Positioner's API Reference Manual

Radioline

September 28, 2020

Contents

1 Introduction

This document describes the command system used for manual and automatic positioner control.

The main logical object of the protocol is logical axis. Logical axis represents one or more devices such as servo amplifier or syncronization module. All commands are divided into axis commands, device commands and system commands. Commands in this manual are sorted by functionality, their description, the specifics of application are noted. Usage examples are given where necessary.

Manual provides general description of control commands. Depending on specific configuration, positioner type or functional purpose of its logical axes some commands are not implemented or are ignored. Please check with the manufacturer for specific product type features.

1.1 Protocol

Positioner is controlled by the SCPI std V1999.0 4.2.1 protocol extended by a notification subscription system (NCPI).

SCPI protocol is used for parameters setup, devices status queries, move commands. There are three types of commands:

- **:SYSTem:** – system commands,
- **:AXIS#:** – logical axes functions,
- **:DEV#:** – hardware modules functions (serve amplifiers and other devices).

NCPI extension allows you to receive information about positioner state change in form of notifications. The only type here is:

- **:NOT:** - notification subscription.

Communication with positioner is carried out via TCP/IP through the rj45 (Ethernet) connector on the controller. Two tcp ports are used:

- SCPI console port 5025 (standard port for SCPI consoles),
- NCPI console port 5026.

When working with positioner via VISA library, socket mode is supported, instrument mode is not.

1.2 Namespacing

COMMAND# – where '#' is a natural number (beginning with 0).

<**argument**> – the argument of a command is in <> brackets.

CMD:[SUBCMD] – the part in [ ] brackets is an unnescesarry argument or part of a command.

There are also code blocks:

```
*IDN?                          % Command query.
> RADIOLINE,XXX,XXX,XXX        % The answer is indicated by symbol '>'.
                               % - oneline commentary.
```

1.3 On distance units

There are several types of distance units used in protocol commands:

- Revolution – an engine revolution.
- Pulse – servo drive feedback discrete. There is a constant ratio of the number of pulses to the number of revolutions.
- Unit – natural unit based on system requirements (typically a centimeter for linear axes and a degree for axes of rotation).

** Pulse number per unit for the axis can be requested by a command "AXIS#:SETTINGS:RATIO?".*

1.4 On acceleration units (acceleration time)

Acceleration is a number of milliseconds required for engine to reach its nominal operating mode. Please note that when the acceleration time is reduced, the acceleration increases.

2 Getting started

SCPI protocol allows you to work in automatic and manual modes, depending on the source of commands. For automatic mode the source of commands is the operating software. For manual mode commands are entered by user via the I/O terminal. These modes have no functional differences and both work through established TCP connection to the SCPI port (5025 by default) of the controller.

Manual mode can be used for connection check, for control commands debugging before using them in automatic mode, also for simple commands execution when operating software is unavailable.

Before you start working with the protocol, you should check the availability of the controller in local network and establish connection to the SCPI port of the positioner. To establish connection in manual mode you can use such utilities as *PuTTY* (mode: *raw*) or analogs.

SCPI protocol packet is an ASCII string, terminated by a symbol of line ending (LF: \n, dec:10, bin:0x0A).

Test product ID query:

```
*IDN?
> RADIOLINE,XXX,XXX,XXX
```

Status query:

```
SYST:STAT?              % Positioner ready state query.
> 0                     % Positioner is ready for work.
```

Axis info query:

```
SYST:AXESTOT?           % Number of axes query.
> 3                     % There are three axes in the system: 0, 1, 2.
AXIS2:UPOSition?        % Indexes are zero-based.
> 0.378123
```

Parameters setup:

```
AXIS2:USPD 2            % Parameters setup and executional commands
                        % are not followed by the answer.
```

3 SCPI commands

3.1 Protocol and error stack

Protocol version and error control.

SYSTem:VERSion?
Get SCPI protocol version.

SYSTem:ERRor[:NEXT]?
Get next error message from stack.

SYSTem:ERRor:COUNt?
Get error stack length.

***CLS**
Clean error stack.

3.2 Other standart functions

***ESE**
Ignored.

***ESE?**
Ignored. (return: 1)

***ESR?**
Ignored. (return: 1)

***OPC**
Ignored.

***OPC?**
Ignored. (return: 1)

***RST**
Ignored.

***SRE**
Ignored.

***SRE?**
Ignored. (return: 1)

***STB?**
Ignored. (return: 1)

***WAI**
Ignored.

3.3 Positioner info

Invariable parameters of the positioner, its devices and axes info.

***IDN?**
Get positioner ID.

SYSTem:DEVSTOTal?
Get number of devices registered.

SYSTem:AXESTOTal?
Get number of axes regist.

3.4 System commands

Commands which affect all the axes and devices of system and system as a whole.

SYSTem:PRESet
Reset all the devices to default state, reset all settings.

SYSTem:IPADDR
Set IPv4 adress of positioner. Commas are used instead of points. Example:

```
SYST:IPADDR 192,168,1,42
```

SYSTem:STOP
Stop all axes.

SYSTem:POWOFF
Disable power of all axes drivers.
** For individual products only. For most products, power management is not required.*

SYSTem:STATus?
Get system ready state.
Answer:
0 – System is ready.
1 – System is not ready.

3.5 Devices commands

Information on the devices.

DEV#:IDN?
Get device ID.

DEV#:PRESET
Reset device parameters to default.

DEV#:STATus?
Get device ready state
Answer:
0 – Device is ready.
1 – Device is not ready.

DEV#:ALM?
Request alarm code of the device in case of its failure.
** Answer depends on device type.*

3.6 Axis commands. Compatibility

Depending on functional purpose of the axis and on servo drive type the axis can support additional functions.

AXIS#:COMPat:SCAN?
Ask if axis has synchronous scanning subsystem.

3.7 Axis commands. Info and state

AXIS#:STATus:IDN?
Get axis ID.

AXIS#:STATus:DEVS?
Get axis devices codes.

AXIS#:STATus:POSition?
Get position in encoder pulses.

AXIS#:STATus:UPOSition?
Get position in units.

AXIS#:SETTINGS:RATIO?
Get number of encoder pulses per unit.

AXIS#:SETTINGS:DEFSPEed?
Get default speed in revolutions per minutes.

AXIS#:SETTINGS:DEFACCel?
Get default acceleration time in milliseconds.

AXIS#:SETTINGS:MAXSPEed?
Get maximum speed in revolutions per minute.

AXIS#:SETTINGS:MINAccel?
Get minimum acceleration time in milliseconds.

AXIS#:STATus[:STATus]?
Get axis ready state.
Answer:
0 — Axis is ready.
1 — Axis is not ready.

AXIS#:STATus:LSWItch?
Get limit switches state.
Answer:
0 – no operation,
1 – left,
2 – right,
10 – both limit switches are triggered (likely an error).
** Only for products with limit switches. In case they are abscent 0 is always returned*

AXIS#:STATus:OPcode?
Get code of the current operation.
Answer:
0 – no operation,
1 – moving is in progress,
-1 – axis is in initialization state.

3.8 Axis commands. System functions

AXIS#:PRESET
Set all the axes to default.

AXIS#:SON
[Remove brake] and power up engine.
** For individual products only. For most products, power management is not required.*

AXIS#:SOFF
Remove power from engine, [set brake].
** For individual products only. For most products, power management is not required.*

3.9 Axis commands. Software limits, zero set

Unlike other settings, zero and move limits settings do not reset when powering off.

AXIS#:SETTINGS:UBACKLIMit <arg>
AXIS#:SETTINGS:UBACKLIMit?
Set/get negative move limit.

AXIS#:SETTINGS:UFORWLIMit <arg>
AXIS#:SETTINGS:UFORWLIMit?
Set/get positive move limit.

AXIS#:SETTINGS:ULIMITS <back>, <forw>
Set both move limits; *back* must be lesser then *forw*.

AXIS#:SETZERo
Define current axis position as zero.

3.10 Axis commands. Movement

AXIS#:SPEeed <rpm>
AXIS#:SPEeed?
Set/get speed in revolutions per minute.

AXIS#:USPEeed <ups>
AXIS#:USPEeed?
Set/get speed in unit per second.

AXIS#:ACCel <ms>
AXIS#:ACCel?
Set/get acceleration/break time interval (for commands of MOV and JOG type).
<ms> – time interval in milliseconds.

AXIS#:MOVe[:RELative] <distance>
Move a distance (relative movement, in encoder pulses).
<distance> – number of pulses. Sign defines movement direction.

AXIS#:MOVe:ABSolute <position>
Move to position (absolute movement, in encoder pulses).
<position> – coordinate in pulses.

AXIS#:UMOVe[:RELative] <distance>
Move a distance (relative movement, in units).
<distance> – distance in units. Sign defines movement direction.

AXIS#:UMOVe:ABSolute <position>
Move to position (absolute movement, in units).
<position> – coordinate in units.

AXIS#:JOG <direction>
Begin movement with constant speed.
<direction:> 1 – forward, -1 – backward.

AXIS#:STOP
Stop axis.

AXIS#:UNSAFE:MOVe <distance>
Move a distance ignoring limits (in encoder pulses).
Command can be used for debugging. Using it in regular mode is not recommended.

AXIS#:UNSAFE:UMOVe <distance>
Move a distance ignoring limits (in units).
Command can be used for debugging. Using it in regular mode is not recommended.

3.11 Axis commands. Continious scanning

Continious scanning system is avalible for axes with synchronization module. Activating continious scanning mode makes synchronization device start sending trigger signals when passing predefined points.

Also when passing predefined points or getting reverse trigger (see NOTRIGMODE) notifications are generated (see NOT:AXIS#:SCANPOINT). To get information on passing the points by client, you need to subscribe to the approptiate notifications.

The formula for scanning step is: STEP = UMOVE / (POINTS - 1)

AXIS#:SCAN:MOVE <distance>
AXIS#:SCAN:MOVE?
AXIS#:SCAN:UMOVE <distance>
AXIS#:SCAN:UMOVE?
Set/get scanning zone.
Distance here is distance, processed by the scanning algorithm in pulses or units. Hereinafter commands with U prefix work with units, commands without it – with pulses.

AXIS#:SCAN:FWRDzone <distance>
AXIS#:SCAN:FWRDzone?
AXIS#:SCAN:UFWRDzone <distance>
AXIS#:SCAN:UFWRDzone?
Set/get distance to the first point (in pulses/units).
Distance here is distance, passed by the algorithm before scanning.

AXIS#:SCAN:BWRDzone <distance>
AXIS#:SCAN:BWRDzone?
AXIS#:SCAN:UBWRDzone <distance>
AXIS#:SCAN:UBWRDzone?
Set/get distance passed after last point (only for SCAN:START mode).

AXIS#:SCAN:COMPSTART
Activate scanning mode.
Activating scanning mode doesn't start movement, it only turns on generation of triggers and notifications on passing points.

AXIS#:SCAN:POINTS <npnts>
AXIS#:SCAN:POINTS?
Set/get scanning points number.
<npnts> – number of bypass point.

AXIS#:SCAN:NOTRIGMODE <en>
Enable/disable waiting for the reverse trigger before the next point is notified.
<en>:
0 – notification on reaching next point is send after getting reverse trigger (by default),
1 – waiting for reverse trigger disabled

AXIS#:MANTRIGmode <en>
Enable/disable manual trigger generation for axis.
Configures the synchronizer to work with the command AXIS#:TRIGGER.
en:
1 – enable manual mode
0 – disable manual mode

AXIS#:TRIGGER
Generate trigger.
To use this command, manual trigger mode must be enabled (see AXIS#:MANTRIGmode).

AXIS#:TRIGRETTIME?
Get reverse trigger return time (in milliseconds).

4 NCPI API Reference

4.1 Introduction

The NCPI system allows you to receive information about events occuring in a device without polling it.

When connected to server, the client must activate nessesary notification types (or "themes") using commands described in this section. Client can subscribe to any number of types simultaniously, but only one time per notification type. When the subscription is active, server generates notification messages with structure "Name:Themes [Answer]". TCP guarantees message delivering.

NCPI has three message types:

- simple notification,
- status change notification, notification with argument,
- smooth parameter notification.

Simple notification

A simple event means a notification structure: SUBSYST:THEME and does not carry any information other then that the event is occured.

```
NOT:SUBSYST:THEME 1         % Subscribe.
NOT:SUBSYST:THEME 0         % Unsubscribe.
                            %
> SUBSYST:THEME             % Notification format.
```

Status change notification, events with argument

Status change notification occures when the corresponding variable is changed and passes the corresponding string, code or value.

```
NOT:SUBSYST:THEME 1         % Subscribe.
NOT:SUBSYST:THEME 0         % Unsubscribe.
                            %
> SUBSYST:THEME <argument>  % Notification format.
```

Smooth parameter change notification

Such notification pass information about continious parameters that are smoothly changing over time (for example, current position). There are two notification strategies:

1. Notification with time discrete (in milliseconds). Discrete specifies the minimum time interval between notifications. It is important to note that the real intervals of time exceed the preset interval, since the event is generated at the moment of receiving the response to the request of the specified parameter by the controlling system. This ensures maximum accuracy for the time the notification received, but does not allow to confidently calculate the time of the next notification. Besides, the minimum interval for the system exists and messages can't be send with time interval shorter (usually this interval equals a few tens of milliseconds).

```
NOT:SUBSYST:THEME TIMERED,1000  % Subscribe with 1 second interval.
NOT:SUBSYST:THEME 0             % Unsubscribe.
%
SUBSYST:THEME <argument>        % Notification format.
```

2. Notification when parameter is changed to a preset value. As in the case of time interval notifications, the real parameter change value can be larger then preset value.

```
NOT:SUBSYST:THEME SMOOTH,5.34   % Subscribe on value delta = 5.34.
NOT:SUBSYST:THEME 0             % Unsubscribe.
                                %
SUBSYST:THEME <argument>        % Notification format.
```

4.2 Example

```
NOT:AXIS2:OPSTAT 1   % Subscribe to axis 2 status change.
                     %
> AXIS2:OPSTAT 1     % Axis 2 begins movement.
> AXIS2:OPSTAT 0     % Axis 2 ends movement
                     %
NOT:AXIS2:OPSTAT 0   % Unsubscribe.
```

4.3 NCPI commands

NOT:SYSTem:STATus
System ready notification.
Type: status change.

System is ready when none of the devices signals any error.
Parameter:
0 – ready,
1 – not ready.

NOT:AXIS#:STATus
Axis ready notification.
Type: status change.

Axis is ready when none of the devices signals any error.
Parameter:
0 – ready,
1 – not ready.

NOT:DEV#:STATus
Device ready notification.
Type: status change.

Parameter:
0 - ready,
1 - not ready

NOT:AXIS#:POSition
Axis position notification.
Type: continious parameter.

NOT:AXIS#:UPOSition
Axis position notification (in units).
Type: continious parameter.

NOT:AXIS#:OPSTATus
Status of operation notification.
Type: status change.

Parameter:
0 - No operation,
1 - Task for movement,
2 - Scan Mode.

NOT:AXIS#:OPSTOPtype
Notification on interruption of last operation (carries the interuption type).
Type: status change.

Parameter:
0 - operation started,
1 - normal completion of the operation,
2 - operation completed with stop command,
3 - emergency termination of the operation.

NOT:AXIS#:SCAN:POINT
Notification on reaching a new point in scan mode.
Type: notification with argument.

Returns the point number in the body of the notification.

NOT:AXIS#:SCAN:TRIGERROR
Notification on synchronisation system error during waiting for reverse trigger (The new trigger is generated before the previous one is returned).
Type: event.

NOT:AXIS#:SCAN:LSWItch
Limit switches status change notification.
Type: status change.

Parameter:
0 - no operation,
1 - left,
2 - right,
10 - both limit switches are triggered (likely an error).

5 Examples

5.1 Example of script for product (python3)

```
#!/usr/bin/env python3

import sys
import time
import socket
import threading

HOST = '192.168.1.224' # ip address
SCPI_PORT = 5025
NCPI_PORT = 5026

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
n = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, SCPI_PORT))
n.connect((HOST, NCPI_PORT))

# request identifier
s.send("AXIS0:STAT:IDN?\r\n".encode("utf-8"))
answer = s.recv(1024).decode("utf-8")
print("axis0 idn is {}".format(answer))

# set speed : 1 unit per second
s.send("AXIS0:USPE 1\r\n".encode("utf-8"))

# set acceleration : 2 seconds for acceleration to nominal speed
s.send("AXIS0:ACCEL 2000\r\n".encode("utf-8"))

# absolute move to coordinate 4u
print("move to 4u")
s.send("AXIS0:UMOV:ABS 4\r\n".encode("utf-8"))

time.sleep(5)

# absolute move to coordinate 0u
print("move to 0u")
s.send("AXIS0:UMOV:ABS 0\r\n".encode("utf-8"))
```

5.2 Example of script for product using notification system (python3)

```
#!/usr/bin/env python3

import sys
import time
import socket
import threading

HOST = '192.168.1.224' # ip address
SCPI_PORT = 5025
NCPI_PORT = 5026

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
n = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, SCPI_PORT))
n.connect((HOST, NCPI_PORT))

s.send("AXIS0:STAT:IDN?\r\n".encode("utf-8"))
answer = s.recv(1024).decode("utf-8")
print("axis0 idn is {}".format(answer))

print("enable notification")
n.send("NOT:AXIS0:UPOS SMOOTH, 0.1\r\n".encode("utf-8"))
n.send("NOT:AXIS0:OPSTAT 1\r\n".encode("utf-8"))

cancel = False
def foo():
        n.settimeout(0.5)
        while cancel is False:
                try:
                        message = n.recv(1024).decode("utf-8")
                except:
                        pass
                sys.stdout.write("notification: {}".format(message))

thr = threading.Thread(target=foo)
thr.start()

print("move to 4u")
s.send("AXIS0:UMOV:ABS 4\r\n".encode("utf-8"))

time.sleep(5)

print("move to 0u")
s.send("AXIS0:UMOV:ABS 0\r\n".encode("utf-8"))

time.sleep(5)
cancel = True
```

6 Typical problems

Connection is not established
Make sure that the control unit is avalible in local network. Check the hardware connection. Ping the positioner. Make sure that the firewall on your PC does not block the connection. Restart the control unit if needed.

Product does not react on control commands
Check the format and text of command. Make sure that the command is terminated with line end symbol LF.

Product answers the commands, but does not move
Check the SCPI error stack. Check system and axes status using commands:

```
SYST:STAT?
AXIS\#:STAT?
```

Please contact the manufacturer in case of errors.

Axis moves in the same direction regardless of the given direction of movement.
Check UNSAFE movement commands. Make sure the axis limits are set correctly and that the current position is not down. Reconfigure zero and limits if needed.

No notifications comes in continious scanning mode
Make sure the axis supports continious scanning mode Check that the trigger circuits are installed correctly. Check that scan setting are set correctly.