

NAME

E712 – Physik Instrumente E-712 Piezo Controller

DESCRIPTION

The Physik Instrument E-712 is a modular digital piezo controller available in 3- and 6-channel configurations. `spec` supports the controller's serial and Ethernet interfaces.

Use the **Device** screen of `spec`'s hardware configuration editor, the `edconf` program (normally run from the `config` macro), to select the controller type and to assign the interface parameters.

Motor and Counter Device Configuration (Not CAMAC)

| MOTORS | DEVICE | ADDR | <>MODE | NUM | <>TYPE |
|--------|------------|------|--------|-----|------------------------------|
| YES | /dev/ttyS1 | <> | 115200 | 3 | PI E-712 Controller (Serial) |
| YES | E712host | | | 3 | PI E-712 Controller (Socket) |

For the socket configuration, the `DEVICE` column contains an IP address or resolvable host name. A optional port number can be specified as in `192.168.1.123:50000`. If not specified, the default port of 50000 is used. (That is currently the only port supported by the controller.)

The piezo stages supported by the E-712 can be operated in servo or open-loop mode. When `spec` makes a connection to the controller, `spec` queries the current setting of each stage. If in open-loop mode, `spec` uses the `SVA` command to move the motor and the `VOL?` command to read the position. In servo mode, the commands `MOV` and `POS?` are used. The mode can be changed using the `"servo_mode"` option to `motor_par()`. The `"servo_mode"` parameter can be also added to the hardware `config` file as a nonstandard optional parameter, as described below.

When in open-loop mode, there is no status available from the controller as to whether the move has completed or if the stage is on target, so moves are considered complete virtually immediately after sending the move command. Configure the settle time and dead band parameters, described below, if delays are needed.

On the **Motor** screen of the configuration editor, choose the controller type as `PI_E712`. An appropriate value for the steps-per-degree parameter is `1e6`. The slew rate, base rate and acceleration time are not used by `spec` for this controller. However, some of the standard optional motor parameters, reached by typing a lower case `m` from the main motor configuration screen, are recognized and may be useful.

To configure a settling time at the end of each move, set the optional parameters DC dead band (`"dc_dead_band"`) and DC settle time (`"dc_settle_time"`). If those parameters are configured, after the controller reports the move is complete, `spec` will continue to wait an additional time given by the settle time before reporting the move complete. In addition, `spec` will also wait for the difference between the target position and the actual position to be within the dead band. If the motor doesn't settle within five seconds, `spec` reports an error. Note, the dead band and settling time parameters set here are not associated with the controller's internal on-target tolerance parameter (`0x07000900`) and settling time parameter (`0x07000901`).

Additionally, the read-back slop parameter (`"slop"`) and/or the hardware read-mode parameter (`"read_mode"`) can be used as needed to suppress position discrepancy messages. If the slop parameter is set to some number of steps, `spec` will not report position discrepancies smaller than that value and will instead silently accept the reported hardware value as the position. The read-mode parameter controls how often the position is read from the hardware and can also tell `spec` to always accept the reported hardware value, no matter how large. `spec` normally assumes a motor is at its last position. If the `PREMOVE` read-mode option is

set, `spec` will always read the hardware before moving the motor. If the `ALWAYS` option is set, `spec` will read the hardware whenever the `get_angles` command is run from user level. If the `NO_QUERY` option is set, `spec` will always assume the hardware is correct and not query the user as to how to resolve the discrepancy if the returned hardware position differs from the current software position. If the `NO_QUERY` option is set, the `"slop"` parameter is not used.

PARAMETER CLASSES

The E-712 has hundreds of parameters which are described in more detail in the Physik Instrumente user manual for the controller. Physik Instrumente assigns the parameters to nine classes, each associated with a particular function, as follows:

- Servo* – logical axis
- Input* – input channel
- Output* – output channel
- System* – whole system
- DDL* – Dynamic Digital Linearization
- StdIF* – standard interfaces
- WaveGen* – wave generator
- Recorder* – data recorder
- FW_Update* – firmware

For any given class, there are values for a varying number of channel or index identifiers. For example, there are as many *Servo* indices as there are logical axes. The number of *Input*, *Output* and *FW_Update* indices depends on the hardware model. Some *System* parameters have only one index while the number of indices for other *System* parameters depends on the hardware model. There is only one index for *DDL*, *StdIF*, *WaveGen* and *Recorder* parameters for all hardware models.

Each parameter is associated with a unique hexadecimal number code, an axis (or index) identifier number from 1 to the maximum number of indices, a class as listed above, an attribute that indicates what privileges are needed to modify the parameter and a description of the parameter. `SPEC` only allows modification to unprotected parameters. For other parameters, the controller requires a password to make changes.

PARAMETER DISPLAY

Various parameter attributes can be displayed with options to the `motor_par()` function, as described below. For example the following displays non-zero parameters associated with axis 1 of the controller:

```
SPEC.1> p motor_par(z, "dump_servo")
Servo parameters:
      Range Limit max [1] (0x07000001) : 50
      Servo Loop Slew-Rate [1] (0x07000200) : 5000
      Open Loop Slew-Rate [1] (0x07000201) : 1e+06
      Servo-loop P-Term [1] (0x07000300) : 0.031
      Servo-loop I-Term [1] (0x07000301) : 0.000363552
      Position from Sensor 1 [1] (0x07000500) : 0.5
      Position from Sensor 2 [1] (0x07000501) : 0.5
      Axis Name [1] (0x07000600) : Z
      Axis Unit [1] (0x07000601) : um
      ...
```

The following options are available. Note, the argument is case insensitive – upper and/or lower case letters can be used interchangeably. All commands are directed to the controller associated with motor `mne`. Only commands in the *Servo* class are automatically associated

with the particular motor axis *mne*. Some forms of the commands suppress displaying parameters that have a value of zero for ease of viewing.

`motor_par(mne, "dump")` – Display all parameters from all classes.

`motor_par(mne, "dump_all")` – Display all parameters from all classes, but suppress printing parameters with values of zero.

`motor_par(mne, "dump_Servo")` – Display non-zero parameters associated with the motor *mne*.

`motor_par(mne, "dump_Servo", n)` – Display all parameters associated with the axis *n*.

`motor_par(mne, "dump_Input")` – Display non-zero parameters associated with all input channels.

`motor_par(mne, "dump_Input", n)` – Display all parameters associated with input channel *n*.

`motor_par(mne, "dump_Output")` – Display non-zero parameters associated with all output channels.

`motor_par(mne, "dump_Output", n)` – Display all parameters associated with output channel *n*.

`motor_par(mne, "dump_System")` – Display non-zero system parameters.

`motor_par(mne, "dump_System", n)` – Display all system parameters with index *n*.

`motor_par(mne, "dump_FW_Update")` – Display all non-zero firmware-related parameters.

`motor_par(mne, "dump_FW_Update", n)` – Display all firmware-related parameters with index *n*.

`motor_par(mne, "dump_StdIF")` – Display all non-zero interface-related parameters.

`motor_par(mne, "dump_DDL")` – Display all non-zero dynamic digital linearization parameters.

`motor_par(mne, "dump_WaveGen")` – Display all non-zero wave generator parameters.

`motor_par(mne, "dump_Recorder")` – Display all non-zero recorder parameters.

INDIVIDUAL PARAMETERS

Individual parameters can be displayed or set using the hexadecimal parameter code. A particular parameter code may be associated with more than one parameter, as differentiated by index number. For *Servo* class parameters, the index number can come from the motor number. For all classes of parameters, the index number can be specified using a string in the form "*hexpar:index*". For example, to print the stage type (a *System* class parameter) associated with axis 2, use:

```
p motor_par(mne, "0x0f000100:2")
```

For *Servo* class parameters, if the index number is missing, it will be derived from the motor number. If the index count for the parameter is one, no index need be given. Otherwise, an error message is displayed. If an index is not needed, the parameter number does not need to be passed as a string. For example, to print the axis name, the following works:

```
p motor_par(mne, 0x07000600)
```

Parameter values are set by supplying the value in the third argument:

```
motor_par(mne, 0x7000200, 5000)
```

The abovet sets the servo loop slew-rate for motor *mne* to 5000.

NONSTANDARD OPTIONAL PARAMETERS

Nonstandard optional parameters are hardware parameters that are not part of spec's device-independent hardware support. The parameters can be recognized by spec's device-dependent support, as documented for the particular controller.

From the spec hardware configuration editor, nonstandard optional parameters are created by typing the `p` command when a device row or motor column contains the highlighted cell. For the E-712, parameters can be associated either with the controller unit on the **Devices** screen or with individual motors on the **Motor** screen.

Nonstandard optional parameters recognized by the E-712 support are the parameter "servo_mode" and all the hexadecimal parameter codes recognized by the controller. The "servo_mode" parameter and hexadecimal parameters in the *Servo* class are associated with an individual motor axis. The rest of the hexadecimal parameters are associated with the controller entry.

The configuration screen for nonstandard optional parameters for a motor channel looks like this (reached by typing the `p` command from the main **Motor** screen):

```
Custom Parameters for Motor "Xrot" (xrot)          2/2 configured

NAME                VALUE
servo_mode          1
0x7000300           .031

Type p when done          Type ? and H for help, ^C to quit
```

For the controller entry on the **Devices** screen, the nonstandard optional parameter screen looks like this:

```
Custom Parameters for "PI E-712 Piezo Controller" 6/6 configured

NAME                VALUE
0x02000100:1        1
0x02000100:2        1
0x02000101:1        1
0x02000101:2        1
0x02000102:1        16
0x02000102:2        16
```

The nonstandard optional parameters can also be accessed via the `motor_par()` command, although if the parameters are given values in the *config* file, the *config* file values will be reprogrammed when the hardware is reinitialized on startup or with the `reconfig` command.

PARAMETER ACCESS VIA `motor_par()`

The nonstandard optional parameters may be accessed via `motor_par()`, whether or not the parameters have been entered into the *config* file, as described above. When using `motor_par()` to access parameters not associated with a particular axis, such as all of the hexadecimal parameters that are not in the *Servo* class, the *mne* argument simply identifies the controller to which the parameter belongs.

Note, values for nonstandard optional parameters associated with the controller device in the *config* file cannot be modified using `motor_par()`. Their values can only be changed through the configuration editor.

`motor_par(mne, "servo_mode")` - Returns one or zero, indicating whether servo mode is on or off for the specified axis.

`motor_par(mne, "servo_mode", 1|0)` - Sets or clears servo motor for motor *mne*.

`motor_par(mne, hexpar)` - Returns current value for *hexpar*. If associated with the *Servo* class, returns for motor *mne*. Otherwise, if only one "item" associated with the parameter, returns that item. Otherwise, returns an error.

`motor_par(mne, "hexpar:index")` - Returns current value for *hexpar* associated with item *index*. Item numbers start at one.

`motor_par(mne, hexpar, value)` - Set value for *hexpar*. If associated with the *Servo* class, sets value for motor *mne*. Otherwise, if only one "item" associated with the parameter, sets that item. Otherwise, returns an error.

`motor_par(mne, "hexpar:index", value)` - Set value for *hexpar* associated with item *index*. Item numbers start at one.

COMMAND PASS THROUGH

Command pass through is available using the following functions. Command pass through should be used with caution to avoid interfering with the built-in programming commands *spec* sends to the controllers.

`motor_par(mne, "send", cmd)` - Sends the string *cmd* to the channel associated with *mne*.

`motor_par(mne, "read", cmd)` - Sends the string *cmd* to the channel associated with *mne*, as above, and returns a string containing the response.

`motor_par(mne, "usend", cmd)` - Sends the *cmd* to the controller associated with *mne*. Unlike the "send" option above, no channel address is added to the string.

`motor_par(mne, "uread", cmd)` - Sends the string *cmd* to the controller associated with *mne*, as above, and returns a string containing the response. Again, unlike the "read" option above, no channel address is added to the string.