newfocus8742 Documentation

Release 0.1

Robert Jördens

Jul 04, 2019

CONTENTS

Contents:

CHAPTER

ONE

NEWFOCUS8742.PROTOCOL MODULE

class newfocus8742.protocol.**NewFocus8742Protocol**

New Focus/Newport 8742 Driver.

Four Channel Picomotor Controller and Driver Module, Open-Loop, 4 Channel. https://www.newport.com/p/8742

abort(*xx=None*, **nn*)

Abort motion.

This command is used to instantaneously stop any motion that is in progress. Motion is stopped abruptly. For stop with deceleration see ST command which uses programmable acceleration/deceleration setting.

async ask(*cmd*, *xx=None*, **nn*)

Execute a command and return a response.

The command needs to include the final question mark.

See Also:

***fmt_cmd()*: for the formatting and additional** parameters.

check_motor(*xx=None*, **nn*)

Motor check.

This command scans for motors connected to the controller, and sets the motor type based on its findings. If the piezo motor is found to be type 'Tiny' then velocity (VA) setting is automatically reduced to 1750 if previously set above 1750. To accomplish this task, the controller commands each axis to make a one-step move in the negative direction followed by a similar step in the positive direction. This process is repeated for all the four axes starting with the first one. If this command is issued when an axis is moving, the controller will generate "MOTION IN PROGRESS" error message.

do(*cmd*, *xx=None*, **nn*)

Format and send a command to the device

See Also:

***fmt_cmd()*: for the formatting and additional** parameters.

async done(*xx=None*, **nn*)

Motion done query.

This command is used to query the motion status for an axis.

async error_code(*xx=None*, **nn*)

Get Error code.

This command is used to read the error code. The error code is one numerical value up to three(3) digits long. (see Appendix for complete listing) In general, non-axis specific errors numbers range from 0-99. Axis-1 specific errors range from 100-199, Axis-2 errors range from 200-299 and so on. Note: Errors are

maintained in a FIFO buffer ten(10) elements deep. When an error is read using TB or TE, the controller returns the last error that occurred and the error buffer is cleared by one(1) element. This means that an error can be read only once, with either command.

async error_message(*xx=None*, **nn*)
Query error code and the associated message.

The error code is one numerical value up to three(3) digits long. (see Appendix for complete listing) In general, non-axis specific errors numbers range from 0- 99. Axis-1 specific errors range from 100-199, Axis-2 errors range from 200-299 and so on. The message is a description of the error associated with it. All arguments are separated by commas. Note: Errors are maintained in a FIFO buffer ten(10) elements deep. When an error is read using TB or TE, the controller returns the last error that occurred and the error buffer is cleared by one(1) element. This means that an error can be read only once, with either command.

fmt_cmd(*cmd*, *xx=None*, **nn*)
Format a command.

Args: cmd (str): few-letter command xx (int, optional for some commands): Motor channel nn (multiple int, optional): additional parameters

async get_acceleration(*xx=None*, **nn*)
Get acceleration.

This command is used to query the acceleration value for an axis.

async get_home(*xx=None*, **nn*)
Get home position.

This command is used to query the home position value for an axis.

async get_position(*xx=None*, **nn*)
Get target position.

This command is used to query the target position of an axis.

async get_relative(*xx=None*, **nn*)
This command is used to query the target position of an axis.

async get_type(*xx=None*, **nn*)
Get motor type.

This command is used to query the motor type of an axis. It is important to note that the QM? command simply reports the present motor type setting in memory. It does not perform a check to determine whether the setting is still valid or corresponds with the motor connected at that instant. If motors have been removed and reconnected to different controller channels or if this is the first time, connecting this system then issuing the Motor Check (MC) command is recommended. This will ensure an accurate QM? command response.

async get_velocity(*xx=None*, **nn*)
Get Velocity.

This command is used to query the velocity value for an axis.

async identify(*xx=None*, **nn*)
Get product identification string.

This query will cause the instrument to return a unique identification string. This similar to the Version (VE) command but provides more information. In response to this command the controller replies with company name, product model name, firmware version number, firmware build date, and controller serial number. No two controllers share the same model name and serial numbers, therefore this information can be used to uniquely identify a specific controller.

move (*xx=None*, **nn*)
Indefinite move.

This command is used to move an axis indefinitely. If this command is issued when an axis' motion is in progress, the controller will ignore this command and generate "MOTION IN PROGRESS" error message. Issue a Stop (ST) or Abort (AB) motion command to terminate motion initiated by MV

async position (*xx=None*, **nn*)
Get actual position.

This command is used to query the actual position of an axis. The actual position represents the internal number of steps made by the controller relative to its position when controller was powered ON or a system reset occurred or Home (DH) command was received. Note that the real or physical position of the actuator/motor may differ as a function of mechanical precision and inherent open-loop positioning inaccuracies.

recall (*xx=None*, **nn*)
Recall settings.

This command restores the controller working parameters from values saved in its nonvolatile memory. It is useful when, for example, the user has been exploring and changing parameters (e.g., velocity) but then chooses to reload from previously stored, qualified settings. Note that "*RCL 0" command just restores the working parameters to factory default settings. It does not change the settings saved in EEPROM.

reset (*xx=None*, **nn*)
Reset.

This command performs a "soft" reset or reboot of the controller CPU. Upon restart the controller reloads parameters (e.g., velocity and acceleration) last saved in non-volatile memory. Note that upon executing this command, USB and Ethernet communication will be interrupted for a few seconds while the controller re-initializes. Ethernet communication may be significantly delayed (~30 seconds) in reconnecting depending on connection mode (Peer-to-peer, static or dynamic IP mode) as the PC and controller are negotiating TCP/IP communication.

set_acceleration (*xx=None*, **nn*)
Set acceleration.

This command is used to set the acceleration value for an axis. The acceleration setting specified will not have any effect on a move that is already in progress. If this command is issued when an axis' motion is in progress, the controller will accept the new value but it will use it for subsequent moves only.

set_home (*xx=None*, **nn*)
Set home position.

This command is used to define the "home" position for an axis. The home position is set to 0 if this command is issued without "nn" value. Upon receipt of this command, the controller will set the present position to the specified home position. The move to absolute position command (PA) uses the "home" position as reference point for moves.

set_position (*xx=None*, **nn*)
Target position move command.

This command is used to move an axis to a desired target (absolute) position relative to the home position defined by DH command. Note that DH is automatically set to 0 after system reset or a power cycle. If this command is issued when an axis' motion is in progress, the controller will ignore this command and generate "MOTION IN PROGRESS" error message. The direction of motion and number of steps needed to complete the motion will depend on where the motor count is presently at before the command is issued. Issue a Stop (ST) or Abort (AB) motion command to terminate motion initiated by PA

set_relative (*xx=None*, **nn*)
Relative move.

This command is used to move an axis by a desired relative distance. If this command is issued when an axis' motion is in progress, the controller will ignore this command and generate "MOTION IN PROGRESS" error message. Issue a Stop (ST) or Abort (AB) motion command to terminate motion initiated by PR

set_type (*xx=None*, **nn*)
Motor type set command.

This command is used to manually set the motor type of an axis. Send the Motors Check (MC) command to have the controller determine what motors (if any) are connected. Note that for motor type 'Tiny', velocity should not exceed 1750 step/sec. To save the setting to non-volatile memory, issue the Save (SM) command. Note that the controller may change this setting if auto motor detection is enabled by setting bit number 0 in the configuration register to 0 (default) wit ZZ command. When auto motor detection is enabled the controller checks motor presence and type automatically during all moves and updates QM status accordingly.

set_velocity (*xx=None*, **nn*)
Set Velocity.

This command is used to set the velocity value for an axis. The velocity setting specified will not have any effect on a move that is already in progress. If this command is issued when an axis' motion is in progress, the controller will accept the new value but it will use it for subsequent moves only. The maximum velocity for a 'Standard' Picomotor is 2000 steps/sec, while the same for a 'Tiny' Picomotor is 1750 steps/sec

stop (*xx=None*, **nn*)
Stop motion.

This command is used to stop the motion of an axis. The controller uses acceleration specified using AC command to stop motion. If no axis number is specified, the controller stops the axis that is currently moving. Use Abort (AB) command to abruptly stop motion without deceleration.

CHAPTER

TWO

NEWFOCUS8742.TCP MODULE

class newfocus8742.tcp.**NewFocus8742TCP**(*reader*, *writer*)

classmethod connect(*host*, *port=23*, ***kwargs*)

Connect to a Newfocus/Newport 8742 controller over Ethernet/TCP.

Args: host (str): Hostname or IP address of the target device.

Returns: NewFocus8742: Driver instance.

CHAPTER

THREE

NEWFOCUS8742.USB MODULE

class newfocus8742.usb.**NewFocus8742USB**(*dev*)

classmethod connect(*idVendor=4173*, *idProduct=16384*, ***kwargs*)

Connect to a Newfocus/Newport 8742 controller over USB.

Args: **kwargs: passed to *usb.core.find*

Returns: NewFocus8742: Driver instance.

flush()

Drain the input buffer from read data.

CHAPTER

FOUR

NEWFOCUS8742.SIM MODULE

class newfocus8742.sim.**NewFocus8742Sim**

classmethod connect(**args*, ***kwargs*)

Connect to a Newfocus/Newport 8742 controller simulation.

Args: any: ignored

Returns: NewFocus8742: Driver instance.

CHAPTER

FIVE

NEWFOCUS8742.ACQTL_NEWFOCUS8742 MODULE

```
usage: aqctl_newfocus8742 [-h] [--tcp TCP] [--simulation]
```

5.1 Named Arguments

--tcp use TCP device, else use first USB device

--simulation simulation device

Default: False

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

n

INDEX

S