumberOfPoints> [{<TrigOutID>}]]]	Get Trigger Line Action At Waveform Point	p. 111
VAR	VAR <Variable> <String>	Set Variable Value	p. 113
VAR?	VAR? [{<Variable>}]	Get Variable Value	p. 114
VCO	VCO {<AxisID> <VelCtrlState>}	Set Velocity Control Mode	p. 115
VCO?	VCO? [{<AxisID>}]	Get Velocity Control Mode	p. 116
VEL	VEL {<AxisID> <Velocity>}	Set Closed-Loop Velocity	p. 116
VEL?	VEL? [{<AxisID>}]	Get Closed-Loop Velocity	p. 117
VOL?	VOL? [{<OutputSignalID>}]	Get Voltage Of Output Signal Channel	p. 118
WAC	WAC <CMD?> <OP> <Value>	Wait For Condition	p. 119
WAV	WAV <WaveTableID> <AppendWave> <WaveType> <WaveTypeParameters>	Set Waveform Definition	p. 120
WAV?	WAV? [{<WaveTableID> <WaveParameterID>}]	Get Waveform Definition	p. 126
WCL	WCL {<WaveTableID>}	Clear Wave Table Data	p. 126
WGC	WGC {<WaveGenID> <Cycles>}	Set Number Of Wave Generator Cycles	p. 127
WGC?	WGC? [{<WaveGenID>}]	Get Number Of Wave Generator Cycles	p. 127

Command	Format	Short Description	Details see
WGO	WGO {<WaveGenID> <StartMode>}	Set Wave Generator Start/Stop Mode	p. 128
WGO?	WGO? [{<WaveGenID>}]	Get Wave Generator Start/Stop Mode	p. 133
WGR	WGR	Starts Recording In Sync With Wave Generator	p. 134
WOS	WOS {<WaveGenID> <Offset>}	Set Wave Generator Output Offset	p. 134
WOS?	WOS? [{<WaveGenID>}]	Get Wave Generator Output Offset	p. 136
WPA	WPA <Pswd> [{<ItemID> <PamID>}]	Save Parameters To Nonvolatile Memory	p. 137
WSL	WSL {<WaveGenID> <WaveTableID>}	Set Connection Of Wave Table To Wave Generator	p. 139
WSL?	WSL? [{<WaveGenID>}]	Get Connection Of Wave Table To Wave Generator	p. 140
WTR	WTR {<WaveGenID> <WaveTableRate> <InterpolationType>}	Set Wave Generator Table Rate	p. 140
WTR?	WTR? [{<WaveGenID>}]	Get Wave Generator Table Rate	p. 142

2.3 GCS Command Reference (alphabetical)

#5 (Request Motion Status)

Description: Requests motion status of the axes. Only effective in closed-loop operation (servo ON).

Format: #5 (single ASCII character number 5)

Arguments: None

Response: The answer <uint> is bit-mapped and returned as the hexadecimal sum of the following codes:

1=first axis is moving
2=second axis is moving
4=third axis is moving
...

Examples: 0 indicates motion of all axes complete
3 indicates that the first and the second axis are moving

Notes: During an AutoZero procedure (see ATZ command (p. 22)), the motion status can be queried with #5 irrespective of the current operating mode (open-loop or closed-loop control).

#7 (Request Controller Ready Status)

Description:	Asks controller for ready status (tests if controller is ready to perform a new command).
Format:	#7 (single ASCII character number 7)
Arguments:	None
Response:	B1h (ASCII character 177 = "±" in Windows) if controller is ready B0h (ASCII character 176 = "°" in Windows) if controller is not ready (e.g. performing a referencing command)
Troubleshooting:	The response characters may appear differently in non-Western character sets or other operating systems. They may be indistinguishable on the controller screen.

#8 (Query if Macro Is Running)

Description:	Tests if a macro is running on the controller.
Format:	#8
Arguments:	None
Response:	<uint>=0 no macro is running <uint>=1 a macro is currently running

#9 (Get Wave Generator Status)

Description: Requests the status of the wave generator(s).

The #9 single-character command can be used to query the current activation state of the wave generators. The reply shows if a wave generator is running or not, but does not contain any information about the wave generator start mode (e.g. with DDL). With WGO? you can ask for the last-commanded wave generator start options (WGO settings (p. 128)).

Format: #9 (single ASCII character number 9)

Arguments: None

Response: The answer <uint> is bit-mapped and returned as the hexadecimal sum of the following codes:
1 = Wave Generator 1 is running,
2 = Wave Generator 2 is running,
4 = Wave Generator 3 is running, etc.

Examples: 0 indicates that no wave generator is running
5 indicates that wave generators 1 and 3 are running

#24 (Stop All Motion)

Description: Stops all motion abruptly. For details see the notes below.

Sets error code to 10.

This command is identical in function to STP (p. 96), but only one character must be send via the interface. Therefore #24 can also be used while the controller is performing time-consuming tasks.

Format: #24 (ASCII character 24)

Arguments: None

Response: None

Notes: #24 stops motion of all axes. Also stops macro execution.

After the axes are stopped, if servo is on their target positions are set to their current positions, or if servo is off, their open-loop control values are set to their last valid control values.

When the analog input is used as control source and the axis motion is stopped with STP or #24, the behaviour depends on the value of the Discon. Target Man. In With Stop parameter (ID 0x0E001E00): 1 = the analog input channel is disconnected from the axis; 0 = the analog input channel remains connected to the axis. If the analog input channel is disconnected from the axis: To recommence commanding the axis via the analog input, the corresponding input signal channel must be reconnected to the axis. See "How to work with the Analog Input" in the E727T0005 user manual for more information.

The Disable Error 10 parameter (ID 0x0e000301) can be used to avoid that error code 10 is set when axes are stopped with the STP, #24 or HLT commands.
0 = OFF (Error code 10 is set.)
1 = ON (Error code 10 is not set.)

*IDN? (Get Device Identification)

Description: Reports the device identity number.

Format: *IDN?

Arguments: None

Response: One-line string terminated by line feed with controller name, serial number and firmware version

Notes: For the E-727, *IDN? replies something like:

(c)2015 Physik Instrumente (PI) GmbH & Co. KG, E-727, 116031766, 13.21.00.09

*IDN? is identical in function with the IDN? command (p. 59).

ADD (Add and Save to Variable)

Description: Adds two values and saves the result to a variable. See also “Variables” in the E727T0005 user manual.

The variable is present in volatile memory (RAM) only.

Format: ADD <Variable> <FLOAT1> <FLOAT2>

Arguments: <Variable> is the name of the variable to which the result is to be saved.

<FLOAT1> is the first summand.

<FLOAT2> is the second summand.

For the summands, floating point numbers are expected. They can be given directly or via the value of a variable.

Response: None

Notes: Local variables can be set using ADD in macros only.

Example 1: This example only works within macros.

Value $B is added to value $A, and the result is saved to variable C:

ADD C $A $B

Example 2: This example only works within macros.

The name of the variable to which the result is to be copied is given via the value of another variable:

Send: VAR?

Receive:

A=468

B=123

3Z=WORKS

Send: ADD A${3Z} $A $B

Send: VAR?

Receive:

A=468

B=123

AWORKS=591

3Z=WORKS

Send: ADD ${3Z} $A $B

Send: VAR?

Receive:

A=468

B=123

AWORKS=591

WORKS=591

3Z=WORKS

AOS (Set Analog Input Offset)

Description: Set an offset to be added to the analog input scaled value for the given axis (corresponding parameter is Analog Target Offset, ID 0x06000501).

This offset is only effective when an input signal channel of the controller is connected to the axis for control-value generation. The connection can be made via the "ADC Channel for Target" parameter (parameter ID 0x06000500) using SPA (p. 90) or SEP (p. 87).

The control value for an axis which is connected to an input signal channel consists of:
Control Value = Analog Input Scaled Value of the Input Signal Channel + Offset

NOTICE: There is no range check for the given <Offset> value. Make sure that the resulting control value does not exceed the travel range limits of the axis (Range Limit Min, parameter ID 0x70000000 and Range Limit Max, parameter ID 0x70000001).

The AOS command changes the offset setting in volatile memory (RAM) only. On controller power-on or reboot, the offset value is loaded from the controllers non-volatile memory, and any changes made with AOS will be lost unless they have been saved.

To save the currently valid AOS setting to non-volatile memory, where it becomes the power-on default, use WPA (p. 137).

To have write access to the parameter(s), it might be necessary to switch to the proper command level using CCL (p. 26).

Format: AOS {<AxisID> <Offset>}

Arguments	<AxisID> is one axis of the controller
	<Offset> is the offset value, any floating point number. In closed-loop operation (servo ON), the offset is interpreted as position value in either case. In open-loop operation (servo OFF), with the default settings of the output matrix, the offset also corresponds numerically to axis position (see "Output Generation" in the E727T0005 user manual).
Response:	None
Troubleshooting:	Illegal axis identifier
Notes:	See also "Control Value Generation" and "How to work with the Analog Input" in the E727T0005 user manual.
Example:	A E-727.3CDA is used which can control 3 axes and is equipped with 4 analog input lines. Axis 2 of the E-727 is in closed-loop operation (servo on) in this example, and the current target value can be read with the MOV? command. In open-loop operation, you would use SVA? instead to ask for the current open-loop control value.

Send:	CCL 1 advanced
Note:	Switch to command level 1 before you change parameter values with SPA or SEP.
Send:	SPA 2 0x06000500 4
Note:	Select input signal channel 4 (Analog In 1, pins 2 and 9 of the Analog I/O socket) as control source for axis 2. Now the control value of axis 2 will result from the scaled input value of channel 4 plus the offset.
Send	AOS 2 0.0
Note:	Set offset of axis 2 zero.
Send	TSP? 4
Receive	4=3.22
Note:	Request the filtered and scaled value of input signal channel 4. The current value is 3.22. This

value plus the offset is the current target value of axis 2.

Send MOV? 2

Receive 2=3.22

Note: Request the current target position of axis 2. The target position and the scaled value of input signal channel 4 are the same because the offset is zero.

Send AOS 2 1.50

Note: Set offset of axis 2 to 1.5.

Send TSP? 4

Receive 4=3.22

Send MOV? 2

Receive 2=4.72

Note: The target value of axis 2 is the scaled value of input signal channel 4 plus the offset of axis 2.

Send MOV 2 6.0

Send ERR?

Receive 72

Note: As long as the control value of axis 2 is given by an analog input, it is not possible to set the target using the MOV command.

Send: SPA 2 0x06000500 0

Note: Disconnect any analog input from axis 2. Now its target position can be set by the MOV command. The AOS setting is no longer effective for the control value generation of axis 2.

AOS? (Get Analog Input Offset)

Description: Get currently valid offset to the analog input scaled value for the given axis (Analog Target Offset parameter value in volatile memory (ID 0x06000501)).

Get all axes when <AxisID>=""

Format: AOS? [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller

Response: {<AxisID>"="<Offset> LF}

where

<Offset> is the offset value, see AOS (p. 19) for details

Troubleshooting: Illegal axis identifier

ATZ (Set Automatic Zero Point Adjustment)

Description: Automatic zero-point adjustment. Sets the output voltage which is to be applied at the zero position of the axis and starts an appropriate adjustment procedure.

The adjustment procdure can be cancelled by #24 (p. 15) or STP (p. 96). After the cancellation, the last output voltage set by the procedure remains in effect.

ATZ works in open-loop operation (servo off). If the servo is on, it will be switched off automatically at the start of the ATZ procedure and switched on again when it is finished.

The AutoZero procedure has the highest priority, i.e. it will overwrite the control values given by all other sources. When the analog control input is enabled, it will be disabled automatically at the start of the AutoZero procedure and reenabled again when AutoZero is finished.

ATZ is not effective on non-linear axes (rotation axes).

The success of the automatic zero-point adjustment can be queried with the ATZ? command (p. 25).

The automatic zero-point adjustment can take several seconds. During this time, the controller is busy and only very limited able to execute or answer commands.

Format: ATZ [{<AxisID> <LowVoltage>}]

Arguments <AxisID> is one axis of the controller

<LowVoltage> gives the voltage value to be applied at the zero position of the axis; in volts; float.
Can also be NaN ("not a number")—in this case the value of the Autozero Low Voltage parameter saved in the controller (ID 0x07000A00) will be used.

If all arguments are omitted, ATZ will be carried out for all linear axes using their AutoZero Low Voltage parameter values.

Response: None

Troubleshooting: ATZ will be not successful when an invalid axis identifier is used, e.g.

ATZ 9 NAN

or when NaN was omitted and no voltage value was given

Notes: NOTICE:

The AutoZero procedure will move the axis, and the motion may cover the whole travel range. Make sure that it is safe for the stage to move.

The AutoZero procedure changes the mechanical zero position of the axis. If you need an absolute zero point that changes only within the limits of sensor accuracy during the life of the system:

- Run the AutoZero procedure a maximum of once. Afterwards save the values of the parameters ***Sensor Mech. Correction 1*** (ID 0x02000200) and ***Sensor Offset factor*** (ID 0x02000102) to nonvolatile memory.
- Repeat the AutoZero procedure and the subsequent saving of the parameters only in case your system has been recalibrated.

Procedure details:

To match voltage and position as required, the axis is moved—the motion range is specified by the <LowVoltage> value given in the ATZ command (lower limit) and by the Autozero High Voltage parameter value saved in the controller (parameter ID 0x07000A01; upper limit). The final position is the zero position, with the given <LowVoltage> value applied.

There is no range check for the given <LowVoltage> value. Make sure that this value does not exceed the voltage limits of the amplifier(s) (Min Output Voltage of Amplifier, parameter ID 0x0B000007 and Max Output Voltage of Amplifier, parameter ID 0x0B000008). Otherwise the <LowVoltage> value will be set to the corresponding limit.

If NaN is entered for the <LowVoltage> value, the AutoZero Low Voltage parameter value saved in the controller will be used (parameter ID 0x07000A00). You can modify this parameter with SPA (p. 90) or SEP (p. 87).

The AutoZero procedure changes the values of the parameters Sensor Mech. Correction 1 (ID 0x02000200). With the models for piezoresistive sensors and strain gauge sensors, the AutoZero procedure also changes the Sensor Offset factor (ID 0x02000102) parameters.

To save the current valid values of the above-mentioned parameters to non-volatile memory, where they become the power-on defaults, use WPA (p. 137). To have write access to the parameters, it might be necessary to switch to a higher command level using CCL (p. 26).

See also "AutoZero Procedure" in the E727T0005 user manual.

Sensor autoscaling can be included in the AutoZero procedure, if necessary. For more information, see "Special Function: Sensor Autoscaling" in the E727T0005 user manual.

Example 1:	Send:	SEP? 1 0x07000A00
	Receive:	1 0x7000a00=0.000000e+00
	Note:	The value of the AutoZero Low Voltage parameter saved in the controller is 0 V.

	Send:	ATZ 1 NaN
	Note:	Starts autozero for axis 1 with the value of the AutoZero Low Voltage parameter. Do not omit "NaN"!
	Send:	ATZ? 1
	Receive:	1
	Note:	Autozero for axis 1 was successful
Example 2:	Send:	ATZ 1 15.0
	Note:	Starts autozero for axis 1 with a voltage value of 15 V
	Send:	ATZ? 1
	Receive:	0
	Note:	Autozero for axis 1 was not successful

ATZ? (Get Automatic Zero Point Calibration)

Description:	Query success or failure of the automatic zero-point calibration (see ATZ (p. 22) for details).
Format:	ATZ? [{<AxisID>}]
Arguments	<AxisID> is one axis of the controller
Response:	{<AxisID>"="<uint> LF} where <uint> indicates whether the automatic zero-point calibration of the given axis was successful (=1) or not (=0).
Troubleshooting:	Illegal axis identifier

CCL (Set Command Level)

Description:	Changes the active "command level" and determines thus the availability of commands and of write access to system parameters.
Format:	CCL <Level> [<PSWD>]
Arguments:	<Level> is one command level of the controller <PSWD> is the password required for changing to the appropriate command level The following command levels and passwords are valid: Level = 0 is the default setting, all commands provided for "normal" users are available, read access to all parameters, no password required. Level = 1 adds additional commands and write access to level-1 parameters (commands and parameters from level 0 are included). The required password is "advanced". Level > 1 is provided for PI service personnel only. Users cannot change to a level > 1. Contact your Physik Instrumente Sales Engineer or write info@pi.ws if there seem to be problems with level 2 or higher parameters.
Response:	None
Troubleshooting:	Invalid password
Notes:	HLP? (p. 54) lists all commands available in the current command level. HPA? (p. 56) lists the parameters including the information about which command level allows write access to them. For more information about parameter handling see "Parameters" in the E727T0005 user manual. After controller power-on or reboot, the active command level is always Level 0.

CCL? (Get Command Level)

Description:	Get the active "command level".
Format:	CCL?
Arguments:	None
Response:	<Level> is the currently active command level; uint.
Notes:	<Level> should be 0 or 1. <Level> = 0 is the default setting, all commands provided for "normal" users are available, as is read access to all parameters <Level> = 1 provides additional commands and write access to level-1 parameters (commands and parameters from Level 0 are included)

CPY (Copy Into Variable)

Description:	Copies a command response to a variable. See also “Variables” in the E727T0005 user manual. The variable is present in volatile memory (RAM) only.
Format:	CPY <Variable> <CMD?>
Arguments:	<Variable> is the name of the variable to which the command response is to be copied. <CMD?> is one query command in its usual notation. The response has to be a single value and not more.
Response:	None
Notes:	Local variables can be set using CPY in macros only.

Example: It is possible to copy the value of one variable (e. g. SOURCE) to another variable (e. g. TARGET):

CPY TARGET VAR? SOURCE

CST? (Get Stage Type Of Selected Axis)

Description: Returns the name of the connected stage for the queried axis.

Format: CST? [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller

Response: {<AxisID>"="<string> LF}

where

<string> is the name of the stage assigned to the axis.

Notes: The stage name is read from the Stage Type parameter (ID 0x0F000100). Normally, the value of this parameter is written during the calibration at the factory or when a stage with ID-chip is connected.

You can change the parameter value using SPA (p. 90) or SEP (p. 87).

If the parameter should be empty, "Default_Stage_Z" will be returned.

CSV? (Get Current Syntax Version)

Description: Get current GCS syntax version used in the firmware.

Format: CSV?

Arguments: None

Response: The current GCS syntax version, can be 1.0 (for GCS 1.0) or 2.0 (for GCS 2.0)

CTO (Set Configuration of Trigger Output)

Description: Configures the trigger output conditions for the given digital output line.

The trigger output conditions will become active immediately.

Format: CTO {<TrigOutID> <CTOPam> <Value>}

Arguments: <TrigOutID> is one digital output line of the controller, see below for details

<CTOPam> is the CTO parameter ID in decimal format, see below for the available IDs

<Value> is the value to which the CTO parameter is set, see below

Response: None

Available output lines and trigger conditions: <TrigOutID> corresponds to the output lines OUT1 to OUT3, IDs = 1 to 3; see "Digital I/O Socket" (p. 164).

<CTOPam> parameter IDs available for E-727:

1 = TriggerStep
2 = Axis
3 = TriggerMode
5 = MinThreshold
6 = MaxThreshold
7 = Polarity
8 = Start Threshold
9 = Stop Threshold
16 = Trigger Out Mask

<Value> available for the appropriate <CTOPam> ID:

for TriggerStep: step size in physical units (default value is 0.1)

for Axis: the axis to connect to the trigger output line. By default, axis 1 is connected to line 1, axis 2 to line 2, and axis 3 to line 3.

for TriggerMode (default value is 2):

0 = Position Distance; with this TriggerMode, a trigger pulse is written whenever the axis has covered the TriggerStep distance (<CTOPam> ID 1). Optionally, values for StartThreshold and StopThreshold (<CTOPam> IDs 8 and 9) can be defined to enable the trigger output for a limited position range and a certain direction of motion only (negative or positive). When StartThreshold and StopThreshold are set to the same value, they will not be used. For further options which cannot be configured with CTO but only via parameters, see "Example—"Position Distance" Trigger Mode" in the E727T0005 user manual.

2 = OnTarget; with this TriggerMode, the on-target status of the selected axis is written to the selected trigger output line (this status can also be read with the ONT? command)

3 = MinMaxThreshold; with this TriggerMode, values for MinThreshold and MaxThreshold (<CTOPam> IDs 5 and 6) must be defined. When the axis position of the selected axis is inside the band specified by the MinThreshold and MaxThreshold values, the selected trigger output line is set high, otherwise it is set low.

4 = Generator Level Trigger; with this TriggerMode, the trigger line action must be defined with TWS (p. 109). The length of a single trigger pulse is the same as the duration of one servo cycle. If the signal level is set to

HIGH with TWS for consecutive points of a wave table, the signal level therefore does not change back to LOW between the points.

9 = Generator Pulse Trigger; with this TriggerMode, the trigger line action must be defined with TWS (p. 109). A single trigger pulse is shorter than the servo cycle duration. If the signal level is set to HIGH with TWS for consecutive points of a wave table, the signal level therefore changes back to LOW after each point. This way, the trigger output can be used to count the waveform points that are output by the wave generator.

14 = TriggerOutAND; with this TriggerMode, the digital output line <TrigOutID> outputs the signal states of the output lines selected with TriggerOutMask (<CTOPam> ID 16). The states of the selected lines are combined via AND bit operation.

15 = TriggerOutOR; with this TriggerMode, the digital output line <TrigOutID> outputs the signal states of the output lines selected with TriggerOutMask (<CTOPam> ID 16). The states of the selected lines are combined via OR bit operation.

for MinThreshold/MaxThreshold: position value in physical units; used for the MinMaxThreshold trigger mode; both values must be set to form a band (no default values)

for Polarity: Sets the signal polarity for the digital output line
0 = Active Low
1 = Active High

for StartThreshold/StopThreshold: Position value; can be used for the PositionDistance trigger mode; both thresholds must be set to determine the position range and the direction of motion for the trigger output

for TriggerOutMask: Bit-mapped mask used for the trigger modes TriggerOutAND and TriggerOutOR. The mask selects the digital output lines whose signal states are to be logically combined:
bit 0 = digital output line 1
bit 1 = digital output line 2
bit 2 = digital output line 3
The mask can be specified in hex or decimal format.

For further application examples and details, see "Configuring Trigger Output" and "Trigger Output Synchronized with Wave Generator" in the E727T0005 user manual.

Example 1: A pulse on the digital output line OUT1 (ID 1) is to be generated whenever the axis 2 has covered a distance of 0.05 µm. The following parameters must be set:

TrigOutID = 1

Axis = 2

TriggerMode = 0

TriggerStep = 0.05

Send: CTO 1 2 2 1 3 0 1 1 0.05

Example 2: The digital output line OUT2 (ID 2) is set to the trigger mode TriggerOutAND (14). In this mode, line 2 outputs the result of a logical AND operation which combines the states of digital output lines 1 and 3. The first part of the CTO command selects the trigger mode (2 3 14), the second part selects the digital output lines to be combined (2 16 0x5).

Send: CTO 2 3 14 2 16 0x5

Notes: CTO changes the values of the following parameters in volatile memory:

Parameter ID	Corresponding <CTOPam>
0x18000201	1 = Trigger Step
0x18000202	2 = Axis
0x18000203	3 = Trigger Mode
0x18000205	5 = Min.Threshold
0x18000206	6 = Max.Threshold
0x18000207	7 = Polarity
0x18000208	8 = Start Threshold
0x18000209	9 = Stop Threshold
0x18000210	16 = Trigger Out Mask

You can also change these parameters using SPA (volatile memory) or SEP (non-volatile memory). Furthermore, you can use WPA to copy the current values from volatile memory to non-volatile memory, where they become the power-on defaults. When using SPA, SEP or WPA, it is necessary to switch to command level 1 with CCL to have write access to the parameters. To read the parameter values, you can also query with the SPA? or SEP? commands.

For the “Position Distance” trigger mode, settings for pulse length definition and reduction of false triggers can be made. These settings are only available via parameters but not via the CTO command. For further details, see “Example—"Position Distance" Trigger Mode” in the E727T0005 user manual.

CTO? (Get Configuration of Trigger Output)

Description: Replies with the values set for specified trigger output lines and parameters

Format: CTO? [{<TrigOutID> <CTOPam>}]

Arguments: <TrigOutID>: is one digital output line of the controller; see CTO

<CTOPam>: parameter ID; see CTO

If all arguments are omitted, the values for all parameters are given for all output lines.

Response: One or more lines of the following format:

<TrigOutID> <CTOPam>"="<Value>

For <Value> see CTO.

DDL (Set DDL Table Value(s))

Description: Dynamic Digital Linearization (DDL) data load: writes data to the given DDL table.

NOTICE: Write the correct number of points to the DDL table. It must be equal to the length of the waveform which is output with the "Use DDL" option (see WGO (p. 128)) for the corresponding axis.

The DDL command will stop a running DDL initialization process.

The DDL table content will be lost when the controller is powered down or rebooted.

Format: DDL <DDLtableID> <StartPoint> {<ValueN>}

Arguments:	<DDLtableID> is one DDL table of the controller, see below for details <StartPoint> is the start point in the DDL table, starts with index 1 <ValueN> is the value of point n
Response:	None
Troubleshooting:	Not enough memory space available: delete the content of DDL tables which are not used. See the DTC command (p. 49).
Notes:	The number of DDL tables present in the E-727 is the same as the number of logical axes, and each DDL table is dedicated to one axis. For a detailed description of the DDL feature see "Dynamic Digital Linearization (DDL)" in the E727T0005 user manual.
Example:	Send: DDL 1 10 2 4 6 8 10 12 14 16 Note: The values 2, 4, 6, 8, ... are written to DDL table 1, starting with the 10th point in the table

DDL? (Get DDL Table Value(s))

Description:	Dynamic Digital Linearization (DDL) data query: Gets the DDL data from the specified DDL table. Only tables with the same length can be read in the same command line. Because DDL tables do not have a common length, use the DTL? (p. 49) command to read the table length before reading the table data.
Format:	DDL? [<StartPoint> [<NumberOfPoints> [{<DDLtableID>}]]]

Arguments: <StartPoint> is the start point in the DDL table, starts with index 1

<NumberOfPoints> is the number of points to be read per table

<DDLtableID> is one DDL table of the controller, see below for details

Response: The DDL data as GCS array, see the separate manual for the GCS array, SM 146E, and the example below

Troubleshooting: The DDL tables to be read with the same DDL command line have different lengths

Note: The number of DDL tables present in the E-727 is the same as the number of logical axes, and each DDL table is dedicated to one axis. The number of points in a DDL table corresponds to the length of the waveform which was output during the DDL initialization.

For a detailed description of the DDL feature see "Dynamic Digital Linearization (DDL)" in the E727T0005 user manual.

Example:

```
ddl? 1 20 1
# TYPE = 1
# SEPARATOR = 9
# DIM = 1
# SAMPLE_TIME = 0.000050
# NDATA = 20
# NAME0 = DDL of axis1
# END_HEADER
-79.720367
-79.714294
-79.713760
-79.711037
-79.707939
-79.702576
-79.701523
-79.698792
```

```
-79.693787
-79.693268
-79.689827
-79.692619
-79.689949
-79.685356
-79.687393
-79.682693
-79.682991
-79.679008
-79.683807
-79.684433
```

DEL (Delay the Command Interpreter)

Description:	Delays <uint> milliseconds.
Format:	DEL <uint>
Arguments:	<uint> is the delay value in milliseconds.
Response:	None
Notes:	DEL can only be used in macros. Do not mistake MAC DEL (deletes macros) for DEL (delays). See the MAC command (p. 71) and "Controller Macros" in the E727T0005 user manual for more information.

DIA? (Get Diagnosis Information)

Description:	Gets the current value of a specified measurand. If all arguments are omitted, the current value of all measurands is queried.
Format:	DIA? [{<MeasureID>}]

Arguments: <MeasureID> is the identifier of one measurand, see below for details.

Response: {<MeasureID>"="<MeasuredValue> LF}

where

<MeasuredValue> gives the current value of the measurand, see below for details.

Notes: Use the response to HDI? (p. 53) to get descriptions and physical units of the supported measurands.

E-727 supports the following measurands:

<MeasureID>	<Description> (get with HDI?)	Possible values of <MeasuredValue>
1	Temperature Status	0 = temperature of amplifier exceeds no threshold 1 = temperature of amplifier exceeds threshold 1 (“alert” threshold) – only with E-727.xxxP and .xxxAP models for higher output current 2 = temperature of amplifier exceeds threshold 2 (“switch-off” threshold)
2	Amplifier Output Status (On/Off)	1 = ON: amplifier output is active 0 = OFF: amplifier output is not active (temperature of amplifier exceeds switch-off threshold

Examples: No temperature threshold is exceeded:

Send: DIA?
Receive: 1=0
2=1

Temperature threshold 1 (“alert” threshold) is exceeded:

Send: DIA?
Receive: 1=1
2=1

Temperature threshold 2 (“switch-off” threshold) is exceeded, and the amplifier output has been switched

off automatically for that reason:
Send: DIA?
Receive: 1=2
2=0

For possible measures in case of exceeding a threshold see “Overtemp Protection” in the E727T0005 user manual.

DIO? (Get Digital Input Lines)

Description: Gets the states of the specified digital input lines.

Use TIO? (p. 103) to get the number of available digital I/O lines.

Format: DIO? [{<DIOID>}]

Arguments: <DIOID> is the identifier of the digital input line, see below for details.

Response: {<DIOID>"="<InputOn> LF}

where

<InputOn> gives the state of the digital input line, see below for details.

Notes: You can use the DIO? command to directly read the digital input lines 1 to 4 that are located on the Digital I/O socket (p. 164).

The <DIOID> identifiers to use for the lines are 1 to 4. If the identifier is omitted, all lines are queried.

If <InputOn>=0, the digital input is LOW/OFF; if <InputOn>=1, the digital input is HIGH/ON.

DPO (DDL Parameter Optimization)

Description: Dynamic Digital Linearization (DDL) Parameter Optimization. Recalculates the internal DDL processing parameters (Time Delay Max, ID 0x14000006, Time Delay Min, ID 0x14000007).

DPO usage is required when the servo parameters (notch filter frequency, servo-loop P-term, servo-loop I-term and servo-loop slew rate) have changed for an axis.

The DPO command changes the processing parameters in volatile memory (RAM) only. On controller power-on or reboot, the parameter values are loaded from the controllers non-volatile memory, and any changes made with DPO will be lost unless they have been saved.

To save the currently valid processing parameters to non-volatile memory, where they become the power-on default, use WPA (p. 137).

To have write access to the parameters, it might be necessary to switch to a higher command level using CCL (p. 26).

For a detailed description of the DDL feature see "Dynamic Digital Linearization (DDL)" in the E727T0005 user manual.

Format: DPO [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller

Response: None

DRC (Set Data Recorder Configuration)

Description: Set data recorder configuration: determines the data source and the kind of data (RecordOption) used for the given data recorder table.

Format: DRC <RecTableID> <Source> <RecOption>

Arguments: <RecTableID>: is one data recorder table of the controller, see below

<Source>: is the data source, for example an axis, output signal channel, input signal channel, digital input or digital output of the controller. The required source depends on the selected record option.

<RecOption>: is the kind of data to be recorded (record option).

See below for a list of the available record options and the corresponding data sources.

Response: None

Notes: The number of available data recorder tables can be read with TNR? (p. 105). The answer gives the value of the Data Recorder Chan Number parameter, ID 0x16000300. Using SPA (p. 90) or SEP (p. 87) you can change the parameter value in the range of 1 to 8 to increase or decrease the number of data recorder tables.

The total number of points available for data recording is 262144 (Data Recorder Max Points, ID 0x16000200). These points are allocated in equal shares to the available tables (i.e. to the number of tables given in the answer to TNR?).

With HDR? (p. 54) you will obtain a list of available record options and information about additional parameters and commands concerned with data recording.

For detailed information see "Data Recording" in the E727T0005 user manual.

Record options for the appropriate data sources:	<Source>	<RecOption>
	Axis	1 = Target Position of axis (i.e. target value in closed-loop operation), corresponds to the MOV? response
		2 = Current Position of axis, corresponds to the POS? response
		3 = Position Error of axis
		13 = DDL Output of axis (DDL table values), corresponds to the DDL? response
		14 = Open Loop Control of axis (i.e. open-loop control value), corresponds to the SVA? response
		15 = Control Output of axis (before the Axis-to-OutputSignalChannel transformation)
		22 = Slowed Target of axis (in closed-loop operation), target position after slew rate limitation
		23 = Target velocity of axis, not relevant for E-727
		24 = Target acceleration of axis, not relevant for E-727
		25 = Target jerk of axis, not relevant for E-727

Output Signal Channel	7 = Control Voltage of output signal channel (after the Axis-to-OutputSignalChannel transformation but before the output type definition as axis position or piezo driving voltage)
	16 = Voltage of output signal channel (after the Axis-to-OutputSignalChannel transformation and the output type definition, can be axis position or piezo drive voltage), corresponds to the VOL? response
Input Signal Channel	17 = Sensor Normalized of input signal channel, corresponds to the TNS? response
	18 = Input signal channel, after sensor filtering
	19 = Input signal channel, after sensor electronics linearization
	20 = Input signal channel, after sensor mechanics linearization, corresponds to the TSP? response
Digital Input	26 = Value of Digital Input. Hexadecimal sum all digital inputs (binary coded): DigIn = In1 * 1 + In2 * 2 + In3 * 4 + In4 * 8
Digital Output	27 = Value of Digital Output. Hexadecimal sum all digital outputs (binary coded): DigOut = Out1 * 2 + Out2 * 4 + Out3 * 8

For each data recorder table, the configuration can be saved via the DRC Data Source (ID 0x16000700) and DRC Record Option (ID 0x16000701) parameters. DRC changes the values of these parameters in volatile memory, and WPA can be used to save the values permanently.

See "Control Value Generation" and "Output Generation" in the E727T0005 user manual for more information on the signals.

Example: Send: DRC 4 1 2

to record the current position of axis 1 in record table 4.

DRC? (get Data Recorder Configuration)

Description: Returns settings made with DRC (p. 41).

Format: DRC? [{<RecTableID>}]

Arguments: <RecTableID>: is one data recorder table of the controller; if omitted settings for all tables are given.

Response: The current DRC settings:

{<RecTableID>"="<Source> <RecOption> LF}

where

<Source>: is the data source, for example an axis, output signal channel, input signal channel, digital input or digital output of the controller. The source type depends on the record option.

<RecOption>: is the kind of data to be recorded

See DRC for a list of the available record options and the corresponding data sources.

DRL? (Get Number of Recorded Points)

Description:	Reads the number of points comprised by the last recording.
Format:	DRL? [{<RecTableID>}]
Arguments:	<RecTableID> is one data recorder table of the controller
Response:	{<RecTableID>"="<uint> LF} where <uint> gives the number of points recorded with the last recording
Notes:	The number of points is reset to zero for a data recorder table when changing its configuration with DRC.

DRR? (Get Recorded Data Values)

Description:	Reading of the last recorded Data Set. Reading can take some time depending on the number of points to be read! It is possible to read the data while recording is still in progress.
Format:	DRR? [<StartPoint> [<NumberOfPoints> [{<RecTableID>}]]]
Arguments:	<StartPoint>: is the start point in the data recorder table, starts with index 1 <NumberOfPoints>: is the number of points to be read per table <RecTableID>: is one data recorder table of the controller

Response: The recorded data in GCS array format, see the separate manual for GCS array, SM 146E, and the example below

Notes: If <RecTableID> is omitted, the data from all available tables will be read.

With HDR? (p. 54) you will obtain a list of available record options and trigger options and information about additional parameters and commands concerned with data recording.

For detailed information see "Data Recording" in the E727T0005 user manual.

Example:

```
drr? 1 10 1 2 3 4
# TYPE = 1
# SEPARATOR = 9
# DIM = 4
# SAMPLE_TIME = 0.000050
# NDATA = 10
# NAME0 = Current Position of axis1
# NAME1 = Target Position of axis1
# NAME2 = Position Error of axis1
# NAME3 = DDL Output of axis1
# END_HEADER
2.596565        0.000010        -2.596565       -79.720367
2.597346        0.000000        -2.597346       -79.714294
2.597455        0.000010        -2.597455       -79.713760
2.597178        0.000039        -2.597178       -79.711037
2.597124        0.000089        -2.597124       -79.707939
2.597249        0.000158        -2.597249       -79.702576
2.596964        0.000247        -2.596964       -79.701523
2.597389        0.000355        -2.597389       -79.698792
2.596945        0.000484        -2.596945       -79.693787
2.597482        0.000632        -2.597482       -79.693268
```

DRT (Set Data Recorder Trigger Source)

Description:	Defines a trigger source for the given data recorder table.
Format:	DRT <RecTableID> <TriggerSource> <Value>
Arguments:	<RecTableID> is one data recorder table of the controller. See below for details. <TriggerSource> ID of the trigger source, see below for a list of available options <Value> depends on the trigger source, can be a dummy, see below.
Response:	None
Notes:	The number of available data recorder tables can be read with TNR? (p. 105). The answer gives the value of the Data Recorder Chan Number parameter, ID 0x16000300. Using SPA (p. 90) or SEP (p. 87) you can change the parameter value in the range of 1 to 8 to increase or decrease the number of data recorder tables. At present, the specified trigger source is always set for all data recorder tables, irrespective of the <RecTableID> value given in the DRT command. With HDR? (p. 54) you will obtain a list of available record options and trigger options and additional information about data recording. For detailed information see "Data Recording" in the E727T0005 user manual.
Available trigger options:	0 = default setting; data recording is triggered with IMP (p. 65), STE (p. 95), WGO (p. 133), WGR (p. 134); <Value> must be a dummy 1 = any command changing target position or open-loop control value (MVR (p. 77), MOV (p. 75), SVA (p. 97), SVR (p. 102); in addition to IMP, STE, WGO, WGR);

<Value> must be a dummy

3 = external trigger; <Value> gives the ID of the digital input line to be used for trigger input; if 0, any digital input line is used (see “Digital I/O Socket”, p. 164 for available lines).
For reliable triggering, the pulse width of the input signal has to be at least 2 x the servo update time of the E-727. The servo update time is given in seconds by parameter 0x0E000200.

4 = immediately (means that the DRT command itself triggers); <Value> must be a dummy

DRT? (Get Data Recorder Trigger Source)

Description: Gets the trigger source for the data recorder tables.

Format: DRT? [{<RecTableID>}]

Arguments: <RecTableID> is one data recorder table of the controller

Response: {<RecTableID>"="<TriggerSource> <Value> LF}

where

<TriggerSource> is the identifier of the trigger source.

<Value> depends on the trigger source; 0 is a dummy.

Further information is found in the description of the DRT command (p. 47).

DTC (Clears DDL Table Data)

Description: Clears the given DDL table.

DDL table content is also deleted when new DDL data is written to the table during an initialization process (WGO (p. 128) with "Use and reinitialize DDL" start option) or with the DDL command (p. 34). But only DTC marks DDL tables as "free" so that their memory space can be reallocated. Before new DDL data are written, it is therefore recommended to apply DTC to tables whose content is no longer used. This will avoid error messages during the next write operation.

The DTC command also stops a running DDL initialization process.

For a detailed description of the DDL feature see "Dynamic Digital Linearization (DDL)" in the E727T0005 user manual.

Format: DTC {<DDLtableID>}

Arguments: <DDLtableID> is one DDL table of the controller

Response: None

DTL? (Get DDL Table Length)

Description: Get Dynamic Digital Linearization (DDL) table length.

The table length should be read before reading data with the DDL? (p. 35) command.

For a detailed description of the DDL feature see "Dynamic Digital Linearization (DDL)" in the E727T0005 user manual.

Format: DTL? [{<DDLtableID>}]

Arguments: <DDLtableID> is one DDL table of the controller

Response: {<DDLtableID>"="<DDLTableLength> LF}

where

<DDLTableLength> is the length of the table in number of points

ERR? (Get Error Number)

Description: Get error code <int> of the last occurred error and reset the error to 0.

Only the last error is buffered. Therefore you should call ERR? after each command.

The error codes and their descriptions are fully listed in "Error Codes" (p. 160).

Format: ERR?

Arguments: None

Response: The error code of the last occurred error (int).

Troubleshooting: Communication breakdown

GWD? (Get Wave Table Data)

Description: Query waveform shape for given wave table.

Depending on the waveform definition with WAV (p. 120), the wave tables may have different lengths. Due to the GCS array response format definition, it is not possible to read from tables of different lengths with one command line.

The response to GWD? does not contain any offset to the wave generator output set with WOS (p. 134).

Format: If the length of the wave tables differs, only tables with identical length can be read with the same command:

GWD? <StartPoint> <NumberOfPoints> {<WaveTableID>}

If *all* wave tables have the same length, arguments are optional as follows:

GWD? [<StartPoint> [<NumberOfPoints> [{<WaveTableID>}]]]

Arguments: <StartPoint> is the start point in the wave table, starts with index 1

<NumberOfPoints> is the number of points to be read per table

<WaveTableID> is one wave table of the controller; all specified wave tables must have the same length

Response: The wave table contents (waveform) in GCS array format (see the separate manual for the GCS array, SM 146E, and the example below)

Example:

```
gwd? 1 10 1 2 3
# TYPE = 1
# SEPARATOR = 9
# DIM = 3
# SAMPLE_TIME = 0.000050
```

```
# NDATA = 10
# NAME0 = Wave Table1
# NAME1 = Wave Table2
# NAME2 = Wave Table3
# END_HEADER
0.000010        2.000000        0.000000
0.000000        2.000000        0.000000
0.000010        2.000000        0.000000
0.000039        2.000000        0.000001
0.000089        2.000000        0.000005
0.000158        2.000000        0.000012
0.000247        2.000000        0.000023
0.000355        2.000000        0.000039
0.000484        2.000000        0.000063
0.000632        2.000000        0.000093
```

HDI? (Get Help For Interpretation Of DIA? Response)

Description:	Lists descriptions and physical units for the measurands that can be queried with the DIA? command (p. 37).
Format:	HDI?
Arguments:	None
Response:	{<MeasureID>"="<Description>TAB<PhysUnit> LF} where <MeasureID> is the identifier of the measurand. <Description> is the name of the measurand. <PhysUnit> is the physical unit of the measurand.

Notes:	With E-727, the response to HDI? is as follows:
	
	`HDI?`
	`1=Temperature Threshold Exceeded  NO_UNIT`
	`2=Amplifier Output Status (On/Off)  NO_UNIT`
	`end of help`
	
	The measurands that can be queried with E-727 have no physical unit.
	
	See DIA? for possible values of the measurands.

HDR? (Get All Data Recorder Options)

Description: List a help string which contains all information available about data recording (record options and trigger options, information about additional parameters and commands concerned with data recording).

Format: HDR?

Arguments: None

Response

#RecordOptions

{<RecordOption>"="<DescriptionString>[of <Channel>]}

#TriggerOptions

[{<TriggerOption>"="<DescriptionString>}]

#Parameters to be set with SPA

[{<ParameterID>“=“<DescriptionString>}]

#Additional information

<[{<Command description>"("<Command>")"}]

<end of help

Note: In the HDR? response, TriggerOptions = 0 (default) means that recording is triggered by the IMP (p. 65), STE (p. 95), WGO (p. 128) and WGR (p. 134) commands

HLP? (Get List Of Available Commands)

Description: List a help string which contains all commands available.

Format: HLP?

Arguments: None

Response: List of commands available

Troubleshooting:	Communication breakdown
Notes:	The HLP? response contains the commands provided by the current command level. See CCL (p. 26) for more information.

HLT (Halt Motion Smoothly)

Description:	Halts the motion of given axes smoothly. For details see the notes below. Error code 10 is set. #24 and STP in contrast abort current motion as fast as possible for the controller without taking care of a given deceleration.
Format:	HLT [{<AxisID>}]
Arguments:	<AxisID>: is one axis of the controller, if omitted all axes are halted
Response:	None
Troubleshooting:	Illegal axis identifier
Notes:	HLT stops motion of all axes caused by move commands (MOV, MVR, SVA, SVR). After the axes have been stopped, if servo is on their target positions are set to their current positions, or if servo is off, their open-loop control values are set to their last valid control values. The Disable Error 10 parameter (ID 0x0e000301) can be used to avoid that error code 10 is set when axes are stopped with the STP, #24 or HLT commands. 0 = OFF (Error code 10 is set.) 1 = ON (Error code 10 is not set.)

HPA? (Get List Of Available Parameters)

Description: Responds with a help string which contains all available parameters with short descriptions. See "Parameters" in the E727T0005 user manual for further details.

The listed parameters can be changed and/or saved using the following commands:

SPA (p. 90) affects the parameter settings in volatile memory (RAM).

WPA (p. 137) copies parameter settings from RAM to non-volatile memory.

SEP (p. 87) writes parameter settings directly into non-volatile memory (without changing RAM settings).

RPA (p. 83) resets RAM to the values from non-volatile memory.

Format: HPA?

Arguments: None

Response {<PamID>"="<string> LF}

where

<PamID> is the ID of one parameter, hexadecimal format

<string> is a string which describes the corresponding parameter.

The string has following format:

<CmdLevel>TAB<MaxItem>TAB<DataType>TAB<FunctionGroupDescription>TAB<ParameterDescription>[{TAB<PossibleValue>"="<ValueDescription>}]

where

<CmdLevel> is the command level which allows write access to the parameter value

<MaxItem> is the maximum number of items of the same type which are affected by the parameter (the meaning of "item" depends on the parameter, can be axis, output signal channel, input signal channel, data recorder table, digital output line, the whole system, or firmware units)

<DataType> is the data type of the parameter value, can be INT, FLOAT or CHAR

<FunctionGroupDescription> is the name of the function group to which the parameter belongs (parameters are grouped according to their purpose to clarify their interrelation)

<ParameterDescription> is the parameter name

<PossibleValue> is one value from the allowed data range

<ValueDescription> is the meaning of the corresponding value

HPV? (Get Parameter Value Description)

Description: Responds with a help string which contains possible parameters values. Use HPA? instead to get a help string which contains all available parameters with short descriptions.

Format: HPV?

Arguments: None

Response: <string>

<string> has the following format:

“#Possible parameter values are:

{<PamID> <ItemID> “=” <ListType>

[{TAB <PossibleValue> “=” <ValueDescription>}] }

#CCL levels are:

{<PamID> <ItemID> “=”<CmdLevel> }

end of help”

where

<PamID> is the ID of one parameter, hexadecimal format

<ItemID> is one item of the controller (see HPA?), if item=0 the description is valid for all items

<ListType> determines how the possible parameter values listed in the string have to be interpreted:

0 = parameter not applicable for this item

1 = enumeration

2 = min/max

<PossibleValue> is one value from the allowed data range

<ValueDescription> is the meaning of the corresponding value

Some parameters are write protected (by a command level > 1) for certain items. These parameters are listed below the “#CCL levels are” line.

<CmdLevel> is the command level which allows write access to the parameter value.

IDN? (Get Device Identification)

Description: Reports the device identity number. Is identical in function with the *IDN? command (p. 16).

Format: IDN?

Arguments: None

Response: One-line string terminated by line feed with controller name, serial number and firmware version, see *IDN? for an example.

IFC (Set Interface Parameters Temporarily)

Description: Interface configuration.

The baud rate setting for the RS-232 serial interface is specified. After IFC is sent, the new setting becomes active and the host PC interface configuration may need to be changed to maintain communication (close the current connection and re-open it with the new baud rate).

Baud rate settings made with IFC are lost when the controller is powered down. To save settings to non-volatile memory and thus make them the power-on defaults, use IFS (p. 62) instead.

Alternatively, you can change the baud rate setting with SPA (p. 90) or SEP (p. 87) and save the current value with WPA (p. 137) to non-volatile memory (provided that the current command level provides write access to the parameter, see CCL (p. 26)). For the appropriate parameter ID, see below.

Format: IFC {<InterfacePam> <PamValue>}

Arguments: <InterfacePam> is the interface parameter to be changed, see below

<PamValue> gives the value of the interface parameter, see below

The following interface parameters can be set:

For <InterfacePam> = RSBAUD,
<PamValue> gives the baud rate to be used for RS-232 communication, default is 115200;
is also accessible as parameter ID 0x11000400, Uart Baudrate

Response: None

IFC? (Get Current Interface Parameters)

Description: Get the interface configuration parameter values from volatile memory.

The values from volatile memory can also be queried with SPA? (p. 94), for the corresponding parameter IDs see below.

Format: IFC? [{<InterfacePam>}]

Arguments: <InterfacePam> is the interface parameter to be queried, can be RSBAUD, IPADR, IPSTART, IPMASK, MACADR and IPMAXCONN

Response: {<InterfacePam>"="<PamValue> LF}

where

<PamValue> gives the value of the interface parameter from volatile memory

For <InterfacePam> = RSBAUD, <PamValue> gives the

current baud rate of the RS-232 communication;
is also accessible as parameter ID 0x11000400, Uart Baudrate

For <InterfacePam> = IPADR, the first four portions of <PamValue> give the IP address used for TCP/IP communication, the last portion gives the port;
is also accessible as parameter ID 0x11000600, IP Address

For <InterfacePam> = IPSTART, <PamValue> gives the current startup behavior setting for configuration of the IP address for TCP/IP communication,
0 = use IP address defined with IPADR
1 = use DHCP to obtain IP address, if this fails, use IPADR;
is also accessible as parameter ID 0x11000800, IP Configuration

For <InterfacePam> = IPMASK, <PamValue> gives the current IP mask setting to be used for TCP/IP communication, in the form uint.uint.uint.uint;
is also accessible as parameter ID 0x11000700, IP Mask

For <InterfacePam> = MACADR, <PamValue> gives the fixed, unique address of the network hardware in the E-727;
is also accessible as parameter ID 0x11000B00, MAC Address

IFS (Set Interface Parameters As Default Values)

Description: Interface parameter store.

The power-on default parameters for the interface are changed in non-volatile memory, but the current active parameters are not. Settings made with IFS become active with the next power-on or reboot.

To change the baud rate setting for the RS-232 serial connection immediately (but temporarily) use IFC (p. 59).

It is also possible to change the default settings in non-volatile memory with SEP (p. 87) and to read them with the SEP? (p. 89) command (provided that the current command level provides write access to the parameter, see CCL (p. 26)). Do **not** use RPA (p. 83) to activate the changed settings—except of baud rate changes after which the host PC interface configuration may need to be changed— because it will not be possible to maintain communication afterwards. For the appropriate parameter IDs see below.

NOTICE: The number of write cycles of non-volatile memory is limited. Write default values only when necessary.

Format: IFS <Pswd> {<InterfacePam> <PamValue>}

Arguments: <Pswd> is the password for writing to non-volatile memory, default is "100"

<InterfacePam> is the interface parameter to be changed, see below

<PamValue> gives the value of the interface parameter, see below

The following interface parameters can be set:

RSBAUD

<PamValue> gives the baud rate to be used for RS-232 communication, default is 115200;
is also accessible as parameter ID 0x11000400, Uart Baudrate

IPADR

The first four portions of <PamValue> specify the default IP address for TCP/IP communication, the last portion specifies the default port to be used, default is 192.168.168.10:50000;
is also accessible as parameter ID 0x11000600, IP Address
Note: While the IP address can be changed, the port must always be 50000!

IPSTART

<PamValue> defines the startup behavior for configuration of the IP address for TCP/IP communication,
0 = use IP address defined with IPADR
1 = use DHCP to obtain IP address, if this fails, use IPADR (default);
is also accessible as parameter ID 0x11000800, IP Configuration

IPMASK

<PamValue> gives the IP mask to be used for TCP/IP communication, in the form uint.uint.uint.uint, default is 255.255.255.0;
is also accessible as parameter ID 0x11000700, IP Mask

Response: None

IFS? (Get Interface Parameters As Default Values)

Description: Get the interface configuration parameter values stored in non-volatile memory (i.e. the current power-on default)

Format: IFS? [{<InterfacePam>}]

Arguments: <InterfacePam> is the interface parameter to be queried, can be RSBAUD, IPADR, IPSTART, IPMASK or MACADR, see IFS (p. 62) for details

Response: {<InterfacePam>"="<PamValue> LF}

where

<PamValue> is the value of the interface parameter in non-volatile memory, see IFS for possible values

For <InterfacePam> = MACADR, <PamValue> gives the unique address of the network hardware in the E-727 (is also accessible as parameter ID 0x11000B00, MAC Address)

IMP (Start Impulse And Response Measurement)

Description: Starts performing an impulse and recording the impulse response for the given axis.

An "impulse" consists of a relative move of the specified amplitude followed by an equal relative move in the opposite direction. Irrespective of the current operating mode (servo on or off), the impulse is performed relative to the current position.

The data recorder configuration, i.e. the assignment of data sources and record options to the recorder tables, can be set with DRC (p. 41).

The recorded data can be read with the DRR? command (p. 45).

Format: IMP <AxisID> <Amplitude>

Arguments <AxisID> is one axis of the controller

<Amplitude> is the height of the impulse
In closed-loop operation (servo ON), the given amplitude is interpreted as relative position value in either case. In open-loop operation (servo OFF), with the default settings of the output matrix, the amplitude also corresponds numerically to a relative axis position (see "Output Generation" in the E727T0005 user manual for more information).

Response: None

Troubleshooting: The control value resulting from the specified impulse height is out of limits:

Open-loop operation: the amplitude limitation depends on the voltage limit parameters (IDs 0x0B000007, 0x0B000008, 0x0C000000 and 0x0C000001)
Closed-loop operation: use TMN? (p. 104) and TMX? (p. 105) to ask for the current valid travel range limits.

Motion commands like IMP are not allowed when analog control input or wave generator output are active. See "Control Value Generation" in the E727T0005 user manual for details.

JOG (Start Motion With Given Velocity)

Description: Starts motion with the given (constant) velocity for the given axis. The sign of the velocity value gives the direction of motion.

The servo mode must be switched on when this command is used (closed-loop operation).

When motion started with JOG is executed, the target value is changed continuously according to the given velocity (can be checked with MOV?).

Motion started with JOG is executed in addition to motion started with other move commands (e.g. MOV or MVR).

As long as the axis motion is caused by JOG only, the axis stays on target (i.e. ONT? responds with 1 since the target is continuously adapted to the actual motion).

Motion started by JOG is stopped in the following cases:

- The velocity is set to 0 with JOG.
- #24, STP or HLT is sent: these commands set the velocity for JOG to 0.
- A travel range limit is reached: the velocity for JOG remains unchanged, no error is set, the target value is set equal to the limit value.

JOG can be changed while the axis is moving.

Format: JOG {<AxisID> <Velocity>}

Arguments <AxisID> is one axis of the controller

<Velocity> gives the velocity and the direction for axis motion started with JOG. With a positive value the target is increased, with a negative value the target is decreased; zero stops the motion caused by JOG. float, signed

Response: None

Example:

Send:	MOV? 1
Receive:	1=0.000000000e+00
Send:	POS? 1
Receive:	1=-9.843791835e-03
Send:	JOG 1 0.001
Note:	According to the JOG command, axis 1 now moves with 0.001 µm/s in positive direction. The target is changed with every servo cycle.
Send:	ONT?
Receive:	1=1
Send:	#5
Recevice:	0
Note:	Although axis 1 is moving, it is on target since motion is caused by JOG only. Hence the motion status queried with #5 is “no axis is

moving".

Send:	MOV? 1
Receive:	1=1.914202720e-01
Send:	POS? 1
Receive:	1=1.929343045e-01
Send:	MOV? 1
Receive:	1=1.950851232e-01
Send:	POS? 1
Receive:	1=1.963570416e-01
Send:	MOV? 1
Receive:	1=2.011617124e-01
Send:	POS? 1
Receive:	1=2.026674747e-01
Send:	MOV? 1
Receive:	1=2.049337626e-01
Send:	POS? 1
Receive:	1=2.078935951e-01
Note:	MOV? and POS? are sent alternately. The response shows that the target is constantly increased, and that the current position follows the target values.
Send:	JOG? 1
Receive:	1=1.000000047e-03
Send:	STP
Send:	JOG? 1
Receive:	1=0.000000000e+00
Send:	TMX? 1
Receive:	1=1.000000000e+02
Send:	MOV 1 100

Send:	POS? 1
Receive:	1=9.998706818e+01
Send:	JOG 1 0.001
Send:	ERR?
Receive:	0
Send:	JOG? 1
Receive:	1=1.000000047e-03
Note:	The upper travel range limit of axis 1 is 100 µm. (TMX? 1 responds 100). After the corresponding MOV command, axis 1 is at its upper range limit. JOG cannot start motion in positive direction, but the given velocity value remains active. If axis 1 would be moved, for example, to position 10, then the motion commanded with JOG would start.

JOG? (Get Velocity For Motion Caused By JOG)

Description:	Gets the velocity and direction for motion caused by JOG (p. 66).
Format:	JOG? [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller
Response:	{<AxisID>"="<Velocity>LF}
	where
	<Velocity> is the velocity and the direction for axis motion caused by JOG. See JOG for details. float, signed
Troubleshooting:	Illegal axis identifier

JRC (Jump Relatively Depending On Condition)

Description: Jumps relatively depending on a given condition of the following type: one given value is compared with a queried value according to a given rule.

Can only be used in macros.

Format: JRC <Jump> <CMD?> <OP> <Value>

Arguments: <Jump> is the size of the relative jump. -1 means the macro execution pointer jumps back to the previous line, 0 means the command is executed again, which is the same behavior as with WAC (p. 119). 1 jumps to the next line, making the command unnecessary, and 2 jumps over the next command. Only jumps within the current macro are allowed.

<CMD?> is one query command in its usual syntax. The response has to be a single value and not more. For an example see below.

<OP> is the operator to be used. The following operators are possible:
= <= < > >= !=
Important: There must be a blank space before and after the operator!

<Value> is the value to be compared with the response to <CMD?>.

Response: None

Troubleshooting: Check proper jump target

Example: Using the following macro, you can stop motion of axis 1 using a stop button connected to a digital input. The stop button is checked until the axis has reached the target position (ONT? query). When the stop button is pressed as long as the target position has not been reached yet: The response to the POS? 1 query is copied to the TARGET variable. Then this variable is used as second argument for the MOV command. Thus the stage stays where it just was. To clean up, TARGET is defined as empty with the VAR command which deletes the variable.

Write the "stop" macro:
MAC BEG stop
MOV 1 20
JRC 2 DIO? 1 = 1
JRC -1 ONT? 1 = 0
CPY TARGET POS? 1
MOV 1 ${TARGET}
VAR TARGET
MAC END

MAC (Call Macro Function)

Description: Calls a macro function. Permits recording, deleting and running macros on the controller.

Format: MAC <keyword> {<parameter>}

in particular:

MAC BEG <macroname>

MAC DEF <macroname>

MAC DEF?

MAC DEL <macroname>

MAC END

MAC ERR?

MAC FREE?

MAC NSTART <macroname> <uint> [<String1> [<String2>]]

MAC START <macroname> [<String1> [<String2>]]

Arguments <keyword> determines which macro function is called. The following keywords and parameters are used:

MAC BEG <macroname>

Starts recording a macro to be named macroname on the

controller; may not be used in a macro; the commands that follow become the macro. End the recording with MAC END. Note that erroneous macro content cannot be detected by sending the ERR? command.

MAC END

Stops macro recording (cannot become part of a macro)

MAC ERR?

Reports the last error which occurred during macro execution.

Response: <macroname> <uint1>"="<uint2> <"<"CMD">">

where <macroname> is the name of the macro, <uint1> is the line in the macro, <uint2> is the error code and <"<"CMD">"> is the erroneous command which was sent to the parser.

MAC FREE?

Gets the free memory space for macro recording (unit: number of characters)

MAC DEF <macroname>

Sets specified macro as start-up macro. This macro will be automatically executed with the next switching-on or reboot of the controller. If <macroname> is omitted, the current start-up macro selection is canceled.

MAC DEF?

Asks for the start-up macro
Response: <macroname>
If no start-up macro is defined, the response is an empty string with the terminating character.

MAC DEL <macroname>

Deletes specified macro.

MAC NSTART <macroname> <uint> [<String1> [<String2>]]

Repeats the specified macro <uint> times. Another execution is started when the last one is finished.

<String1> and <String2> are optional arguments which give the values for local variables 1 and 2 used in the given macro. <String1> and <String2> can be given directly or via the values of variables. Macro execution will fail if the macro contains local variables but <String1> and <String2> are omitted in the MAC NSTART command. See “Variables” in the E727T0005 user manual for further details.

MAC START <macroname> [<String1> [<String2>]]

Starts one execution of the specified macro. <String1> and <String2> have the same function as with MAC NSTART.

Response: None

Troubleshooting: Macro recording is active (keywords BEG, DEL) or inactive (END)

Macro contains a disallowed MAC command

Notes: During macro recording no macro execution is allowed.

When macros are recorded on the Controller macros tab in PIMikroMove, the MAC BEG and MAC END commands must be omitted.

A running macro sends no responses to any interface.

The following commands provided by the E-727 can only be used in macros:

DEL (p. 37), JRC (p. 70), MEX (p. 74) and WAC (p. 119).

You can query with #8 (p. 14) if a macro is currently running on the controller.

For further details, see “Controller Macros” in the E727T0005 user manual.

MAC? (List Macros)

Description: Lists macros or content of a given macro.

Format: MAC? [<macroname>]

Arguments: <macroname>: name of the macro whose content shall be listed; if omitted, the names of all stored macros are listed.

Response: <string>

If <macroname> was given, <string> is the content of this macro;

If <macroname> was omitted, <string> is a list with the names of all stored macros

Troubleshooting: Macro <macroname> not found

MEX (Stop Macro Execution Due To Condition)

Description: Stops macro execution due to a given condition of the following type: a given value is compared with a queried value according to a given rule.

Can only be used in macros.

When the macro interpreter accesses this command, the condition is checked. If it is true, the current macro is stopped; otherwise macro execution is continued with the next line. Should the condition be fulfilled later, the interpreter will ignore it.

See also the WAC command (p. 119).

Format: MEX <CMD?> <OP> <Value>

Arguments <CMD?> is one query command in its usual syntax. The response has to be a single value and not more. For an example see below.

<OP> is the operator to be used. The following operators are possible:
= <= < > >= !=
Important: There must be a blank space before and after the operator!

<Value> is the value that is compared with the response to <CMD?>.

Response: None

MOV (Set Target Position)

Description: Set new absolute target position for given axis.

Servo must be enabled for the commanded axis prior to using this command (closed-loop operation).

Format: MOV {<AxisID> <Position>}

Arguments <AxisID> is one axis of the controller

<Position> is the new absolute target position in physical units.

Response: None

Troubleshooting: Target position out of limits. Use TMN? (p. 104) and TMX? (p. 105) to ask for the current valid travel range limits.

Illegal axis identifier

Servo is Off for one of the axes specified.

Motion commands like MOV are not allowed when analog control input or wave generator output are active on the axis. See "Control Value Generation" in the E727T0005 user manual for details.

Notes: During a move, a new move command resets the target to a new value and the old one may never be reached.

The motion can be stopped by #24 (p. 15), STP (p. 96) and HLT (p. 55).

Example 1:

Send:	MOV 1 10
Note:	Axis 1 moves to 10 (target position in µm)

Example 2:

Send:	MOV 1 243
Send:	ERR?
Receive:	7
Note:	The axis does not move. The error code "7" in the reply to the ERR? command (p. 50) indicates that the target position given in the move command is out of limits.

Example 3:

Send:	MOV 1 10 2 100 3 4000
Send:	ERR?
Receive:	7
Note:	The axes do not move. The error code "7" in the reply to the ERR? command (p. 50) indicates that at least one of the target positions given in the move command is out of limits.

MOV? (Get Target Position)

Description: Returns last valid commanded target position.

Format: MOV? [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller

Response: {<AxisID>"="<float> LF}

where

<float> is the last commanded target position in physical units

Troubleshooting: Illegal axis identifier

Notes: The target position can be changed by various sources, e.g. by commands that cause motion (MOV (p. 75), MVR (p. 77), IMP (p. 65), STE (p. 95)), by the wave generator, by an analog input signal, and by the SPI interface. See "Control Value Generation" in the E727T0005 user manual for details.

MOV? gets the commanded positions. Use POS? (p. 81) to get the current positions.

MVR (Set Target Relative To Current Position)

Description: Move given axes relative to the last commanded target position.

The new target position is calculated by adding the given value <Distance> to the last commanded target value.

Servo must be enabled for the commanded axis prior to using this command (closed-loop operation).

Format: MVR {<AxisID> <Distance>}

Arguments: <AxisID> is one axis of the controller.

<Distance> gives the distance to move; the sum of the distance and the last commanded target position is set as new target position (in physical units).

Response: None

Troubleshooting: Target position out of limits. Use TMN? (p. 104) and TMX? (p. 105) to ask for the current valid travel range limits, and MOV? (p. 77) for the current target.

Illegal axis identifier

Servo is Off for one of the axes specified.

Motion commands like MVR are not allowed when analog control input or wave generator output are active on the axis. See "Control Value Generation" in the E727T0005 user manual for details.

Notes: The motion can be stopped by #24 (p. 15), STP (p. 96) and HLT (p. 55).

Example:

Send:	MOV 1 0.5
Note:	This is an absolute move.
Send:	POS? 1
Receive:	1=0.500000
Send:	MOV? 1
Receive:	1=0.500000
Send:	MVR 1 2
Note:	This is a relative move.
Send:	POS? 1
Receive:	1=2.500000
Send:	MVR 1 2000
Note:	New target position of axis 1 would exceed

motion range. Command is ignored, i.e. the target position remains unchanged, and the axis does not move.

Send: MOV? 1

Receive: 1=2.500000

Send: POS? 1

Receive: 1=2.500000

ONT? (Get On Target State)

Description: Get on-target status of given axis.

If all arguments are omitted, gets status of all axes.

Format: ONT? [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller.

Response: {<AxisID>"="<uint> LF}

where

<uint> = "1" when the specified axis is on-target, "0" otherwise.

Troubleshooting: Illegal axis identifier

In open-loop operation (servo off), the axis will never be on target because sensor feedback is not used and hence the current on-target status cannot be determined.

Notes: The on-target status is influenced by two parameters: settling window (On Target Tolerance, ID 0x07000900) and settling time (Settling Time, ID 0x07000901).

The on-target status is true when the current position is inside the settling window and stays there for at least the settling time. The settling window is centered around the target position.

OVF? (Get Overflow State)

Description: Get overflow status of given axis.

If all arguments are omitted, gets status of all axes.

Overflow means that the control variables are out of range (can only happen if controller is in closed-loop operation).

Format: OVF? [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller.

Response: {<AxisID>"="<uint> LF}

where

<uint> = "0" (axis is not in overflow) or "1" (axis is in overflow)

Troubleshooting: Illegal axis identifier

POS? (Get Real Position)

Description:	Returns the current axis position. If all arguments are omitted, gets current position of all axes.
Format:	POS? [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller.
Response:	{<AxisID>"="<float> LF} where <float> is the current axis position in physical units
Troubleshooting:	Illegal axis identifier
Note:	To request the current position of input signal channels (sensors) in physical units, use the TSP? (p. 108) command instead.

PUN? (Get Axis Unit)

Description:	Gets the current unit of the axis. If all arguments are omitted, the current unit for all axes is queried.
Format:	PUN? [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller.
Response:	{<AxisID>"="<string> LF} where <string> is the current unit of the axis.
Troubleshooting:	Illegal axis identifier

Note: Gets the Axis Unit parameter value in volatile memory (ID 0x07000601).

RBT (Reboot System)

Description: Reboot system. Controller behaves just like after power-on.

Format: RBT

Arguments: None

Response: None

Notes: With TCP/IP and USB connections, communication cannot be maintained after the E-727 is power-cycled or rebooted. The connection must then be closed and reopened.

RMC? (List Running Macros)

Description: Lists macros which are currently running.

Format: RMC?

Arguments: None

Response: {<macroname> LF}

where

<macroname> is the name of one macro which is saved on the controller and currently running. The response is an empty line when no macro is running.

RPA (Reset Volatile Memory Parameters)

Description: Resets the given parameter of the given item. The value from non-volatile memory is written into volatile memory.

Related commands:

With HPA? (p. 56) you can obtain a list of the available parameters. SPA (p. 90) affects the parameter settings in volatile memory, WPA (p. 137) writes parameter settings from volatile to non-volatile memory, and SEP (p. 87) writes parameter settings directly into non-volatile memory (without changing the settings in volatile memory).

See SPA for an example.

Format: RPA [{<ItemID> <PamID>}]

Arguments: <ItemID> is the item for which a parameter is to be reset. See below for details.

<PamID> is the parameter ID, can be written in hexadecimal or decimal format. See below for details.

Response: None

Troubleshooting: Illegal item identifier, wrong parameter ID

Notes: This procedure can take a few seconds.

NOTICE: If the communication between host PC and E-727 is done via TCP/IP, do not use RPA after you have changed the parameters of the TCP/IP communication with IFS (p. 62) or SEP (p. 87) in non-volatile memory, because it will not be possible to maintain communication afterwards.

While parameter values are being read from or written to the non-volatile memory of the E-727, the servo is temporarily switched off. This may lead to faulty control or large position errors.

Observe the following when you use the commands SEP, SEP?, RPA, WPA:

- Avoid motions of the axes.

- Do not carry out measurements with the axes.

Available item IDs and parameter IDs: The item type depends on the parameter, see "Parameter Overview" in the E727T0005 user manual for the item type concerned. See "Axes, Channels, Functional Elements" in the E727T0005 user manual for the identifiers of the items.

Valid parameter IDs are given in "Parameter Overview" in the E727T0005 user manual.

RTR (Set Record Table Rate)

Description: Sets the record table rate, i.e. the number of servo-loop cycles to be used in data recording operations. Settings larger than 1 make it possible to cover longer time periods.

Format: RTR <RecordTableRate>

Arguments: <RecordTableRate> is the table rate to be used for recording operations (unit: number of servo-loop cycles), must be an integer value larger than zero

Response: None

Notes: RTR affects the Data Recorder Table Rate parameter, ID 0x16000000.

The duration of the recording can be calculated as follows:

Rec. Duration = Servo Update Time * RTR value * Number of Points

where

Servo Update Time is given in seconds by parameter 0x0E000200

Number of Points is the length of the data recorder table

For more information see "Data Recording" in the E727T0005 user manual.

The record table rate set with RTR is saved in volatile memory (RAM) only. To save the currently valid value to non-volatile memory, where it becomes the power-on default, you must use WPA (p. 137). Changes not saved with WPA will be lost when the controller is powered down. To have write access to the parameter, it might be necessary to switch to a higher command level using CCL (p. 26).

RTR? (Get Record Table Rate)

Description: Gets the current record table rate, i.e. the number of servo-loop cycles used in data recording operations.

Format: RTR?

Arguments: None

Response: <RecordTableRate> is the table rate used for recording operations (unit: number of servo-loop cycles)

Notes: Gets the Data Recorder Table Rate parameter value in volatile memory (ID 0x16000000).

For more information see "Data Recording" in the E727T0005 user manual.

SAI? (Get List Of Current Axis Identifiers)

Description: Gets the axis identifiers.

See also "Axes, Channels, Functional Elements" in the E727T0005 user manual.

Format: SAI? [ALL]

Arguments: [ALL] is optional and provided for compatibility with controllers which allow for axis deactivation. [ALL] then ensures that the answer also includes the axes which are "deactivated" (i.e. not assigned to stages).

Response: {<AxisID> LF}

<AxisID> is one axis of the controller.

SEP (Set Non-Volatile Memory Parameters)

Description: Set a parameter of a given item to a different value in non-volatile memory, where it becomes the new power-on default.

After parameters were set with SEP, you can use RPA (p. 83) to activate them (write them to volatile memory) without controller reboot.

NOTICE: This command is for setting hardware-specific parameters. Wrong values may lead to improper operation or damage of your hardware!

Related commands:

HPA? (p. 56) returns a list of the available parameters.

SPA (p. 90) writes parameter settings into volatile memory (without changing the settings in non-volatile memory).

WPA (p. 137) writes parameter settings from volatile to non-volatile memory.

See SPA for an example.

Format: SEP <Pswd> {<ItemID> <PamID> <PamValue>}

Arguments <Pswd> is the password for writing to non-volatile memory, default is "100"

<ItemID> is the item for which a parameter is to be changed in non-volatile memory. See below for details.

<PamID> is the parameter ID, can be written in hexadecimal or decimal format. See below for details.

<PamValue> is the value to which the given parameter of

the given item is set

Response: None

Troubleshooting: Illegal item identifier, wrong parameter ID, invalid password, command level too low for write access

Notes: To have write access to the parameter(s), it might be necessary to switch to a higher command level using CCL (p. 26).

While parameter values are being read from or written to the non-volatile memory of the E-727, the servo is temporarily switched off. This may lead to faulty control or large position errors.

Observe the following when you use the commands SEP, SEP?, RPA, WPA:

- Avoid motions of the axes.

- Do not carry out measurements with the axes.

NOTICE: The number of write cycles of non-volatile memory is limited. Write default values only when necessary.

Available item IDs and parameter IDs: The item type depends on the parameter, see "Parameter Overview" in the E727T0005 user manual for the item type concerned. See "Axes, Channels, Functional Elements" in the E727T0005 user manual for the identifiers of the items.

Valid parameter IDs are given in "Parameter Overview" in the E727T0005 user manual.

SEP? (Get Non-Volatile Memory Parameters)

Description:	Get the value of a parameter of a given item from non-volatile memory. With HPA? (p. 56) you can obtain a list of the available parameters and their IDs.
Format:	SEP? [{<ItemID> <PamID>}]
Arguments:	<ItemID> is the item for which a parameter value from non-volatile memory is to be queried. See below for details. <PamID> is the parameter ID, can be written in hexadecimal or decimal format. See below for details.
Response:	{<ItemID> <PamID>"="<PamValue> LF} where <PamValue> is the value of the given parameter for the given item
Troubleshooting:	Illegal item identifier, wrong parameter ID
Notes:	While parameter values are being read from or written to the non-volatile memory of the E-727, the servo is temporarily switched off. This may lead to faulty control or large position errors. Observe the following when you use the commands SEP, SEP?, RPA, WPA: - Avoid motions of the axes. - Do not carry out measurements with the axes.
Available item IDs and parameter IDs:	The item type depends on the parameter, see "Parameter Overview" in the E727T0005 user manual for the item type concerned. See "Axes, Channels, Functional Elements" in the E727T0005 user manual for the identifiers of the items. Valid parameter IDs are given in "Parameter Overview" in the E727T0005 user manual.

SPA (Set Volatile Memory Parameters)

Description: Set a parameter of a given item to a value in volatile memory (RAM). Parameter changes will be lost when the controller is powered down or rebooted or when the parameters are restored with RPA (p. 83).

NOTICE: This command is for setting hardware-specific parameters. Wrong values may lead to improper operation or damage of your hardware!

Related commands:

HPA? (p. 56) returns a list of the available parameters.

SEP (p. 87) writes parameter settings directly into non-volatile memory (without changing the settings in volatile memory).

WPA (p. 137) writes parameter settings from volatile to non-volatile memory.

RPA resets volatile memory to the value in non-volatile memory.

Format: SPA {<ItemID> <PamID> <PamValue>}

Arguments <ItemID> is the item for which a parameter is to be changed in volatile memory. See below for details.

<PamID> is the parameter ID, can be written in hexadecimal or decimal format. See below for details.

<PamValue> is the value to which the given parameter of the given item is set

Response:	None
Troubleshooting:	Illegal item identifier, wrong parameter ID,value out of range, command level too low for write access
Notes:	Do **not** change the current settings of the communication interface—except of the baud rate—because it will not be possible to maintain communication afterwards. To have write access to the parameter(s), it might be necessary to switch to a higher command level using CCL (p. 26).
Available item IDs and parameter IDs:	The item type depends on the parameter, see "Parameter Overview" in the E727T0005 user manual for the item type concerned. See "Axes, Channels, Functional Elements" in the E727T0005 user manual for the identifiers of the items. Valid parameter IDs are given in "Parameter Overview" in the E727T0005 user manual.
Example 1:	Send: SPA 1 0x16000000 8 Note: Set the Data Recorder Table Rate for the controller to 8, parameter ID written in hexadecimal format Send: SPA 1 369098752 2 Note: Sets the Data Recorder Table Rate for the controller to 2, parameter ID written in decimal format
Example 2:	With an E-727.3CDA, the analog input line "In 2" (which is internally handled as the 5th input signal channel), is to be used as control source for axis 3. For that purpose, the corresponding coefficient in the input matrix (Position From Sensor 5, parameter ID 0x07000504) must be set to 0 for axis 3 so that the analog input line does not participate as sensor in the axis position calculation. Send: CCL 1 advanced Note: Switch to command level 1 because this level is required for write access to the Position

From Sensor 8 parameter.

Send: SPA 3 0x07000504 0

Note: The analog input line "In 2" will not participate in the position calculation of axis 3. The setting is made in volatile memory only.

Now make further configuration settings in volatile memory using SPA and then test the functioning of the system. See "Using the Analog Input" in the E727T0005 user manual for more information. If everything is okay and you want to use this system configuration after the next power-on, save the parameter settings from volatile to non-volatile memory.

Send: WPA 100

Note: When WPA is used without specifying any parameters, all currently valid parameter values from volatile memory are saved. Keep in mind that if the TCP/IP interface is used for communication and the IP address is obtained via DHCP server, the current IP address will also be saved.

Send: SEP? 3 0x07000504

Receive: 3 0x7000504=0.000000e+00

Note: Check the parameter settings in non-volatile memory.

Example 3: The task performed in example 2 can also be done in the following way, provided you are sure that the new system configuration will work:

Send: CCL 1 advanced

Note: Switch to command level 1 because this level is required for write access to the Position From Sensor 5 parameter.

Send: SEP 100 3 0x07000504 0

Note: The analog input line "In 2" no longer participates in the position calculation of axis 3. The setting is made in non-volatile memory and hence is the new power-on default, but is not yet active.

Make further configuration settings in non-volatile memory using SEP. See "Using the Analog Input" in the E727T0005 user manual for more information. To use the new settings immediately, you now have to load them to volatile memory (otherwise they would become active after the next power-on or reboot of the controller).

Send: RPA

Note: The new configuration is now active.

Send: SPA? 3 0x07000504

Receive: 3 0x7000504=0.000000e+00

Note: Check the parameter settings in volatile memory.

SPA? (Get Volatile Memory Parameters)

Description:	Get the value of a parameter of a given item from volatile memory (RAM). With HPA? (p. 56) you can obtain a list of the available parameters and their IDs.
Format:	SPA? [{<ItemID> <PamID>}]
Arguments:	<ItemID> is the item for which a parameter is to be queried in volatile memory. See below for details. <PamID> is the parameter ID, can be written in hexadecimal or decimal format. See below for details.
Response:	{<ItemID> <PamID>"="<PamValue> LF} where <PamValue> is the value of the given parameter for the given item
Troubleshooting:	Illegal item identifier, wrong parameter ID
Available item IDs and parameter IDs:	The item type depends on the parameter, see "Parameter Overview" in the E727T0005 user manual for the item type concerned. See "Axes, Channels, Functional Elements" in the E727T0005 user manual for the identifiers of the items. Valid parameter IDs are given in "Parameter Overview" in the E727T0005 user manual.

SSN? (Get Device Serial Number)

Description:	Gets the serial number of the E-727.
Format:	SSN?
Arguments:	None
Response:	<SerialNumber> is the serial number of the device.

STE (Start Step And Response Measurement)

Description:	Starts performing a step and recording the step response for the given axis. The data recorder configuration, i.e. the assignment of data sources and record options to the recorder tables, can be set with DRC (p. 41). The recorded data can be read with the DRR? (p. 45) command.
Format:	STE <AxisID> <Amplitude>
Arguments	<AxisID> is one axis of the controller <Amplitude> is the height of the step. See below for details.
Response:	None
Troubleshooting:	The control value resulting from the specified step height is out of limits: open-loop operation: the amplitude limitation results from the voltage limit parameters (IDs 0x0B000007, 0x0B000008, 0x0C000000 and 0x0C000001) closed-loop operation: use TMN? (p. 104) and TMX? (p. 105) to ask for the current valid travel range limits. Motion commands like STE are not allowed when analog control input or wave generator output are active. See

"Control Value Generation" in the E727T0005 user manual for details.

Notes: A "step" consists of a relative move of the specified amplitude. Irrespective of the current operating mode (servo on or off), the step is performed relative to the current position.

In closed-loop operation (servo ON), the given amplitude is interpreted as relative position value in either case. In open-loop operation (servo OFF), with the default settings of the output matrix, the amplitude also corresponds numerically to a relative axis position (see "Output Generation" in the E727T0005 user manual for more information).

STP (Stop All Motion)

Description: Stops all motion abruptly. For details see the notes below.

Sets error code to 10.

This command is identical in function to #24 (p. 15) which should be preferred when the controller is performing time-consuming tasks.

Format: STP

Arguments: None

Response: None

Troubleshooting: Communication breakdown

Notes: STP stops motion of all axes. Also stops macro execution.

After the axes are stopped, if servo is on, their target positions are set to their current positions, or if servo is off, their open-loop control values are set to their last valid control values.

When the analog input is used as control source and the axis motion is stopped with STP or #24, the behaviour depends on the value of the Discon. Target Man. In With Stop parameter (ID 0x0E001E00): 1 = the analog input channel is disconnected from the axis; 0 = the analog input channel remains connected to the axis. If the analog input channel is disconnected from the axis: To recommence commanding the axis via the analog input, the corresponding input signal channel must be reconnected to the axis. See "How to work with the Analog Input" in the E727T0005 user manual for more information.

The Disable Error 10 parameter (ID 0x0e000301) can be used to avoid that error code 10 is set when axes are stopped with the STP, #24 or HLT commands.
0 = OFF (Error code 10 is set.)
1 = ON (Error code 10 is not set.)

SVA (Set Open-Loop Axis Value)

Description: Set absolute open-loop control value to move the axis.

Servo must be switched off (open-loop operation) when using this command.

The motion can be stopped by #24 (p. 15), STP (p. 97) and HLT (p. 55).

Format: SVA {<AxisID> <Amplitude>}

Arguments	<AxisID> is one axis of the controller <Amplitude> is the new absolute open-loop control value. See Notes below for details.
Response:	None
Troubleshooting:	Illegal axis identifier Servo is On for one of the specified axes Motion commands like SVA are not allowed when analog control input or wave generator output are active. See "Control Value Generation" in the E727T0005 user manual for details. For axes driven by conventional piezo actuators: The control value specified by the given amplitude is out of limits. The limitation results from the voltage limit parameters (IDs 0x0B000007, 0x0B000008, 0x0C000000 and 0x0C000001) of the output signal channels which would be involved in the axis motion.
Notes:	<Amplitude> is given as dimensionless value. The interpretation of the amplitude value depends on the settings of the output matrix (see "Output Generation" in the E727T0005 user manual for more information). With the default matrix coefficients, open-loop control values numerically correspond to axis position values.
Example 1:	Send: SVA 1 10 Note: Assumed that the default output matrix settings are valid so that the amplitude corresponds to a position value, axis 1 moves to 10 µm (approximately) with no position control (in open-loop operation there will be no correction of drift or other effects, and there is no position feedback used to assure that the target is actually reached).

Example 2:	Send:	SVA 1 300
	Send:	ERR?
	Receive:	17
	Note:	The axis does not move. The error code "17" reported by the ERR? command (p. 50) indicates that the open-loop value set by SVA is out of limits.
Example 3:	Send:	SVA 1 300 2 60 3 100
	Send:	ERR?
	Receive:	17
	Note:	The axes do not move. The error code "17" reported by the ERR? command (p. 50) indicates that at least one of the open-loop values set by SVA is out of limits.

SVA? (Get Open-Loop Axis Value)

Description:	Returns last valid open-loop control value of given axis. The open-loop control value is changed by multiple sources, e.g. by commands that cause motion (SVA (p. 97), SVR (p. 102), IMP (p. 65), STE (p. 95)), by the wave generator, by an analog input signal, and by the SPI interface. See "Control Value Generation" in the E727T0005 user manual for details.
Format:	SVA? [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller
Response:	{<AxisID>"="<float> LF}
	where
	<float> is the last commanded open-loop control value. See the <Amplitude> description of the SVA command for details.
Troubleshooting:	Illegal axis identifier

SVO (Set Servo State)

Description:	Sets servo-control state for given axes (open-loop or closed-loop operation).
Format:	SVO {<AxisID> <ServoState>}
Arguments:	<AxisID> is one axis of the controller <ServoState> can have the following values: 0 = servo off (open-loop operation) 1 = servo on (closed-loop operation)
Response:	None
Troubleshooting:	Illegal axis identifier If the internal temperature of the E-727 exceeds threshold 2 (see DIA?, p. 37), the amplifier output is switched off automatically, and all axes are switched to open-loop control.
Notes:	The current servo state affects the applicable move commands: servo-control off: use SVA (p. 97) and SVR (p. 102) servo-control on: use MOV (p. 75) and MVR (p. 77) When servo is switched off while the axis is moving, the axis stops. Exception: When the analog input is being used as control source and servo is switched off, the axis motion will continue in open-loop mode. Servo-control cannot be switched off while the wave generator is running for the axis. Using the Power Up Servo On Enable parameter (ID 0x07000800), you can configure the controller so that servo is automatically switched on upon power-on or reboot. To do this, set the value of the parameter to 1 in

non-volatile memory (using SEP (p. 87) or SPA (p. 90) + WPA (p. 137)).

When switching from open-loop to closed-loop control, the behaviour depends on the setting made with parameter 0x0e002000. Default: The current axis position is set as the target position. For further details, see “Parameter Overview” in the E727T0005 user manual.

To have write access to the parameters, it might be necessary to switch to a higher command level using CCL (p. 26).

SVO? (Get Servo State)

Description: Gets servo-control state of given axes.

If all arguments are omitted, gets status of all axes.

Format: SVO? [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller

Response: {<AxisID>"="<ServoState> LF}

where

<ServoState> is the current servo state of the axis:
0 = servo off (open-loop operation)
1 = servo on (closed-loop operation)

Troubleshooting: Illegal axis identifier

SVR (Set Relative Open-Loop Axis Value)

Description: Set open-loop control value relative to the current open-loop control value to move the axis.

The new open-loop control value is calculated by adding the given value <Difference> to the last commanded open-loop control value.

Servo must be off when using this command (open-loop operation).

The motion can be stopped by #24 (p. 15), STP (p. 96) and HLT (p. 55).

Format: SVR {<AxisID> <Difference>}

Arguments <AxisID> is one axis of the controller

<Difference> is the value which is added to the current open-loop control value (dimensionless)
The interpretation of the difference value depends on the settings of the output matrix (see "Output Generation" in the E727T0005 user manual for more information). With the default matrix coefficients, open-loop control values numerically correspond to axis position values.

Response: None

Troubleshooting: The specified control value is out of limits. The limitation results from the voltage limit parameters (IDs 0x0B000007, 0x0B000008, 0x0C000000 and 0x0C000001) of the output signal channels which would be involved in the axis motion.

Illegal axis identifier

Servo is On for one of the specified axes

Motion commands like SVR are not allowed when analog control input or wave generator output are active. See "Control Value Generation" in the E727T0005 user manual for details.

TAD? (Get ADC Value Of Input Signal)

Description: Get the current value from the specified input signal channel's A/D converter. This value represents the digitized signal value without filtering and linearization. Using this command it is possible to check for sensor overflow.

Multiple input signal channels (sensors) could be involved in the control of one logical axis (see "Input Signal Processing" in the E727T0005 user manual). TAD? reads the values for the individual input signal channels, not for a logical axis.

Format: TAD? [{<InputSignalID>}]

Arguments: <InputSignalID> is one input signal channel of the controller

Response: {<InputSignalID>"="<float> LF}

where

<float> is the current A/D value, dimensionless

TIO? (Tell Digital I/O Lines)

Description: Tells number of installed digital I/O lines

Format: TIO?

Arguments: None

Response: I=<uint1>
O=<uint2>

where

<uint1> is the number of digital input lines.
<uint2> is the number of digital output lines.

Notes: All digital I/O lines are found on the Digital I/O socket (p. 164) of the E-727.

The digital output lines reported by TIO? are output 1 to output 4. The output lines 1 to 3 can be programmed for trigger output using the CTO command (see „External Triggering/Signaling“ in the E727T0005 user manual).

The 4th output line (pin 8 of the Digital I/O socket) outputs the servo cycles of the E-727 and is not accessible for commands.

The digital input lines reported by TIO? are input 1 to input 4. The state of the lines can be queried with the DIO? command (p. 39). The input 1 and input 2 lines can be used with WGO to start the wave generator output (see “Wave Generator Started by Trigger Input” in the E727T0005 user manual. Input 4 line can be configured as reset input (see p. 164).

TLT? (Get Number of DDL Tables)

Description: Tell number of Dynamic Digital Linearization (DDL) tables available on the controller.

Format: TLT?

Arguments: None

Response <uint> is the number of DDL tables which are available

TMN? (Get Minimum Commandable Position)

Description: Get the minimum commandable position in physical units.

Format: TMN? [{ <AxisID>}]

Arguments: <AxisID>: is one axis of the controller

Response	{<AxisID>"="<float> LF}
	where
	<float> is the minimum commandable position in physical units
Note:	The minimum commandable position is defined by the Range Limit min parameter, ID 0x07000000.

TMX? (Get Maximum Commandable Position)

Description:	Get the maximum commandable position in physical units.
Format:	TMX? [{ <AxisID>}]
Arguments:	<AxisID>: is one axis of the controller
Response	{<AxisID>"="<float> LF}
	where
	<float> is the maximum commandable position in physical units
Description:	The maximum commandable position is defined by the Range Limit max parameter, ID 0x07000001.

TNR? (Get Number of Record Tables)

Description:	Get the number of data recorder tables currently available on the controller.
Format:	TNR?
Arguments:	None
Response	<uint> is the number of data recorder tables which are currently available

Notes: The answer gives the value of the Data Recorder Chan Number parameter, ID 0x16000300. You can change the parameter value to increase or decrease the number of data recorder tables.

For more information see "Data Recording" in the E727T0005 user manual.

TNS? (Get Normalized Input Signal Value)

Description: Get the normalized value for the given input signal channel. This value is internally the input for the mechanics linearization.

Multiple input signal channels (sensors) could be involved in the control of one logical axis (see "Input Signal Processing" in the E727T0005 user manual). TNS? reads the values for the individual input signal channels, not for a logical axis.

Format: TNS? [{<InputSignalID>}]

Arguments: <InputSignalID> is one input signal channel of the controller

Response: {<InputSignalID>"="<float> LF}

where

<float> is the normalized value ranging from controller specific minimum to maximum (e.g. -100 to 100), dimensionless

TPC? (Get Number of Output Signal Channels)

Description: Get the number of output signal channels available on the controller.

Note that the output signal channels are comprised of the piezo channels and any additional analog output channels. The number of piezo channels can be queried with the Number Of Piezo Channels parameter, ID 0x0E000B04. See "Axes, Channels, Functional Elements" in the E727T0005 user manual for more information.

Format: TPC?

Arguments: None

Response <uint> is the number of output signal channels which are available; the answer gives the value of the Number Of Output Channels parameter, ID 0x0E000B01

TSC? (Get Number of Input Signal Channels)

Description: Get the number of input signal channels available on the controller.

Note that the input signal channels are comprised of the sensor channels and any additional analog input channels. The number of sensor channels can be queried with the Number Of Sensor Channels parameter, ID 0x0E000B03. See "Axes, Channels, Functional Elements" in the E727T0005 user manual for more information.

Format: TSC?

Arguments: None

Response <uint> is the number of input signal channels which are available; the answer gives the value of the Number Of Input Channels parameter, ID 0x0E000B00

TSP? (Get Input Signal Position Value)

Description: Requests the current position of the selected input signal channel in physical units (µm).

Multiple input signal channels (sensors) could be involved in the control of one logical axis (see "Input Signal Processing" in the E727T0005 user manual). TSP? reads the position values for the individual input signal channels, not for a logical axis. To get the current position of an axis, use POS? (p. 81) instead.

Format: TSP? [{<InputSignalID>}]

Arguments: <InputSignalID> is one input signal channel of the controller

Response: {<InputSignalID>"="<float> LF}

where

<float> is the current position of the input signal channel, in physical units

TWC (Clear All Wave Related Triggers)

Description: Clears all output trigger settings for the wave generators (the settings made with TWS (p. 109)) by switching the signal state for all points to "low".

For a detailed description see "Wave Generator" and "Configuring Trigger Output" in the E727T0005 user manual.

Format: TWC

Arguments: None

Response: None

TWG? (Get Number of Wave Generators)

Description: Get the number of wave generators available on the controller.

Format: TWG?

Arguments: None

Response <uint> is the number of wave generators which are available

TWS (Set Trigger Line Action To Waveform Point)

Description: Defines trigger actions at certain waveform points for an output trigger line.

TWS provides the following actions for the output trigger line:

- Set the level (low or high) for individual waveform points
- Set rising and falling edges at certain waveform points. All points between two edges will then be set to the same level (low or high, depending on the edge types)

The power-on default state of all points is low. The signal state of the trigger output line can also be switched to "low" for all points using the TWC command (p. 108).

Generator Level Trigger mode or Generator Pulse Trigger mode must be activated for the selected trigger output line with the CTO command (p. 29).

See also "Wave Generator" and "Configuring Trigger Output" in the E727T0005 user manual.

Format: TWS {<TrigOutID> <PointNumber> <Switch>}

Arguments: <TrigOutID> is one digital output line of the controller, see below for details

<PointNumber> is one point in the waveform, starts with index 1
You can calculate the time for the point as follows:
time = generator cycle time * PointNumber
with
generator cycle time = Servo Update Time * WTR value
where
Servo Update Time in seconds is given by parameter 0x0E000200
WTR (wave table rate) value is the number of servo cycles the output of a waveform point takes, default is 1

<Switch> specifies the action, i.e., the desired signal state of the digital output line:
0 = low level

1 = high level

2 = rising edge

3 = falling edge

Response: None

Notes: <TrigOutID> corresponds to the output lines OUT1 to OUT3, IDs = 1 to 3; see "Digital I/O Socket" (p. 164).

Example 1: Send: TWS 2 1 1 2 2 0 2 3 0

Note: Sets trigger actions for the output line OUT2 (identifier 2), at waveform point 1 it is set high, points 2 and 3 are set low.

Example 2: Output line OUT1 (identifier 1) is to be used in “Generator Level Trigger” mode.

Send: CTO 1 3 4

Send: TWS 1 1 3 1 11250 2 1 35150 3

Note: For all waveform points from point 1 to point 11249, the output line is set low. At point 11250, there is a rising edge on the output line. From point 11251 to point 35149, the

output line is set high. At point 35150, there is a falling edge on the output line, and for all subsequent points the line will be set low.

TWS? (Get Trigger Line Action At Waveform Point)

Description: Reading of the trigger line settings made with TWS (p. 109) for the waveform points.

To query the waveform shape, use the GWD? command (p. 51).

See also "Wave Generator" and "Configuring Trigger Output" in the E727T0005 user manual.

Format: TWS? [<StartPoint> [<NumberOfPoints> [{<TrigOutID>}]]]

Arguments: <StartPoint> is the start point in the waveform, starts with index 1

<NumberOfPoints> is the number of points to be read per digital output line

<TrigOutID> is one digital output line of the controller

Response: The trigger settings (signal states) in GCS array format, see the separate manual for GCS array, SM 146E, and the example below.

Example: The trigger settings for the output lines OUT2 (identifier 2) and OUT3 (identifier 3) are queried for the waveform points 1 to 20. The response gives the signal states of the digital output lines at the individual waveform points:
0 = low, 1 = high

```
tws? 1 20 2 3
# TYPE = 1
# SEPARATOR = 9
# DIM = 2
# SAMPLE_TIME = 4.999999874e-05
# NDATA = 20
# END_HEADER
0       0
0       0
0       0
0       0
0       1
0       0
0       0
0       0
0       0
0       0
0       0
0       0
0       0
0       0
0       0
1       0
0       0
0       0
0       0
0       0
0       1
```

VAR (Set Variable Value)

Description: Sets a variable to a certain value.

Local variables can be set using VAR in macros only. See “Variables” in the E727T0005 user manual for details regarding local and global variables.

The variable is present in RAM only.

Format: VAR <Variable> <String>

Arguments: <Variable> is the name of the variable whose value is to be set.

<String> is the value to which the variable is to be set. If omitted, the variable is deleted.

The value can be given directly. In a macro, the value can also be given via the value of a variable.

See “Variables” in the E727T0005 user manual for conventions regarding variable names and values.

Response: None

Example: In a macro, it is possible to set the value of one variable (e.g. TARGET) to that of another variable (e.g. SOURCE):

```
VAR TARGET ${SOURCE}
```

Use curly brackets if the name of the variable is longer than one character:

```
VAR A ONE
VAR VARB TWO
VAR $A 1
```

VAR ${VARB} 2

VAR $VARB 2 // this will result in an unwanted behavior

VAR?

A=ONE

VARB=TWO

ONE=1

TWO=2 // ${VARB}: is replaced by its value “TWO”.

ARB=2 // $VARB: $V is replaced by its (empty) value.

See ADD (p. 17) for another example.

VAR? (Get Variable Values)

Description: Gets values of variables.

If VAR? is combined with CPY (p. 27), JRC (p. 70), MEX (p. 74) or WAC (p. 119), the response to VAR? has to be a single value and not more.

More information regarding local and global variables can be found in “Variables” in the E727T0005 user manual.

Format: VAR? [{<Variable>}]

Arguments: <Variable> is the name of the variable to be queried. More information on name conventions can be found in “Variables” in the E727T0005 user manual.

If <Variable> is omitted, all global variables present in the RAM are listed.

Response:	{<Variable>"="<String>LF} where <String> gives the value to which the variable is set.
Notes:	Local variables can be queried using VAR? only when a macro with local variables is running. See "Variables" in the E727T0005 user manual for details regarding local and global variables.
Example:	See ADD (p. 17) for an example.

VCO (Set Velocity Control Mode)

Description:	Sets the Velocity Control Mode of the specified axis to ON or OFF.
Format:	VCO {<AxisID> <VelCtrlState>}
Arguments:	<AxisID> is one axis of the controller <VelCtrlState> can have the following values: 0 = Velocity Control Mode OFF 1 = Velocity Control Mode ON (default)
Response:	None
Troubleshooting:	Illegal axis identifier
Notes:	When Velocity Control Mode is ON in closed-loop operation, the axis is driven with the specified slew rate (Servo Loop Slew-Rate parameter (ID 0x07000200)).

VCO? (Get Velocity Control Mode)

Description:	Gets Velocity Control Mode of given axes. If all arguments are omitted, gets mode of all axes.
Format:	VCO? [{<AxisID>}]
Arguments:	<AxisID> is one axis of the controller
Response:	{<AxisID>"="<VelCtrlState> LF} where <VelCtrlState> is the current Velocity Control Mode of the axis: 0 = Velocity Control Mode OFF 1 = Velocity Control Mode ON
Troubleshooting:	Illegal axis identifier

VEL (Set Velocity)

Description:	Set velocity of given axes. The VEL setting only takes effect when the given axis is in closed-loop operation (servo on). VEL can be changed while the axis is moving.
Format:	VEL {<AxisID> <Velocity>}
Arguments:	<AxisID> is one axis of the controller <Velocity> is the velocity value in physical units/s.
Response:	None
Troubleshooting:	Illegal axis identifiers, axis is under joystick control (via host PC)
Notes:	<Velocity> must be > 0.

VEL concerns the value of the Servo Loop Slew-Rate parameter, ID 0x07000200.

The velocity set with VEL is saved in volatile memory (RAM) only. To save the currently valid value to non-volatile memory, where it becomes the power-on default, you must use WPA (p. 137). Changes not saved with WPA will be lost when the controller is powered down. To have write access to the parameter, it might be necessary to switch to a higher command level using CCL (p. 26).

VEL? (Get Velocity)

Description: Get the current value of the closed-loop velocity.

If all arguments are omitted, gets current value of all axes.

Format: VEL? [{<AxisID>}]

Arguments: <AxisID> is one axis of the controller

Response: {<AxisID>"="<float> LF}

where

<float> is the current active velocity value in physical units / s.

Description: The current value of the closed-loop velocity is given by the Servo Loop Slew-Rate parameter, ID 0x07000200, in volatile memory.

VER? (Get Versions Of Firmware And Drivers)

Description: Gets the versions of the firmware of the E-727 as well as of further components like, for example, drivers and libraries.

Format: VER?

Arguments: None

Response {<string1>":" <string2> [<string3>]LF}

where

<string1> is the name of the component;
<string2> is the version information of the component <string1>;
<string3> is an optional note.

VOL? (Get Voltage Of Output Signal Channel)

Description: Read the current voltage value of the given output signal channel.

Multiple output signal channels (piezo amplifiers) can be involved in the motion of one logical axis (see "Output Generation" in the E727T0005 user manual). Note that VOL? reads the current voltage values for the individual output signal channels, not for a logical axis.

Format: VOL? [{<OutputSignalID>}]

Arguments: <OutputSignalID> is one output signal channel of the controller

Response: {<OutputSignalID>"="<float> LF}

where

<float> is the current voltage value in V

WAC (Wait For Condition)

Description: Waits until a given condition of the following type occurs: a specified value is compared with a queried value according a specified rule.

Can only be used in macros.

See also the MEX command (p. 74).

Format: WAC <CMD?> <OP> <value>

Arguments <CMD?> is one query command in its usual notation. The response has to be a single value and not more. For an example see below.

<OP> is the operator to be used. The following operators are possible:
= <= < > >= !=
Important: There must be a blank space before and after the operator!

<value> is the value to be compared with the response to <CMD?>.

Response: None

Example: Send:

```
MAC BEG LPMOTION
MVR 1 1
WAC ONT? 1 = 1
MVR 1 -1
WAC ONT? 1 = 1
MAC START LPMOTION
MAC END
MAC START LPMOTION
```

Note: Macro LPMOTION is first recorded and then started. WAC ONT? 1 = 1 waits until the response to ONT? 1 is 1=1. To form an infinite loop, the macro calls itself.

WAV (Set Waveform Definition)

Description: Define waveform of given type for given wave table.

To allow for flexible definition, a waveform (wave table contents) can be built up by adding "segments". Each segment is defined with a separate WAV command. To add a segment, the <AppendWave> argument (see below) is used to concatenate the new segment to the existing wave table contents. (To change individual segments later, or to modify their order, the complete waveform must be recreated segment-by-segment.)

A segment can be based on predefined "curve" shapes (see the <WaveType> argument below).

Waveforms cannot be changed while they are being output by a wave generator. If you want to modify a waveform with WAV, first stop any wave generator output from the associated wave table.

The waveform values are absolute values.

The frequency of the wave generator output depends, among other factors, on the wave table length. When you create waveforms, keep in mind that the usable frequency is limited by the available amplifier power. If the frequency is too high, overheating of the amplifier(s) can occur, and the piezo voltage output will be deactivated automatically.

The duration of one output cycle for the waveform can be calculated as follows:

Output Duration = Servo Update Time * WTR value * Number of Points

where

Servo Update Time in seconds is given by parameter 0x0E000200

WTR (wave table rate) value gives the number of servo cycles the output of a waveform point lasts, default is 1

Number of Points is the length of the wave table (which is the sum of the lengths of all segments in this table)

See "How to work with the Wave Generator" in the E727T0005 user manual for more information.

Format: WAV <WaveTableID> <AppendWave> <WaveType> <WaveTypeParameters>

Arguments: <WaveTableID> is the wave table identifier.

<AppendWave> This can be "X" or "&":
"X" clears the wave table and starts writing with the first point in the table.
"&" appends the defined segment to the already existing wave table contents (i.e. concatenates a segment to lengthen the waveform).

<WaveType> The type of curve used to define the segment. This can be one of
"PNT" (user-defined curve)
"NOISE" (white noise)
"SIN_P"(inverted cosine curve)
"RAMP" (ramp curve)
"LIN" (single scan line curve)

<WaveTypeParameters> stands for the parameters of the curve and can be as follows:

For "PNT":

<WaveStartPoint> <WaveLength> {<WavePoint>}

<WaveStartPoint> The index of the starting point. Must be 1.

<WaveLength> The length of the user-defined curve in points. The segment length, i.e. the number of points written to the wave table, is identical to the <WaveLength> value.

<WavePoint> The value of one single point.

For "NOISE":

<SegLength> <Amp> <Offset>

<SegLength>: The length of the wave table segment in points. If the white noise is to be used for a measurement, <SegLength> must be at least as long as the duration of the measurement.

<Amp>: The amplitude of the white noise. Difference between minimum and maximum value.

<Offset>: The offset of the white noise.

For "SIN_P":

<SegLength> <Amp> <Offset> <WaveLength> <StartPoint> <CurveCenterPoint>

<SegLength>: The length of the wave table segment in points. Only the number of points given by <SegLength> will be written to the wave table.

<Amp>: The amplitude of the sine curve.

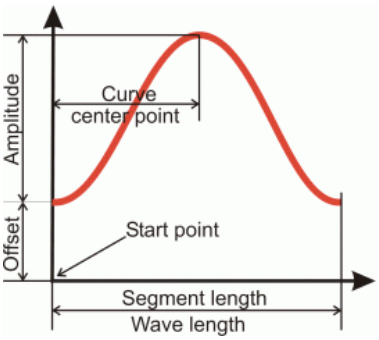
<Offset>: The offset of the sine curve.

<WaveLength>: The length of the sine curve in points.

<StartPoint>: The index of the starting point of the sine curve in the segment. Gives the phase shift. Lowest possible value is 0.

<CurveCenterPoint>: The index of the center point of the sine curve. Determines if the curve is symmetrical or not. Lowest possible value is 0.

Example (for more examples see "Defining Waveforms" in the E727T0005 user manual):



For "RAMP":

<SegLength> <Amp> <Offset> <WaveLength> <StartPoint> <SpeedUpDown> <CurveCenterPoint>

<SegLength>: The length of the wave table segment in points. Only the number of points given by <SegLength> will be written to the wave table.

<Amp>: The amplitude of the ramp curve.

<Offset>: The offset of the ramp curve.

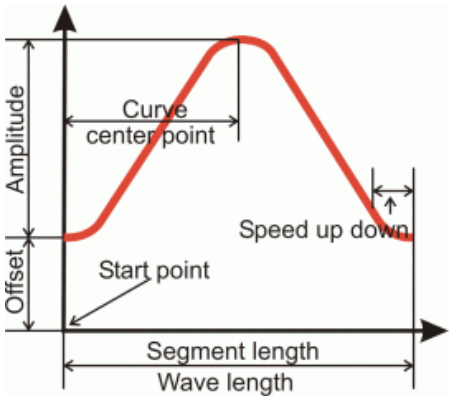
<WaveLength>: The length of the ramp curve in points.

<StartPoint>: The index of the starting point of the ramp curve in the segment. Gives the phase shift. Lowest possible value is 0.

<SpeedUpDown>: The number of points for speed-up and slow-down.

<CurveCenterPoint>: The index of the center point of the ramp curve. Determines if the curve is symmetrical or not. Lowest possible value is 0.

Example (for more examples see "Defining Waveforms" in the E727T0005 user manual):



For "LIN":

<SegLength> <Amp> <Offset> <WaveLength> <StartPoint> <SpeedUpDown>

<SegLength>: The length of the wave table segment in points. Only the number of points given by <SegLength> will be written to the wave table.

<Amp>: The amplitude of the scan line.

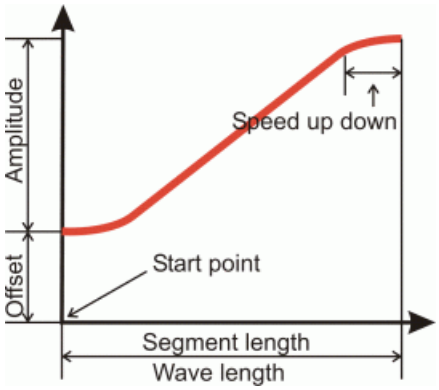
<Offset>: The offset of the scan line.

<WaveLength>: The length of the single scan line curve in points.

<StartPoint>: The index of the starting point of the scan line in the segment. Lowest possible value is 0.

<SpeedUpDown>: The number of points for speed-up and slow-down.

Example (for more examples see "Waveform Definition" in the E727T0005 user manual):



Note for the Sin_P, RAMP and LIN wave types:
If the <SegLength> value is larger than the <WaveLength> value, the missing points in the segment are filled with the endpoint value of the curve.

Response: None

Troubleshooting: Invalid wave table identifier

The total number of points for the waveform (which may consist of several segments) exceeds the available number of memory points.

WAV? (Get Waveform Definition)

Description: Get the value of a wave parameter for a given wave table.

See "How to work with the Wave Generator" in the E727T0005 user manual for more information.

Format: WAV? [{<WaveTableID> <WaveParameterID>}]

Arguments: <WaveTableID> is the wave table identifier.

<WaveParameterID> is the wave parameter ID, 1 = current wave table length in number of points; more parameters may be defined in the future

Response: {<WaveTableID> <WaveParameterID>"="<float> LF}

where

<float> depends on the <WaveParameterID>; gives the current number of waveform points in the wave table for <WaveParameterID> = 1

Troubleshooting: Invalid wave table identifier

WCL (Clear Wave Table Data)

Description: Clears the content of the given wave table.

As long as a wave generator is running, it is not possible to clear the connected wave table.

For a detailed description see "Wave Generator" in the E727T0005 user manual.

Format: WCL {<WaveTableID>}

Arguments: <WaveTableID> is the wave table identifier.

Response: None

WGC (Set Number Of Wave Generator Cycles)

Description: Sets the number of output cycles for the given wave generator (the output itself is started with WGO (p. 128)).

If the digital input line 2 is used to trigger the wave generator output (see WGO for details), the count of output cycles continues with each generator restart. The generator will be stopped when the number of cycles given by WGC are completed, irrespective of any further trigger pulses.

For a detailed description see "Wave Generator" in the E727T0005 user manual.

Format: WGC {<WaveGenID> <Cycles>}

Arguments: <WaveGenID> is the wave generator identifier

<Cycles> is the number of wave generator output cycles. If cycles = 0 then the waveform is output without period limitation until it is stopped by WGO or #24 (p. 15) or STP (p. 96).

Response: None

Notes: The specified number of cycles is always set for all wave generators, irrespective of the <WaveGenID> given in the WGC command.

WGC? (Get Number Of Wave Generator Cycles)

Description: Gets the number of output cycles set for the given wave generator.

For a detailed description see "Wave Generator" in the E727T0005 user manual.

Format: WGC? [{<WaveGenID>}]

Arguments: <WaveGenID> is the wave generator identifier

Response: {<WaveGenID>"="<Cycles> LF}

where

<Cycles> is the number of wave generator output cycles set with WGC (p. 127).

WGO (Set Wave Generator Start/Stop Mode)

Description: Start and stop the specified wave generator in the given mode. In addition, one data recording cycle is started.

The number of output cycles can be limited by WGC (p. 127).

Using the WTR command (p. 140), you can lengthen the individual output cycles of the waveform.

The data recorder configuration can be made with DRC (p. 41). Recording can be restarted with WGR (p. 134).

Keep in mind that wave generator output will continue even if the terminal or the program from which it was started is quit. The wave generator output will also continue if the high voltage output is automatically deactivated due to amplifier overheating. I.e. if a certain number of output cycles was set, the output may be already finished when the high voltage output is reactivated.

The #9 single-character command (p. 15) can be used to query the current activation state of the wave generators. The reply shows if a wave generator is running or not, but does not contain any information about the wave

generator start mode (e.g. with DDL). With WGO? you can ask for the last-commanded wave generator start options (WGO settings).

For more information see "Wave Generator" in the E727T0005 user manual.

For a detailed description of the DDL feature see "Dynamic Digital Linearization (DDL)" in the E727T0005 user manual.

Format: WGO {<WaveGenID> <StartMode>}

Arguments: <WaveGenID> is the wave generator identifier

<StartMode> is the start mode for the specified wave generator.
In the WGO command, you supply the start mode in hex or decimal format. When no bits are set (<StartMode> = 0), there is no wave generator output for the associated axis.
Note that bit 6 (0x40 or 64), bit 7 (0x80 or 128) and bit 8 (0x100 or 256) cannot start the wave generator output by themselves. They simply specify certain start options and must always be combined with one of the start modes specified in bit 0 (0x1 or 1) or bit 1 (0x2 or 2). See the examples below.

The start mode values in detail:

0: wave generator output is stopped. You can also use #24 (p. 15) or STP (p. 96) to stop the wave generator output, but WGO? (p. 133) will then still report the last commanded start mode.

bit 0 = 0x1 (hex format) or 1 (decimal format):
start wave generator output immediately, synchronized by servo cycle

bit 1 = 0x2 (hex format) or 2 (decimal format):
start wave generator output triggered by external

signal, synchronized by servo cycle.
To provide the external signal, the digital input lines IN1 or IN2 can be used (see pinout of the digital I/O socket (p. 164)).
If IN1 is used: The wave generator output starts with the first rising edge which is detected on this input line.
If IN2 is used: The generator output starts with the first rising edge which is detected on this input line, and it will be stopped when a falling edge is detected on this line. With the next rising edge, the generator output will continue at the waveform point where it was stopped. Starting and stopping the wave generator this way can be repeated indefinitely.
If output cycle limitations were made with WGC (p. 127): with each generator restart the count of output cycles continues, and the generator will be stopped when the given number of cycles are completed, irrespective of any further trigger pulses.
It is possible to mix the usage of both digital input lines.
For reliable triggering, the pulse width of the input signal has to be at least 2 x the servo update time of the E-727. The servo update time is given in seconds by parameter 0x0E000200.

bit 6 = 0x40 (hex format) or 64 (decimal format):
use and reinitialize DDL; start option.
The dynamic digital linearization (DDL) feature is used and reinitialized. It is recommended to start the DDL initialization for all axes at the same time. Each new initialization will stop all running initialization processes. The initialization process is also stopped by the DDL command (p. 34).

bit 7 = 0x80 (hex format) or 128 (decimal format):
use DDL; start option.
The dynamic digital linearization feature is used.

bit 8 = 0x100 (hex format) or 256 (decimal format):
wave generator started at the endpoint of the last

cycle; start option.
The second and all subsequent output cycles each start at the endpoint of the preceding cycle. The final position is the sum of the endpoint of the last output cycle and any offset defined with WAV (p. 120) for the waveform.

Response: None

Troubleshooting: Invalid wave generator identifier

There is no wave table connected to the wave generator. Use WSL (p. 139) to connect a wave table.

Wave generator output and analog control input:
It is possible to configure an axis for control by an analog input line while the wave generator output is active for that axis. In that case, the wave generator will continue running, but its output will no longer be used for control value generation. As long as the corresponding axis is set up to be commanded by analog control input, you can stop the wave generator output, but not restart it.

Wave generator output and move commands:
When the wave generator output is active, move commands like MOV (p. 75) or SVA (p. 97) are not allowed for the associated axis.

See "Control Value Generation" in the E727T0005 user manual for details.

Notes: All waveform output is synchronized because there is a common pulse generator used by all wave generators. For that reasons, wave tables which are supposed to run at the same time (each with one wave generator) should have the same length. If the wave tables have different lengths, an output cycle will comprise only the number of points given by the shortest table. This means that all waveform output is cut to the length of the shortest waveform currently running.

When the wave generator is to be started by an external trigger signal (WGO bit 1 is set), the value of the Wave Multi Start By Trigger parameter (ID 0x13000202)

determines if the trigger is enabled for only one generator start or for multiple starts:

0 = Trigger is enabled for only one generator start. Trigger becomes disabled after the generator has been started. To enable the trigger again, WGO must be sent again with start mode bit 1 set. Default setting.

1 = As long as WGO bit 1 is set, the trigger stays enabled for an unlimited number of generator starts. To disable the trigger, the wave generator output must be stopped with WGO, STP or #24.

Example 1: Wave generator 1 is to be used with the DDL feature, i.e. bit 7 on, contributing a value of 0x80 (dec.: 128) to <StartMode>. Because bit 7 is only a "start option" and does not actually start the wave generator output, a "start mode" ("immediately" or "triggered by external signal") must be chosen in addition. In this example, the wave generator is to be started by an external trigger signal, so bit 2 must be turned on, contributing 0x2 (dec.: 2), obtaining a <StartMode> value of 0x82 (dec.: 130).

Send the following WGO command, with the <StartMode> given in hex format:

WGO 1 0x82

The same command with <StartMode> given in decimal format:

WGO 1 130

Example 2: Wave generator 1 is to be started with the "Use and reinitialize DDL" option (bit 6, value 0x40; dec.: 64). Furthermore the option "start at the endpoint of the last cycle" is to be active (bit 8, value 0x100; dec.: 256). The start mode is to be "immediately" (bit 0, value 0x1; dec.: 1). Hence the resulting <StartMode> value is in hex format

0x40 + 0x100 + 0x1 = 0x141

or in dec format

64 + 256 + 1 = 321

Send

WGO 1 0x141

or

WGO 1 321

WGO? (Get Wave Generator Start/Stop Mode)

Description: Get the start/stop mode of the given wave generator.

The #9 single-character command (p. 15) can be used to query the current activation state of the wave generators. The reply shows if a wave generator is running or not, but does not contain any information about the wave generator start mode (e.g. with DDL). With WGO? you can ask for the last-commanded wave generator start options (WGO settings (p. 128)).

Note that #24 (p. 15) or STP (p. 97) stop the wave generator output, but do not reset the start/stop mode settings so that WGO? will still report the start mode which was set by the last WGO command (p. 128).

For more information see "Wave Generator" in the E727T0005 user manual.

Format: WGO? [{<WaveGenID>}]

Arguments: <WaveGenID> is the wave generator identifier

Response: {<WaveGenID>"="<StartMode> LF}

where

<StartMode> is the last commanded start mode of the wave generator, in decimal format. The value may be the sum of several start options and one start mode. See the WGO command description for details.

WGR (Starts Recording In Sync With Wave Generator)

Description: Restarts recording when the wave generator is running (a first data recording cycle is started with the WGO command (p. 128) which starts the wave generator output).

The data recorder configuration can be made with DRC (p. 41). The recorded data can be read with the DRR? command (p. 45).

For more information see "Wave Generator" and "Data Recording" in the E727T0005 user manual.

Format: WGR

Arguments: None

Response: None

WOS (Set Wave Generator Output Offset)

Description: Sets an offset to the output of a wave generator. The current wave generator output is then created by adding the offset value to the current wave value:

Generator Output = Offset + Current Wave Value

Do not confuse the output-offset value set with WOS with the offset settings specified during waveform creation with

WAV (p. 120). While the WAV offset affects only one segment (i.e. only one waveform), the WOS offset is added to all waveforms which are output by the given wave generator.

WOS sets the value of the Wave Offset parameter, ID 0x1300010b, in volatile memory. You can change this parameter also with SPA (p. 90) or SEP (p. 87) and save the value with WPA (p. 137) to non-volatile memory, where it becomes the power-on default. To have write access to the parameter, it might be necessary to switch to a higher command level using CCL (p. 26).

If the wave generator is started with the option "start at the endpoint of the last cycle", the E-727 at the end of each output cycle equates the volatile value of the Wave Offset parameter with the current generator output.

Deleting wave table content with WCL (p. 126) has no effect on the settings for the wave generator output offset.

For more information see "Wave Generator" in the E727T0005 user manual.

Format: WOS {<WaveGenID> <Offset>}

Arguments: <WaveGenID> is the wave generator identifier

<Offset> is the wave generator output offset, any float number.
In closed-loop operation (servo ON), the offset is interpreted as position value in either case. In open-loop operation (servo OFF), the interpretation of the offset depends on the settings of the output matrix (see "Output Generation" in the E727T0005 user manual for more information). With the default matrix coefficients, open-loop control values numerically correspond to axis position values.

Response: None

WOS? (Get Wave Generator Output Offset)

Description: Reads the current value of the offset which is added to the wave generator output (Wave Offset parameter value in volatile memory (ID 0x1300010b)). This value results either from WOS (p. 134) / SPA (p. 90) / SEP (p. 87) settings, or from internal calculation during the wave generator output; see WOS for details.

For more information see also "Wave Generator" in the E727T0005 user manual.

Format: WOS? [{<WaveGenID>}]

Arguments: <WaveGenID> is the wave generator identifier

Response: {<WaveGenID>"="<Offset> LF}

where

<Offset> is the current wave generator output offset. In closed-loop operation (servo ON), the offset is interpreted as position value in either case. In open-loop operation (servo OFF), the interpretation of the offset depends on the settings of the output matrix (see "Output Generation" in the E727T0005 user manual for more information). With the default matrix coefficients, open-loop control values numerically correspond to axis position values.

WPA (Save Parameters To Non-Volatile Memory)

Description: Write the currently valid value of a parameter of a given item from volatile memory (RAM) to non-volatile memory. The values saved this way become the power-on defaults.

NOTICE: If current parameter values are incorrect, the system may malfunction. Be sure that you have the correct parameter settings before using the WPA command.

RAM settings not saved with WPA will be lost when the controller is powered down or rebooted or when RPA (p. 83) is used to restore the parameters.

With HPA? (p. 56) you can obtain a list of all available parameters.

Use SPA? (p. 90) to check the current parameter settings in volatile memory.

See SPA (p. 90) for an example.

Format: WPA <Pswd> [{<ItemID> <PamID>}]

Arguments: <Pswd> is the password for writing to non-volatile memory. See below for details.

<ItemID> is the item for which a parameter is to be saved from volatile to non-volatile memory. See below for details.

<PamID> is the parameter ID, can be written in hexadecimal or decimal format. See below for details.

Response: None

Troubleshooting: Illegal item identifier, wrong parameter ID, invalid password, command level too low for write access

Notes:

Parameters can be changed in volatile memory with SPA (p. 90), AOS (p. 19), ATZ (p. 22), CTO (p. 29), DRC (p. 41), DPO (p. 40), IFC (p. 59), RTR (p. 84), VEL (p. 116), WOS (p. 134) and WTR (p. 140). Furthermore, the value of the IP Address parameter (ID 0x11000600) will be changed automatically in volatile memory when a TCP/IP connection is established and the IP address is obtained from DHCP server.

When WPA is used without specifying any arguments except of the password, all currently valid parameter values are saved.

To have write access to the parameter(s), it might be necessary to switch to a higher command level using CCL (p. 26).

While parameter values are being read from or written to the non-volatile memory of the E-727, the servo is temporarily switched off. This may lead to faulty control or large position errors.

Observe the following when you use the commands SEP, SEP?, RPA, WPA:

- Avoid motions of the axes.

- Do not carry out measurements with the axes.

NOTICE: The number of write cycles of non-volatile memory is limited. Write default values only when necessary.

Available passwords, item IDs and parameter IDs:	The password for writing to non-volatile memory is "100". The item type depends on the parameter, see "Parameter Overview" in the E727T0005 user manual for the item type concerned. See "Axes, Channels, Functional Elements" in the E727T0005 user manual for the identifiers of the items. Valid parameter IDs are given in "Parameter Overview" in the E727T0005 user manual.

WSL (Set Connection Of Wave Table To Wave Generator)

Description:	Wave table selection: connects a wave table to a wave generator or disconnects the selected generator from any wave table. Two or more generators can be connected to the same wave table, but a generator cannot be connected to more than one wave table. Deleting wave table content with WCL (p. 126) has no effect on the WSL settings. As long as a wave generator is running, it is not possible to change its wave table connection. For more information see "Wave Generator" in the E727T0005 user manual.
Format:	WSL {<WaveGenID> <WaveTableID>}
Arguments:	<WaveGenID> is the wave generator identifier <WaveTableID> is the wave table identifier. If <WaveTableID> = 0, the selected generator is disconnected from any wave table.
Response:	None

WSL? (Get Connection Of Wave Table To Wave Generator)

Description: Get current wave table connection settings for the specified wave generator.

For more information see "Wave Generator" in the E727T0005 user manual.

Format: WSL? [{<WaveGenID>}]

Arguments: <WaveGenID> is the wave generator identifier

Response: {<WaveGenID>"="<WaveTableID> LF}

where

<WaveTableID> is the wave table identifier. If <WaveTableID> = 0, no wave table is connected to the wave generator.

WTR (Set Wave Generator Table Rate)

Description: Set wave generator table rate and interpolation type:

Using the WTR command, you can lengthen the individual output cycles of the waveform. The duration of one output cycle for the waveform can be calculated as follows:

Output Duration = Servo Update Time * WTR value * Number of Points

where

Servo Update Time is given in seconds by parameter 0x0E000200

WTR value gives the number of servo cycles the output of a waveform point lasts, default is 1

Number of Points is the length of the waveform (i.e. the length of the wave table)

WTR sets the value of the Wave Generator Table Rate

parameter, ID 0x13000109, in volatile memory. You can change this parameter also with SPA (p. 90) or SEP (p. 87) and save the value to non-volatile memory with WPA (p. 137) (switch to command level 1 before with the CCL command). The value is always valid for the whole system and cannot be set separately for individual wave generators. The value of the parameter in volatile memory can be read with the WTR? command (p. 140).

WTR also sets the type of interpolation to use for the wave generator output. If the Wave Generator Table Rate is larger than 1, interpolation helps to avoid sudden position jumps of an axis controlled by the wave generator.

For more information see "Wave Generator" in the E727T0005 user manual. An application example can be found in "Modifying the Wave Generator Table Rate" in the E727T0005 user manual.

Format: WTR {<WaveGenID> <WaveTableRate> <InterpolationType>}

Arguments: <WaveGenID> is the wave generator identifier, must be zero which means that all wave generators are selected.

<WaveTableRate> is the table rate to be used for wave generator output (unit: number of servo-loop cycles), must be an integer value greater than zero

<InterpolationType> When a wave generator table rate higher than 1 is set, this option can be used to apply interpolation to the wave generator output between wave table points. The following interpolation types can be selected:
0 = no interpolation
1 = straight line (default)

Response: None

WTR? (Get Wave Generator Table Rate)

Description: Gets the current wave generator table rate, i.e. the number of servo-loop cycles used by the wave generator to output one waveform point (Wave Generator Table Rate parameter value in volatile memory (ID 0x13000109)). Gets also the interpolation type used with table rate values > 1.

For more information see "Wave Generator" in the E727T0005 user manual. An application example can be found in "Modifying the Wave Generator Table Rate" in the E727T0005 user manual.

Format: WTR? [{<WaveGenID>}]

Arguments: <WaveGenID> is the wave generator identifier; if zero, all wave generators are queried

Response: {<WaveGenID>"="<WaveTableRate> <InterpolationType> LF}

where

<WaveTableRate> is the table rate used for wave generator output (unit: number of servo-loop cycles)

<InterpolationType> interpolation type applied to outputs between wave table points when a wave generator table rate higher than 1 is set:
0 = no interpolation
1 = straight line

2.4 Error Codes

The error codes listed here are those of the PI General Command Set. As such, some may be not relevant to your controller and will simply never occur.

2.4.1 Controller errors

0	PI_CNTR_NO_ERROR	No error
1	PI_CNTR_PARAM_SYNTAX	Parameter syntax error
2	PI_CNTR_UNKNOWN_COMMAND	Unknown command
3	PI_CNTR_COMMAND_TOO_LONG	Command length out of limits or command buffer overrun
4	PI_CNTR_SCAN_ERROR	Error while scanning
5	PI_CNTR_MOVE_WITHOUT_REF_OR_NO_SERVO	Unallowable move attempted on unreferenced axis, or move attempted with servo off
6	PI_CNTR_INVALID_SGA_PARAM	Parameter for SGA not valid
7	PI_CNTR_POS_OUT_OF_LIMITS	Position out of limits
8	PI_CNTR_VEL_OUT_OF_LIMITS	Velocity out of limits
9	PI_CNTR_SET_PIVOT_NOT_POSSIBLE	Attempt to set pivot point while U,V and W not all 0
10	PI_CNTR_STOP	Controller was stopped by command
11	PI_CNTR_SST_OR_SCAN_RANGE	Parameter for SST or for one of the embedded scan algorithms out of range
12	PI_CNTR_INVALID_SCAN_AXES	Invalid axis combination for fast scan
13	PI_CNTR_INVALID_NAV_PARAM	Parameter for NAV out of range
14	PI_CNTR_INVALID_ANALOG_INPUT	Invalid analog channel
15	PI_CNTR_INVALID_AXIS_IDENTIFIER	Invalid axis identifier
16	PI_CNTR_INVALID_STAGE_NAME	Unknown stage name
17	PI_CNTR_PARAM_OUT_OF_RANGE	Parameter out of range
18	PI_CNTR_INVALID_MACRO_NAME	Invalid macro name
19	PI_CNTR_MACRO_RECORD	Error while recording macro
20	PI_CNTR_MACRO_NOT_FOUND	Macro not found

21	PI_CNTR_AXIS_HAS_NO_BRAKE	Axis has no brake
22	PI_CNTR_DOUBLE_AXIS	Axis identifier specified more than once
23	PI_CNTR_ILLEGAL_AXIS	Illegal axis
24	PI_CNTR_PARAM_NR	Incorrect number of parameters
25	PI_CNTR_INVALID_REAL_NR	Invalid floating point number
26	PI_CNTR_MISSING_PARAM	Parameter missing
27	PI_CNTR_SOFT_LIMIT_OUT_OF_RANGE	Soft limit out of range
28	PI_CNTR_NO_MANUAL_PAD	No manual pad found
29	PI_CNTR_NO_JUMP	No more step-response values
30	PI_CNTR_INVALID_JUMP	No step-response values recorded
31	PI_CNTR_AXIS_HAS_NO_REFERENCE	Axis has no reference sensor
32	PI_CNTR_STAGE_HAS_NO_LIM_SWITCH	Axis has no limit switch
33	PI_CNTR_NO_RELAY_CARD	No relay card installed
34	PI_CNTR_CMD_NOT_ALLOWED_FOR_STAGE	Command not allowed for selected stage(s)
35	PI_CNTR_NO_DIGITAL_INPUT	No digital input installed
36	PI_CNTR_NO_DIGITAL_OUTPUT	No digital output configured
37	PI_CNTR_NO_MCM	No more MCM responses
38	PI_CNTR_INVALID_MCM	No MCM values recorded
39	PI_CNTR_INVALID_CNTR_NUMBER	Controller number invalid
40	PI_CNTR_NO_JOYSTICK_CONNECTED	No joystick configured
41	PI_CNTR_INVALID_EGE_AXIS	Invalid axis for electronic gearing, axis can not be slave
42	PI_CNTR_SLAVE_POSITION_OUT_OF_RANGE	Position of slave axis is out of range
43	PI_CNTR_COMMAND_EGE_SLAVE	Slave axis cannot be commanded directly when electronic gearing is enabled
44	PI_CNTR_JOYSTICK_CALIBRATION_FAILED	Calibration of joystick failed
45	PI_CNTR_REFERENCING_FAILED	Referencing failed
46	PI_CNTR_OPM_MISSING	OPM (Optical Power Meter) missing

47	PI_CNTR_OPM_NOT_INITIALIZED	OPM (Optical Power Meter) not initialized or cannot be initialized
48	PI_CNTR_OPM_COM_ERROR	OPM (Optical Power Meter) Communication Error
49	PI_CNTR_MOVE_TO_LIMIT_SWITCH_FAILED	Move to limit switch failed
50	PI_CNTR_REF_WITH_REF_DISABLED	Attempt to reference axis with referencing disabled
51	PI_CNTR_AXIS_UNDER_JOYSTICK_CONTROL	Selected axis is controlled by joystick
52	PI_CNTR_COMMUNICATION_ERROR	Controller detected communication error
53	PI_CNTR_DYNAMIC_MOVE_IN_PROCESS	MOV! motion still in progress
54	PI_CNTR_UNKNOWN_PARAMETER	Unknown parameter
55	PI_CNTR_NO_REP_RECORDED	No commands were recorded with REP
56	PI_CNTR_INVALID_PASSWORD	Password invalid
57	PI_CNTR_INVALID_RECORDER_CHAN	Data Record Table does not exist
58	PI_CNTR_INVALID_RECORDER_SRC_OPT	Source option does not exist; number too low or too high
59	PI_CNTR_INVALID_RECORDER_SRC_CHAN	Source ID (channel or axis) too low or too high
60	PI_CNTR_PARAM_PROTECTION	Protected Param: current Command Level (CCL) too low
61	PI_CNTR_AUTOZERO_RUNNING	Command execution not possible while Autozero is running
62	PI_CNTR_NO_LINEAR_AXIS	Autozero requires at least one linear axis
63	PI_CNTR_INIT_RUNNING	Initialization still in progress
64	PI_CNTR_READ_ONLY_PARAMETER	Parameter is read-only
65	PI_CNTR_PAM_NOT_FOUND	Parameter not found in non-volatile memory
66	PI_CNTR_VOL_OUT_OF_LIMITS	Voltage out of limits
67	PI_CNTR_WAVE_TOO_LARGE	Not enough memory available for requested wave curve
68	PI_CNTR_NOT_ENOUGH_DDL_MEMORY	Not enough memory available for DDL table; DDL can not be started
69	PI_CNTR_DDL_TIME_DELAY_TOO_LARGE	Time delay larger than DDL table; DDL can not be started
70	PI_CNTR_DIFFERENT_ARRAY_LENGTH	The requested arrays have different lengths; query them separately
71	PI_CNTR_GEN_SINGLE_MODE_RESTART	Attempt to restart the generator while it is running in single step mode
72	PI_CNTR_ANALOG_TARGET_ACTIVE	Motion commands and wave generator activation are not allowed

		when analog target is active
73	PI_CNTR_WAVE_GENERATOR_ACTIVE	Motion commands are not allowed when wave generator is active
74	PI_CNTR_AUTOZERO_DISABLED	No sensor channel or no piezo channel connected to selected axis (sensor and piezo matrix)
75	PI_CNTR_NO_WAVE_SELECTED	Generator started (WGO) without having selected a wave table (WSL).
76	PI_CNTR_IF_BUFFER_OVERRUN	Interface buffer did overrun and command couldn't be received correctly
77	PI_CNTR_NOT_ENOUGH_RECORDED_DATA	Data Record Table does not hold enough recorded data
78	PI_CNTR_TABLE_DEACTIVATED	Data Record Table is not configured for recording
79	PI_CNTR_OPENLOOP_VALUE_SET_WHEN_SERVO_ON	Open-loop commands (SVA, SVR) are not allowed when servo is on
80	PI_CNTR_RAM_ERROR	Hardware error affecting RAM
81	PI_CNTR_MACRO_UNKNOWN_COMMAND	Not macro command
82	PI_CNTR_MACRO_PC_ERROR	Macro counter out of range
83	PI_CNTR_JOYSTICK_ACTIVE	Joystick is active
84	PI_CNTR_MOTOR_IS_OFF	Motor is off
85	PI_CNTR_ONLY_IN_MACRO	Macro-only command
86	PI_CNTR_JOYSTICK_UNKNOWN_AXIS	Invalid joystick axis
87	PI_CNTR_JOYSTICK_UNKNOWN_ID	Joystick unknown
88	PI_CNTR_REF_MODE_IS_ON	Move without referenced stage
89	PI_CNTR_NOT_ALLOWED_IN_CURRENT_MOTION_MODE	Command not allowed in current motion mode
90	PI_CNTR_DIO_AND_TRACING_NOT_POSSIBLE	No tracing possible while digital IOs are used on this HW revision. Reconnect to switch operation mode.
91	PI_CNTR_COLLISION	Move not possible, would cause collision
92	PI_CNTR_SLAVE_NOT_FAST_ENOUGH	Stage is not capable of following the master. Check the gear ratio.
93	PI_CNTR_CMD_NOT_ALLOWED_WHILE_AXIS_IN_MOTION	This command is not allowed while the affected axis or its master is in motion.
94	PI_CNTR_OPEN_LOOP_JOYSTICK_ENABLED	Servo cannot be switched on when open-loop joystick control is enabled.
95	PI_CNTR_INVALID_SERVO_STATE_FOR_PARAMETER	This parameter cannot be changed in current servo mode.

96	PI_CNTR_UNKNOWN_STAGE_NAME	Unknown stage name
97	PI_CNTR_INVALID_VALUE_LENGTH	Invalid length of value (too much characters)
98	PI_CNTR_AUTOZERO_FAILED	AutoZero procedure was not successful
99	PI_CNTR_SENSOR_VOLTAGE_OFF	Sensor voltage is off
100	PI_LABVIEW_ERROR	PI LabVIEW driver reports error. See source control for details.
200	PI_CNTR_NO_AXIS	No stage connected to axis
201	PI_CNTR_NO_AXIS_PARAM_FILE	File with axis parameters not found
202	PI_CNTR_INVALID_AXIS_PARAM_FILE	Invalid axis parameter file
203	PI_CNTR_NO_AXIS_PARAM_BACKUP	Backup file with axis parameters not found
204	PI_CNTR_RESERVED_204	PI internal error code 204
205	PI_CNTR_SMO_WITH_SERVO_ON	SMO with servo on
206	PI_CNTR_UUDECODE_INCOMPLETE_HEADER	uudecode: incomplete header
207	PI_CNTR_UUDECODE_NOTHING_TO_DECODE	uudecode: nothing to decode
208	PI_CNTR_UUDECODE_ILLEGAL_FORMAT	uudecode: illegal UUE format
209	PI_CNTR_CRC32_ERROR	CRC32 error
210	PI_CNTR_ILLEGAL_FILENAME	Illegal file name (must be 8-0 format)
211	PI_CNTR_FILE_NOT_FOUND	File not found on controller
212	PI_CNTR_FILE_WRITE_ERROR	Error writing file on controller
213	PI_CNTR_DTR_HINDERS_VELOCITY_CHANGE	VEL command not allowed in DTR Command Mode
214	PI_CNTR_POSITION_UNKNOWN	Position calculations failed
215	PI_CNTR_CONN_POSSIBLY_BROKEN	The connection between controller and stage may be broken
216	PI_CNTR_ON_LIMIT_SWITCH	The connected stage has driven into a limit switch, some controllers need CLR to resume operation
217	PI_CNTR_UNEXPECTED_STRUT_STOP	Strut test command failed because of an unexpected strut stop
218	PI_CNTR_POSITION_BASED_ON_ESTIMATION	While MOV! is running position can only be estimated!
219	PI_CNTR_POSITION_BASED_ON_INTERPOLATION	Position was calculated during MOV motion
220	PI_CNTR_INTERPOLATION_FIFO_UNDERRUN	FIFO buffer underrun during interpolation

221	PI_CNTR_INTERPOLATION_FIFO_OVERFLOW	FIFO buffer overflow during interpolation
230	PI_CNTR_INVALID_HANDLE	Invalid handle
231	PI_CNTR_NO_BIOS_FOUND	No bios found
232	PI_CNTR_SAVE_SYS_CFG_FAILED	Save system configuration failed
233	PI_CNTR_LOAD_SYS_CFG_FAILED	Load system configuration failed
301	PI_CNTR_SEND_BUFFER_OVERFLOW	Send buffer overflow
302	PI_CNTR_VOLTAGE_OUT_OF_LIMITS	Voltage out of limits
303	PI_CNTR_OPEN_LOOP_MOTION_SET_WHEN_SERVO_ON	Open-loop motion attempted when servo ON
304	PI_CNTR_RECEIVING_BUFFER_OVERFLOW	Received command is too long
305	PI_CNTR_EEPROM_ERROR	Error while reading/writing EEPROM
306	PI_CNTR_I2C_ERROR	Error on I2C bus
307	PI_CNTR_RECEIVING_TIMEOUT	Timeout while receiving command
308	PI_CNTR_TIMEOUT	A lengthy operation has not finished in the expected time
309	PI_CNTR_MACRO_OUT_OF_SPACE	Insufficient space to store macro
310	PI_CNTR_EUI_OLDVERSION_CFGDATA	Configuration data has old version number
311	PI_CNTR_EUI_INVALID_CFGDATA	Invalid configuration data
333	PI_CNTR_HARDWARE_ERROR	Internal hardware error
400	PI_CNTR_WAV_INDEX_ERROR	Wave generator index error
401	PI_CNTR_WAV_NOT_DEFINED	Wave table not defined
402	PI_CNTR_WAV_TYPE_NOT_SUPPORTED	Wave type not supported
403	PI_CNTR_WAV_LENGTH_EXCEEDS_LIMIT	Wave length exceeds limit
404	PI_CNTR_WAV_PARAMETER_NR	Wave parameter number error
405	PI_CNTR_WAV_PARAMETER_OUT_OF_LIMIT	Wave parameter out of range
406	PI_CNTR_WGO_BIT_NOT_SUPPORTED	WGO command bit not supported
500	PI_CNTR_EMERGENCY_STOP_BUTTON_ACTIVATED	The \"red knob\" is still set and disables system
501	PI_CNTR_EMERGENCY_STOP_BUTTON_WAS_ACTIVATED	The \"red knob\" was activated and still disables system - reanimation required

502	PI_CNTR_REDUNDANCY_LIMIT_EXCEEDED	Position consistency check failed
503	PI_CNTR_COLLISION_SWITCH_ACTIVATED	Hardware collision sensor(s) are activated
504	PI_CNTR_FOLLOWING_ERROR	Strut following error occurred, e.g. caused by overload or encoder failure
505	PI_CNTR_SENSOR_SIGNAL_INVALID	One sensor signal is not valid
506	PI_CNTR_SERVO_LOOP_UNSTABLE	Servo loop was unstable due to wrong parameter setting and switched off to avoid damage.
507	PI_CNTR_LOST_SPI_SLAVE_CONNECTION	Digital connection to external SPI slave device is lost
508	PI_CNTR_MOVE_ATTEMPT_NOT_PERMITTED	Move attempt not permitted due to customer or limit settings
509	PI_CNTR_TRIGGER_EMERGENCY_STOP	Emergency stop caused by trigger input
530	PI_CNTR_NODE_DOES_NOT_EXIST	A command refers to a node that does not exist
531	PI_CNTR_PARENT_NODE_DOES_NOT_EXIST	A command refers to a node that has no parent node
532	PI_CNTR_NODE_IN_USE	Attempt to delete a node that is in use
533	PI_CNTR_NODE_DEFINITION_IS_CYCLIC	Definition of a node is cyclic
536	PI_CNTR_HEXAPOD_IN_MOTION	Transformation cannot be defined as long as Hexapod is in motion
537	PI_CNTR_TRANSFORMATION_TYPE_NOT_SUPPORTED	Transformation node cannot be activated
539	PI_CNTR_NODE_PARENT_IDENTICAL_TO_CHILD	A node cannot be linked to itself
540	PI_CNTR_NODE_DEFINITION_INCONSISTENT	Node definition is erroneous or not complete (replace or delete it)
542	PI_CNTR_NODES_NOT_IN_SAME_CHAIN	The nodes are not part of the same chain
543	PI_CNTR_NODE_MEMORY_FULL	Unused nodes must be deleted before new nodes can be stored
544	PI_CNTR_PIVOT_POINT_FEATURE_NOT_SUPPORTED	With some transformations pivot point usage is not supported
545	PI_CNTR_SOFTLIMITS_INVALID	Soft limits invalid due to changes in coordinate system
546	PI_CNTR_CS_WRITE_PROTECTED	Coordinate system is write protected
547	PI_CNTR_CS_CONTENT_FROM_CONFIG_FILE	Coordinate system cannot be changed because its content is loaded from a configuration file
548	PI_CNTR_CS_CANNOT_BE_LINKED	Coordinate system may not be linked
549	PI_CNTR_KSB_CS_ROTATION_ONLY	A KSB-type coordinate system can only be rotated by multiples of 90 degrees
551	PI_CNTR_CS_DATA_CANNOT_BE_QUERIED	This query is not supported for this coordinate system type

552	PI_CNTR_CS_COMBINATION_DOES_NOT_EXIST	This combination of work-and-tool coordinate systems does not exist
553	PI_CNTR_CS_COMBINATION_INVALID	The combination must consist of one work and one tool coordinate system
554	PI_CNTR_CS_TYPE_DOES_NOT_EXIST	This coordinate system type does not exist
555	PI_CNTR_UNKNOWN_ERROR	BasMac: unknown controller error
556	PI_CNTR_CS_TYPE_NOT_ACTIVATED	No coordinate system of this type is activated
557	PI_CNTR_CS_NAME_INVALID	Name of coordinate system is invalid
558	PI_CNTR_CS_GENERAL_FILE_MISSING	File with stored CS systems is missing or erroneous
559	PI_CNTR_CS_LEVELING_FILE_MISSING	File with leveling CS is missing or erroneous
601	PI_CNTR_NOT_ENOUGH_MEMORY	not enough memory
602	PI_CNTR_HW_VOLTAGE_ERROR	hardware voltage error
603	PI_CNTR_HW_TEMPERATURE_ERROR	hardware temperature out of range
604	PI_CNTR_POSITION_ERROR_TOO_HIGH	Position error of any axis in the system is too high
606	PI_CNTR_INPUT_OUT_OF_RANGE	Maximum value of input signal has been exceeded
607	PI_CNTR_NO_INTEGER	Value is not integer
608	PI_CNTR_FAST_ALIGNMENT_PROCESS_IS_NOT_RUNNING	Fast alignment process cannot be paused because it is not running
609	PI_CNTR_FAST_ALIGNMENT_PROCESS_IS_NOT_PAUSED	Fast alignment process cannot be restarted/resumed because it is not paused
650	PI_CNTR_UNABLE_TO_SET_PARAM_WITH_SPA	Parameter could not be set with SPA - SEP needed?
651	PI_CNTR_PHASE_FINDING_ERROR	Phase finding error
652	PI_CNTR_SENSOR_SETUP_ERROR	Sensor setup error
653	PI_CNTR_SENSOR_COMM_ERROR	Sensor communication error
654	PI_CNTR_MOTOR_AMPLIFIER_ERROR	Motor amplifier error
655	PI_CNTR_OVER_CURR_PROTEC_TRIGGERED_BY_I2T	Overcurrent protection triggered by I2T-module
656	PI_CNTR_OVER_CURR_PROTEC_TRIGGERED_BY_AMP_MODULE	Overcurrent protection triggered by amplifier module
657	PI_CNTR_SAFETY_STOP_TRIGGERED	Safety stop triggered
658	PI_SENSOR_OFF	Sensor off?

700	PI_CNTR_COMMAND_NOT_ALLOWED_IN_EXTERNAL_MODE	Command not allowed in external mode
710	PI_CNTR_EXTERNAL_MODE_ERROR	External mode communication error
715	PI_CNTR_INVALID_MODE_OF_OPERATION	Invalid mode of operation
716	PI_CNTR_FIRMWARE_STOPPED_BY_CMD	Firmware stopped by command (#27)
717	PI_CNTR_EXTERNAL_MODE_DRIVER_MISSING	External mode driver missing
718	PI_CNTR_CONFIGURATION_FAILURE_EXTERNAL_MODE	Missing or incorrect configuration of external mode
719	PI_CNTR_EXTERNAL_MODE_CYCLETIME_INVALID	External mode cycletime invalid
720	PI_CNTR_BRAKE_ACTIVATED	Brake is activated
731	PI_CNTR_SURFACEDETECTION_RUNNING	Command not allowed while surface detection is running
732	PI_CNTR_SURFACEDETECTION_FAILED	Last surface detection failed
733	PI_CNTR_FIELDBUS_IS_ACTIVE	Fieldbus is active and is blocking GCS control commands
1000	PI_CNTR_TOO_MANY_NESTED_MACROS	Too many nested macros
1001	PI_CNTR_MACRO_ALREADY_DEFINED	Macro already defined
1002	PI_CNTR_NO_MACRO_RECORDING	Macro recording not activated
1003	PI_CNTR_INVALID_MAC_PARAM	Invalid parameter for MAC
1004	PI_CNTR_RESERVED_1004	PI internal error code 1004
1005	PI_CNTR_CONTROLLER_BUSY	Controller is busy with some lengthy operation (e.g. reference move, fast scan algorithm)
1006	PI_CNTR_INVALID_IDENTIFIER	Invalid identifier (invalid special characters, ...)
1007	PI_CNTR_UNKNOWN_VARIABLE_OR_ARGUMENT	Variable or argument not defined
1008	PI_CNTR_RUNNING_MACRO	Controller is (already) running a macro
1009	PI_CNTR_MACRO_INVALID_OPERATOR	Invalid or missing operator for condition. Check necessary spaces around operator.
1010	PI_CNTR_MACRO_NO_ANSWER	No answer was received while executing WAC/MEX/JRC/...
1011	PI_CMD_NOT_VALID_IN_MACRO_MODE	Command not valid during macro execution
1024	PI_CNTR_MOTION_ERROR	Motion error: position error too large, servo is switched off automatically
1025	PI_CNTR_MAX_MOTOR_OUTPUT_REACHED	Maximum motor output reached

1063	PI_CNTR_EXT_PROFILE_UNALLOWED_CMD	User Profile Mode: Command is not allowed, check for required preparatory commands
1064	PI_CNTR_EXT_PROFILE_EXPECTING_MOTION_ERROR	User Profile Mode: First target position in User Profile is too far from current position
1065	PI_CNTR_PROFILE_ACTIVE	Controller is (already) in User Profile Mode
1066	PI_CNTR_PROFILE_INDEX_OUT_OF_RANGE	User Profile Mode: Block or Data Set index out of allowed range
1071	PI_CNTR_PROFILE_OUT_OF_MEMORY	User Profile Mode: Out of memory
1072	PI_CNTR_PROFILE_WRONG_CLUSTER	User Profile Mode: Cluster is not assigned to this axis
1073	PI_CNTR_PROFILE_UNKNOWN_CLUSTER_IDENTIFIER	Unknown cluster identifier
1090	PI_CNTR_TOO_MANY_TCP_CONNECTIONS_OPEN	There are too many open tcpip connections
2000	PI_CNTR_ALREADY_HAS_SERIAL_NUMBER	Controller already has a serial number
4000	PI_CNTR_SECTOR_ERASE_FAILED	Sector erase failed
4001	PI_CNTR_FLASH_PROGRAM_FAILED	Flash program failed
4002	PI_CNTR_FLASH_READ_FAILED	Flash read failed
4003	PI_CNTR_HW_MATCHCODE_ERROR	HW match code missing/invalid
4004	PI_CNTR_FW_MATCHCODE_ERROR	FW match code missing/invalid
4005	PI_CNTR_HW_VERSION_ERROR	HW version missing/invalid
4006	PI_CNTR_FW_VERSION_ERROR	FW version missing/invalid
4007	PI_CNTR_FW_UPDATE_ERROR	FW update failed
4008	PI_CNTR_FW_CRC_PAR_ERROR	FW Parameter CRC wrong
4009	PI_CNTR_FW_CRC_FW_ERROR	FW CRC wrong
5000	PI_CNTR_INVALID_PCC_SCAN_DATA	PicoCompensation scan data is not valid
5001	PI_CNTR_PCC_SCAN_RUNNING	PicoCompensation is running, some actions can not be executed during scanning/recording
5002	PI_CNTR_INVALID_PCC_AXIS	Given axis can not be defined as PPC axis
5003	PI_CNTR_PCC_SCAN_OUT_OF_RANGE	Defined scan area is larger than the travel range
5004	PI_CNTR_PCC_TYPE_NOT_EXISTING	Given PicoCompensation type is not defined
5005	PI_CNTR_PCC_PAM_ERROR	PicoCompensation parameter error

5006	PI_CNTR_PCC_TABLE_ARRAY_TOO_LARGE	PicoCompensation table is larger than maximum table length
5100	PI_CNTR_NEXLINE_ERROR	Common error in NEXLINE® firmware module
5101	PI_CNTR_CHANNEL_ALREADY_USED	Output channel for NEXLINE® can not be redefined for other usage
5102	PI_CNTR_NEXLINE_TABLE_TOO_SMALL	Memory for NEXLINE® signals is too small
5103	PI_CNTR_RNP_WITH_SERVO_ON	RNP can not be executed if axis is in closed loop
5104	PI_CNTR_RNP_NEEDED	Relax procedure (RNP) needed
5200	PI_CNTR_AXIS_NOT_CONFIGURED	Axis must be configured for this action
5300	PI_CNTR_FREQU_ANALYSIS_FAILED	Frequency analysis failed
5301	PI_CNTR_FREQU_ANALYSIS_RUNNING	Another frequency analysis is running
6000	PI_CNTR_SENSOR_ABS_INVALID_VALUE	Invalid preset value of absolute sensor
6001	PI_CNTR_SENSOR_ABS_WRITE_ERROR	Error while writing to sensor
6002	PI_CNTR_SENSOR_ABS_READ_ERROR	Error while reading from sensor
6003	PI_CNTR_SENSOR_ABS_CRC_ERROR	Checksum error of absolute sensor
6004	PI_CNTR_SENSOR_ABS_ERROR	General error of absolute sensor
6005	PI_CNTR_SENSOR_ABS_OVERFLOW	Overflow of absolute sensor position

2.4.2 Interface errors

0	COM_NO_ERROR	No error occurred during function call
-1	COM_ERROR	Error during com operation (could not be specified)
-2	SEND_ERROR	Error while sending data
-3	REC_ERROR	Error while receiving data
-4	NOT_CONNECTED_ERROR	Not connected (no port with given ID open)
-5	COM_BUFFER_OVERFLOW	Buffer overflow
-6	CONNECTION_FAILED	Error while opening port
-7	COM_TIMEOUT	Timeout error
-8	COM_MULTILINE_RESPONSE	There are more lines waiting in buffer
-9	COM_INVALID_ID	There is no interface or DLL handle with the given ID
-10	COM_NOTIFY_EVENT_ERROR	Event/message for notification could not be opened
-11	COM_NOT_IMPLEMENTED	Function not supported by this interface type

-12	COM_ECHO_ERROR	Error while sending "echoed" data
-13	COM_GPIB_EDVR	IEEE488: System error
-14	COM_GPIB_ECIC	IEEE488: Function requires GPIB board to be CIC
-15	COM_GPIB_ENOL	IEEE488: Write function detected no listeners
-16	COM_GPIB_EADR	IEEE488: Interface board not addressed correctly
-17	COM_GPIB_EARG	IEEE488: Invalid argument to function call
-18	COM_GPIB_ESAC	IEEE488: Function requires GPIB board to be SAC
-19	COM_GPIB_EABO	IEEE488: I/O operation aborted
-20	COM_GPIB_ENEB	IEEE488: Interface board not found
-21	COM_GPIB_EDMA	IEEE488: Error performing DMA
-22	COM_GPIB_EOIP	IEEE488: I/O operation started before previous operation completed
-23	COM_GPIB_ECAP	IEEE488: No capability for intended operation
-24	COM_GPIB_EFSO	IEEE488: File system operation error
-25	COM_GPIB_EBUS	IEEE488: Command error during device call
-26	COM_GPIB_ESTB	IEEE488: Serial poll-status byte lost
-27	COM_GPIB_ESRQ	IEEE488: SRQ remains asserted
-28	COM_GPIB_ETAB	IEEE488: Return buffer full
-29	COM_GPIB_ELCK	IEEE488: Address or board locked
-30	COM_RS_INVALID_DATA_BITS	RS-232: 5 data bits with 2 stop bits is an invalid combination, as is 6, 7, or 8 data bits with 1.5 stop bits
-31	COM_ERROR_RS_SETTINGS	RS-232: Error configuring the COM port
-32	COM_INTERNAL_RESOURCES_ERROR	Error dealing with internal system resources (events, threads, ...)
-33	COM_DLL_FUNC_ERROR	A DLL or one of the required functions could not be loaded
-34	COM_FTDIUSB_INVALID_HANDLE	FTDIUSB: invalid handle
-35	COM_FTDIUSB_DEVICE_NOT_FOUND	FTDIUSB: device not found
-36	COM_FTDIUSB_DEVICE_NOT_OPENED	FTDIUSB: device not opened
-37	COM_FTDIUSB_IO_ERROR	FTDIUSB: IO error
-38	COM_FTDIUSB_INSUFFICIENT_RESOURCES	FTDIUSB: insufficient resources
-39	COM_FTDIUSB_INVALID_PARAMETER	FTDIUSB: invalid parameter
-40	COM_FTDIUSB_INVALID_BAUD_RATE	FTDIUSB: invalid baud rate
-41	COM_FTDIUSB_DEVICE_NOT_OPENED_FOR_ERASE	FTDIUSB: device not opened for erase
-42	COM_FTDIUSB_DEVICE_NOT_OPENED_FOR_WRITE	FTDIUSB: device not opened for write
-43	COM_FTDIUSB_FAILED_TO_WRITE_DEVICE	FTDIUSB: failed to write device
-44	COM_FTDIUSB_EEPROM_READ_FAILED	FTDIUSB: EEPROM read failed
-45	COM_FTDIUSB_EEPROM_WRITE_FAILED	FTDIUSB: EEPROM write failed

-46	COM_FTDIUSB_EEPROM_ERASE_FAILED	FTDIUSB: EEPROM erase failed
-47	COM_FTDIUSB_EEPROM_NOT_PRESENT	FTDIUSB: EEPROM not present
-48	COM_FTDIUSB_EEPROM_NOT_PROGRAMMED	FTDIUSB: EEPROM not programmed
-49	COM_FTDIUSB_INVALID_ARGS	FTDIUSB: invalid arguments
-50	COM_FTDIUSB_NOT_SUPPORTED	FTDIUSB: not supported
-51	COM_FTDIUSB_OTHER_ERROR	FTDIUSB: other error
-52	COM_PORT_ALREADY_OPEN	Error while opening the COM port: was already open
-53	COM_PORT_CHECKSUM_ERROR	Checksum error in received data from COM port
-54	COM_SOCKET_NOT_READY	Socket not ready, you should call the function again
-55	COM_SOCKET_PORT_IN_USE	Port is used by another socket
-56	COM_SOCKET_NOT_CONNECTED	Socket not connected (or not valid)
-57	COM_SOCKET_TERMINATED	Connection terminated (by peer)
-58	COM_SOCKET_NO_RESPONSE	Can't connect to peer
-59	COM_SOCKET_INTERRUPTED	Operation was interrupted by a nonblocked signal
-60	COM_PCI_INVALID_ID	No device with this ID is present
-61	COM_PCI_ACCESS_DENIED	Driver could not be opened (on Vista: run as administrator!)
-62	COM_SOCKET_HOST_NOT_FOUND	Host not found
-63	COM_DEVICE_CONNECTED	Device already connected

2.4.3 DLL errors

-1001	PI_UNKNOWN_AXIS_IDENTIFIER	Unknown axis identifier
-1002	PI_NR_NAV_OUT_OF_RANGE	Number for NAV out of range--must be in [1,10000]
-1003	PI_INVALID_SGA	Invalid value for SGA--must be one of 1, 10, 100, 1000
-1004	PI_UNEXPECTED_RESPONSE	Controller sent unexpected response
-1005	PI_NO_MANUAL_PAD	No manual control pad installed, calls to SMA and related commands are not allowed
-1006	PI_INVALID_MANUAL_PAD_KNOB	Invalid number for manual control pad knob
-1007	PI_INVALID_MANUAL_PAD_AXIS	Axis not currently controlled by a manual control pad
-1008	PI_CONTROLLER_BUSY	Controller is busy with some lengthy operation (e.g., reference move, fast scan algorithm)
-1009	PI_THREAD_ERROR	Internal error--could not start thread
-1010	PI_IN_MACRO_MODE	Controller is (already) in macro mode--command not valid in macro mode
-1011	PI_NOT_IN_MACRO_MODE	Controller not in macro mode--command not valid unless macro mode active

-1012	PI_MACRO_FILE_ERROR	Could not open file to write or read macro
-1013	PI_NO_MACRO_OR_EMPTY	No macro with given name on controller, or macro is empty
-1014	PI_MACRO_EDITOR_ERROR	Internal error in macro editor
-1015	PI_INVALID_ARGUMENT	One or more arguments given to function is invalid (empty string, index out of range, ...)
-1016	PI_AXIS_ALREADY_EXISTS	Axis identifier is already in use by a connected stage
-1017	PI_INVALID_AXIS_IDENTIFIER	Invalid axis identifier
-1018	PI_COM_ARRAY_ERROR	Could not access array data in COM server
-1019	PI_COM_ARRAY_RANGE_ERROR	Range of array does not fit the number of parameters
-1020	PI_INVALID_SPA_CMD_ID	Invalid parameter ID given to SPA or SPA?
-1021	PI_NR_AVG_OUT_OF_RANGE	Number for AVG out of range--must be >0
-1022	PI_WAV_SAMPLES_OUT_OF_RANGE	Incorrect number of samples given to WAV
-1023	PI_WAV_FAILED	Generation of wave failed
-1024	PI_MOTION_ERROR	Motion error: position error too large, servo is switched off automatically
-1025	PI_RUNNING_MACRO	Controller is (already) running a macro
-1026	PI_PZT_CONFIG_FAILED	Configuration of PZT stage or amplifier failed
-1027	PI_PZT_CONFIG_INVALID_PARAMS	Current settings are not valid for desired configuration
-1028	PI_UNKNOWN_CHANNEL_IDENTIFIER	Unknown channel identifier
-1029	PI_WAVE_PARAM_FILE_ERROR	Error while reading/writing wave generator parameter file
-1030	PI_UNKNOWN_WAVE_SET	Could not find description of wave form. Maybe WG.INI is missing?
-1031	PI_WAVE_EDITOR_FUNC_NOT_LOADED	The WGWaveEditor DLL function was not found at startup
-1032	PI_USER_CANCELLED	The user cancelled a dialog
-1033	PI_C844_ERROR	Error from C-844 Controller
-1034	PI_DLL_NOT_LOADED	DLL necessary to call function not loaded, or function not found in DLL
-1035	PI_PARAMETER_FILE_PROTECTED	The open parameter file is protected and cannot be edited
-1036	PI_NO_PARAMETER_FILE_OPENED	There is no parameter file open
-1037	PI_STAGE_DOES_NOT_EXIST	Selected stage does not exist
-1038	PI_PARAMETER_FILE_ALREADY_OPENED	There is already a parameter file open. Close it before opening a new file
-1039	PI_PARAMETER_FILE_OPEN_ERROR	Could not open parameter file
-1040	PI_INVALID_CONTROLLER_VERSION	The version of the connected controller is invalid

-1041	PI_PARAM_SET_ERROR	Parameter could not be set with SPA--parameter not defined for this controller!
-1042	PI_NUMBER_OF_POSSIBLE_WAVES_EXCEEDED	The maximum number of wave definitions has been exceeded
-1043	PI_NUMBER_OF_POSSIBLE_GENERATORS_EXCEEDED	The maximum number of wave generators has been exceeded
-1044	PI_NO_WAVE_FOR_AXIS_DEFINED	No wave defined for specified axis
-1045	PI_CANT_STOP_OR_START_WAV	Wave output to axis already stopped/started
-1046	PI_REFERENCE_ERROR	Not all axes could be referenced
-1047	PI_REQUIRED_WAVE_NOT_FOUND	Could not find parameter set required by frequency relation
-1048	PI_INVALID_SPP_CMD_ID	Command ID given to SPP or SPP? is not valid
-1049	PI_STAGE_NAME_ISNT_UNIQUE	A stage name given to CST is not unique
-1050	PI_FILE_TRANSFER_BEGIN_MISSING	A uuencoded file transferred did not start with "begin" followed by the proper filename
-1051	PI_FILE_TRANSFER_ERROR_TEMP_FILE	Could not create/read file on host PC
-1052	PI_FILE_TRANSFER_CRC_ERROR	Checksum error when transferring a file to/from the controller
-1053	PI_COULDNT_FIND_PISTAGES_DAT	The PiStages.dat database could not be found. This file is required to connect a stage with the CST command
-1054	PI_NO_WAVE_RUNNING	No wave being output to specified axis
-1055	PI_INVALID_PASSWORD	Invalid password
-1056	PI_OPM_COM_ERROR	Error during communication with OPM (Optical Power Meter), maybe no OPM connected
-1057	PI_WAVE_EDITOR_WRONG_PARAMNUM	WaveEditor: Error during wave creation, incorrect number of parameters
-1058	PI_WAVE_EDITOR_FREQUENCY_OUT_OF_RANGE	WaveEditor: Frequency out of range
-1059	PI_WAVE_EDITOR_WRONG_IP_VALUE	WaveEditor: Error during wave creation, incorrect index for integer parameter
-1060	PI_WAVE_EDITOR_WRONG_DP_VALUE	WaveEditor: Error during wave creation, incorrect index for floating point parameter
-1061	PI_WAVE_EDITOR_WRONG_ITEM_VALUE	WaveEditor: Error during wave creation, could not calculate value
-1062	PI_WAVE_EDITOR_MISSING_GRAPH_COMPONENT	WaveEditor: Graph display component not installed
-1063	PI_EXT_PROFILE_UNALLOWED_CMD	User Profile Mode: Command is not allowed, check for required preparatory commands
-1064	PI_EXT_PROFILE_EXPECTING_MOTION_ERROR	User Profile Mode: First target position in User Profile is too far from current position

-1065	PI_EXT_PROFILE_ACTIVE	Controller is (already) in User Profile Mode
-1066	PI_EXT_PROFILE_INDEX_OUT_OF_RANGE	User Profile Mode: Block or Data Set index out of allowed range
-1067	PI_PROFILE_GENERATOR_NO_PROFILE	ProfileGenerator: No profile has been created yet
-1068	PI_PROFILE_GENERATOR_OUT_OF_LIMITS	ProfileGenerator: Generated profile exceeds limits of one or both axes
-1069	PI_PROFILE_GENERATOR_UNKNOWN_PARAMETER	ProfileGenerator: Unknown parameter ID in Set/Get Parameter command
-1070	PI_PROFILE_GENERATOR_PAR_OUT_OF_RANGE	ProfileGenerator: Parameter out of allowed range
-1071	PI_EXT_PROFILE_OUT_OF_MEMORY	User Profile Mode: Out of memory
-1072	PI_EXT_PROFILE_WRONG_CLUSTER	User Profile Mode: Cluster is not assigned to this axis
-1073	PI_UNKNOWN_CLUSTER_IDENTIFIER	Unknown cluster identifier
-1074	PI_INVALID_DEVICE_DRIVER_VERSION	The installed device driver doesn't match the required version. Please see the documentation to determine the required device driver version.
-1075	PI_INVALID_LIBRARY_VERSION	The library used doesn't match the required version. Please see the documentation to determine the required library version.
-1076	PI_INTERFACE_LOCKED	The interface is currently locked by another function. Please try again later.
-1077	PI_PARAM_DAT_FILE_INVALID_VERSION	Version of parameter DAT file does not match the required version. Current files are available at www.pi.ws.
-1078	PI_CANNOT_WRITE_TO_PARAM_DAT_FILE	Cannot write to parameter DAT file to store user defined stage type.
-1079	PI_CANNOT_CREATE_PARAM_DAT_FILE	Cannot create parameter DAT file to store user defined stage type.
-1080	PI_PARAM_DAT_FILE_INVALID_REVISION	Parameter DAT file does not have correct revision.
-1081	PI_USERSTAGES_DAT_FILE_INVALID_REVISION	User stages DAT file does not have correct revision.
-1082	PI_SOFTWARE_TIMEOUT	Timeout Error. Some lengthy operation did not finish within expected time.
-1083	PI_WRONG_DATA_TYPE	A function argument has an unexpected data type.
-1084	PI_DIFFERENT_ARRAY_SIZES	Length of data arrays is different.
-1085	PI_PARAM_NOT_FOUND_IN_PARAM_DAT_FILE	Parameter value not found in parameter DAT file.
-1086	PI_MACRO_RECORDING_NOT_ALLOWED_IN_THIS_MODE	Macro recording is not allowed in this mode of operation.
-1087	PI_USER_CANCELLED_COMMAND	Command cancelled by user input.
-1088	PI_TOO_FEW_GCS_DATA	Controller sent too few GCS data sets

-1089	PI_TOO_MANY_GCS_DATA	Controller sent too many GCS data sets
-1090	PI_GCS_DATA_READ_ERROR	Communication error while reading GCS data
-1091	PI_WRONG_NUMBER_OF_INPUT_ARGUMENTS	Wrong number of input arguments.
-1092	PI_FAILED_TO_CHANGE_CCL_LEVEL	Change of command level has failed.
-1093	PI_FAILED_TO_SWITCH_OFF_SERVO	Switching off the servo mode has failed.
-1094	PI_FAILED_TO_SET_SINGLE_PARAMETER_WHILE_PERFORMING_CST	A parameter could not be set while performing CST: CST was not performed (parameters remain unchanged).
-1095	PI_ERROR_CONTROLLER_REBOOT	Connection could not be reestablished after reboot.
-1096	PI_ERROR_AT_QHPA	Sending HPA? or receiving the response has failed.
-1097	PI_QHPA_NONCOMPLIANT_WITH_GCS	HPA? response does not comply with GCS2 syntax.
-1098	PI_FAILED_TO_READ_QSPA	Response to SPA? could not be received.
-1099	PI_PAM_FILE_WRONG_VERSION	Version of PAM file cannot be handled (too old or too new)
-1100	PI_PAM_FILE_INVALID_FORMAT	PAM file does not contain required data in PAM-file format
-1101	PI_INCOMPLETE_INFORMATION	Information does not contain all required data
-1102	PI_NO_VALUE_AVAILABLE	No value for parameter available
-1103	PI_NO_PAM_FILE_OPEN	No PAM file is open
-1104	PI_INVALID_VALUE	Invalid value
-1105	PI_UNKNOWN_PARAMETER	Unknown parameter
-1106	PI_RESPONSE_TO_QSEP_FAILED	Response to SEP? could not be received.
-1107	PI_RESPONSE_TO_QSPA_FAILED	Response to SPA? could not be received.
-1108	PI_ERROR_IN_CST_VALIDATION	Error while performing CST: One or more parameters were not set correctly.
-1109	PI_ERROR_PAM_FILE_HAS_DUPLICATE_ENTRY_WITH_DIFFERENT_VALUES	PAM file has duplicate entry with different values.
-1110	PI_ERROR_FILE_NO_SIGNATURE	File has no signature
-1111	PI_ERROR_FILE_INVALID_SIGNATURE	File has invalid signature
-10000	PI_PARAMETER_DB_INVALID_STAGE_TYPE_FORMAT	PI stage database: String containing stage type and description has invalid format.
-10001	PI_PARAMETER_DB_SYSTEM_NOT_AVAILABLE	PI stage database: Database does not contain the selected stage type for the connected controller.
-10002	PI_PARAMETER_DB_FAILED_TO_ESTABLISH_CONNECTION	PI stage database: Establishing the connection has failed.
-10003	PI_PARAMETER_DB_COMMUNICATION_ERROR	PI stage database: Communication was interrupted (e.g. because database was deleted).

-10004	PI_PARAMETER_DB_ERROR_WHILE_QUERYING_PARAMETERS	PI stage database: Querying data failed.
-10005	PI_PARAMETER_DB_SYSTEM_ALREADY_EXISTS	PI stage database: System already exists. Rename stage and try again.
-10006	PI_PARAMETER_DB_QHPA_CONTANS_UNKNOWN_PAM_IDS	PI stage database: Response to HPA? contains unknown parameter IDs.
-10007	PI_PARAMETER_DB_AND_QHPA_ARE_INCONSISTENT	PI stage database: Inconsistency between database and response to HPA?.
-10008	PI_PARAMETER_DB_SYSTEM_COULD_NOT_BE_ADDED	PI stage database: Stage has not been added.
-10009	PI_PARAMETER_DB_SYSTEM_COULD_NOT_BE_REMOVED	PI stage database: Stage has not been removed.
-10010	PI_PARAMETER_DB_CONTROLLER_DB_PARAMETERS_MISMATCH	Controller does not support all stage parameters stored in PI stage database. No parameters were set.
-10011	PI_PARAMETER_DB_DATABASE_IS_OUTDATED	The version of PISTAGES3.DB stage database is out of date. Please update via PIUpdateFinder. No parameters were set.
-10012	PI_PARAMETER_DB_AND_HPA_MISMATCH_STRICT	Mismatch between number of parameters present in stage database and available in controller interface. No parameters were set.
-10013	PI_PARAMETER_DB_AND_HPA_MISMATCH_LOOSE	Mismatch between number of parameters present in stage database and available in controller interface. Some parameters were ignored.
-10014	PI_PARAMETER_DB_FAILED_TO_SET_PARAMETERS_CORRECTLY	One or more parameters could not be set correctly on the controller.
-10015	PI_PARAMETER_DB_MISSING_PARAMETER_DEFINITIONS_IN_DATABASE	One or more parameter definitions are not present in stage database. Please update PISTAGES3.DB via PIUpdateFinder. Missing parameters were ignored.

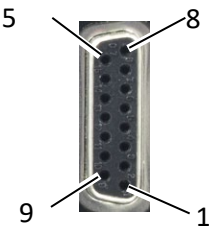
3 Pin Assignments

For the pinout of all connectors not described here see the E727T0005 user manual of the E-727.

3.1 E-727.xxxA, E-727.xxxAx: Analog I/O

Connector type: D-Sub 15 (f)

Pin		Function	Channel identifier*
1		GND	-
	9	-Analog In 1	Input signal channel 4
2		+Analog In 1	
	10	-Analog In 2	Input signal channel 5
3		+Analog In 2	
	11	-Analog In 3	Input signal channel 6
4		+Analog In 3	
	12	-Analog In 4	Input signal channel 7
5		+Analog In 4	
	13	GND	-
6		GND	-
	14	Sensor Monitor 1	-
7		Sensor Monitor 2	-
	15	Sensor Monitor 3	-
8		Analog Out 1	Output signal channel 4



* For further information on channel identifiers, see „Axes, Channels, Functional Elements" in the E727T0005 user manual.

INFORMATION

When using an analog input of the E-727, both the corresponding +Analog In and –Analog In line must be wired.

- Connect a differential signal (+analog, -analog, GND) as follows:

Customer Device	E-727.xxxA, .xxxAx
+Analog Out	+Analog In
-Analog Out	-Analog In
GND	GND

- Connect a single-ended signal as follows (recommended):

Customer Device	E-727.xxxA, .xxxAx
+Analog Out	+Analog In
GND	-Analog In*
GND	GND

* If it is not possible to connect –Analog In to GND on the customer side, -Analog In should be connected to GND on the E-727 side.

- In either case, use a shielded cable.

INFORMATION

To achieve the highest possible resolution and eliminate potential interference that affects the cable used:

➢ Filter the analog output signal of the E-727 in a suitable way, e.g., before you convert it to a digital format. Recommended: low-pass filter with max. 100 kHz cut-off frequency (characteristics: single pole, 6 dB/octave)

Analog inputs (pins 9, 2, 10, 3, 11, 4, 12, 5):

- All inputs are differential.
- Input impedance: 150 kohm to GND
- Max. input voltage (single ended) below which no damage occurs: ±14 V
- Max. input voltage which can be processed by the E-727: ±10 V
- Resolution DAC: 18 bit
- E-727.3CDA, .3CDAx: Via the value of the ***Sensor Range Factor*** parameter (ID 0x02000100), the analog input lines 1 to 4 (accessible as input signal channels 4 to 7) can be configured as follows:
  - 1: Input range ±5 V
  - 2: Input range ±10 V
- E-727.3RDA, .3RDAx, .3SDA, .3SDAx: Via the value of the ***Sensor Range Factor*** parameter (ID 0x02000100), the analog input lines can be configured as follows:
  - Analog input line 1 (accessible as input signal channel 4):
    - 1: No input possible on pins 2 and 9 of **Analog I/O**; input signal channel 4 is used for a piezoresistive or strain gauge sensor (input via pins 6, 24, 25 of the socket for the piezo stage(s)
    - 2: No input possible on pins 2 and 9 of **Analog I/O**; input signal channel 4 is used for a PT1000 temperature sensor (input via pins 1, 20 of the socket for the piezo stage(s))
    - 3: Input range ±5 V
    - 4: Input range ±10 V
  - Analog input lines 2 to 4 (accessible as input signal channels 5 to 7):
    - 1: Input range ±5 V
    - 2: Input range ±10 V

Analog outputs (pins 8, 14, 7, 15):

- All outputs are single-ended.
- All outputs are short circuit protected.
- Max. output current for normal operation, recommended: ±5 mA

- Output voltage range: ±10 V
  The specified output voltage range can be controlled linearly if the recommended maximum output current is observed.

- Sensor Monitor lines 1, 2, 3 (pins 14, 7, 15): The (raw) signals of the sensors 1, 2 and 3 which are fed into the E-727 on the socket for the piezo stage(s) are looped through to these lines. Note the following:

  - The connection of the Sensor Monitor lines to the sensor raw signals is hardware based and cannot be changed by commands and parameters. Therefore, the Sensor Monitor lines are not accessible for PC software (e.g. PIMikroMove) and, in contrast to Analog Out 1, cannot be configured as freely usable outputs.

  - The scaling of the Sensor Monitor lines is hardware based and cannot be changed by commands and parameters. The sensor raw signals are transferred directly analog with a fixed scaling to the Sensor Monitor lines. Since the linearization of the sensor signals in the E-727 only takes place after digitization, the signals at the Sensor Monitor lines are also not linearized.

  - A feedback effect of the Sensor Monitor lines on the sensor raw signals is prevented by an internal buffering of the signals.

- Analog Out 1 (pin 8; accessible as output signal channel 4) can be configured using the value of the ***Select Output Type*** parameter (ID 0x0A000003) as follows:

  - 1: Output has no meaning; output signal channel 4 is used as output voltage for a piezo actuator in the stage, output as Piezo Ch 4 on the socket for piezo stages.

  - 2: Position monitor of an axis. The value of the ***Select Output Index*** parameter (ID 0x0A000004) determines the axis whose position is to be output.

  - 5: Control signal for an external motor driver. The value of the ***Select Output Index*** parameter (ID 0x0A000004) determines the output signal channel whose control value is to be output.

- Analog Out 1 resolution DAC: 20 bit

If a total of four piezo actuators are present in the stage(s), output signal channel 4 must always be configured for use as output voltage (Piezo Ch 4). Note that PI will supply E-727 and the piezo stage(s) as a system with appropriate settings. If you are not sure whether your system can be configured for output of position monitor or control signal, contact our customer service department (service@pi.de).

3.2 Digital I/O Socket

Connector type: MDR14

Function	Pin		Function
3.3 V out, internal resistance: 100 ohm	14	7	not connected
Digital In 1	13	6	not connected
Digital In 2	12	5	Digital Out 1
Digital In 3	11	4	Digital Out 2
not connected	10	3	Digital Out 3
not connected	9	2	Digital In 4 / Reset (active low)
Output of the servo cycles	8	1	GND

Digital inputs (pins 2, 11, 12, 13)

- TTL (low: 0 to 0.8 V, high: 2 to 5 V, max.: 5 V)
- When nothing is connected to a digital input, the signal level is high due to an internal pull-up with 10 kohm resistor.
- The digital input lines can be used for the following purposes:
  - Start data recording: configure the lines IN1 to IN4 with DRT (p. 48)
  - Trigger the wave generator output (lines IN1 and IN2) and stop it (line IN2), see “Wave Generator Started by Trigger Input” in the E727T0005 user manual and the WGO command (p. 128) for details
- Digital In 4 (pin 2) can be configured as reset input using the ***Reboot On DIO Input*** parameter (ID 0x0e001500). Changes of the parameter value become effective immediately. The value of the parameter enables/disables the Reset input as follows:
  - 0 = OFF: Reset input is disabled (default setting)
  - 1 = ON: Reset input is enabled. If the signal level on the Reset input becomes low, the E-727 is rebooted (same behaviour as with the RBT command, p. 82).

Digital outputs (pins 3, 4, 5, 8):

- High level:
  - at -2 mA output current => min. 2.2 V
  - at -0.1 mA output current => min. 3.0 V
- Low level:
  - at +2 mA output current => max. 0.6 V
  - at +0.1 mA output current => max. 0.21 V
- Each digital output has an output impedance of 100 ohm.
- The digital output lines OUT1 to OUT3 can be configured with the CTO (p. 29) and TWS (p. 109) commands for triggering tasks.
- The servo cycle output on pin 8 is not accessible for commands.