NANOMOTION
A Johnson Electric Company

XCD Software Version 1.5.0.7

User Manual

XCD0458002-00 Rev. 02 **March 2014**

Nanomotion Ltd. POB 623, Yokneam 20692, Israel
Tel: 972-73-2498000 Fax: 972-73-2498099
Web Site: www.nanomotion.com
E-mail: nano@nanomotion.com

Copyright Notice

Limited Warranty

Nanomotion Ltd. (hereinafter NM) warrants the product (other than software) manufactured by it to be free from defects in material and workmanship for a period of time of one year (except those parts normally considered as consumable/expendable components such as motor conditioning brushes). The warranty commences thirty (30) days from the date of shipment.

NM warrants those parts replaced under warranty for a period equal to the remaining warranty coverage of the original part.

NM's sole and exclusive obligation under this warranty provision shall be to repair, or at its sole option exchange defective products or the relevant part or component, but only if: (i) the Purchaser reports the defect to NM in writing and provides a description of the defective product and complete information about the manner of its discovery within ten (10) days of its discovery; (ii) NM has the opportunity to investigate the reported defect and to determine that the defect arises from faulty material, parts or workmanship; and (iii) the Purchaser returns the affected product to a location designated by NM. These provisions constitute the exclusive remedy of the Purchaser for product defects or any other claim of liability in connection with the purchase or use of NM products.

This warranty policy applies only to NM products purchased directly from NM or from an authorized NM distributor or representative.

This warranty shall not apply to (i) products repaired or altered by anyone other than those authorized by NM; (ii) products subjected to negligence, accidents or damage by circumstances beyond NM control; (iii) product subjected to improper operation or maintenance (i.e. operation not in accordance with NM Installation Manuals and/or instructions) or for use other than the original purpose for which the product was designed to be used.

NM shall not in any event have obligations or liabilities to the Purchaser or any other party for loss of profits, loss of use or incidental, increased cost of operation or delays in operation, special or consequential damages, whether based on contract, tort (including negligence), strict liability, or any other theory or form of action, even if NM has been advised of the possibility thereof, arising out of or in connection with the manufacture, sale, delivery, use, repair or performance of the NM products. Without limiting the generality of the preceding sentence, NM shall not be liable to the Purchaser for personal injury or property damages.

Patents

Nanomotion products are covered under one or more patents registered or applied for. For a list of applicable patents refer to the appendix of this document or to the Nanomotion website.

Customer Service

Website: www.nanomotion.com

Contact your local distributor or email Nanomotion Ltd. Technical Support Department at techsupport@nanomotion.com, with detailed problem description, additions, corrections or suggestions.

Nanomotion Ltd. Worldwide Headquarters
Mordot HaCarmel Industrial Park
HaYetsira Street, PO Box 623
Yokneam 20692
Tel: +972-73-249-8000
Fax: +972-73-249-8099
Email: nano@nanomotion.com

Nanomotion Inc - US Headquarters
1 Comac Loop, Suite 14B2
Ronkonkoma
NY 11779
Tel: +1-800-8216266
Fax: +1-631-5851947
Email: nanoUS@nanomotion.com

Patent Information

The following patents apply to or relate to the products and information in this user manual.

5,453,653; 5,616,980; 5,714,833; 111597; 5,640,063; 6,247,338; 6,244,076; 6,747,391; 6,661,153; 69838991.3; 6,384,515; 7,119,477; 7,075,211; 69932359.5;1186063; 7,211,929; 69941195.5; 1577961; 4813708; 6,879,085; 6,979,936; 7,439,652; 7061158;1800356; 1800356; 1800356; 2007-533057 (pending); 2011-093431 (pending); 7,876,509; 10-2007-7009928 (pending); 200780019448.6; 7713361.9 (pending); 12/294,926 (pending); GB2008000004178 (pending); GB2009000003796 (pending); 12/398,216 (pending); GB2446428; 12/517,261 (pending); 08702695.1 (pending); 10-2009-7017629 (pending); 12/524,164 (pending); 12/581,194 (pending)

Revision History

The following table shows the last three revisions to this document.

ECO	Doc Rev	Date	Description
727	A	March 2014	Initial release of FW version 1.5.0.7

Table of Contents

Table of Contents

CHAPTER 1 INTRODUCTION

This manual provides information for developing and using Nanomotion's XCD™ Software to control Nanomotion motors with the XCD Controller. The manual is divided into the following sections,

- Overview of Nanomotion XCD Software
- XCD Commander™ - user interface that provides access to program parameters and values to allow tuning the servo loop system, and developing unique motion control systems
- XMS software information - this section includes a description of the parameters and allowed values.
- Communication Protocol

1.1 CONVENTIONS USED IN THIS MANUAL

Throughout this manual commands are shown in BOLD and parameter values are shown in italics

NOTE: Notes provide additional information that is not included in the normal text flow.

CAUTION: Caution provides information about actions that will adversely affect system performance.

Best Known Methods: Provides additional detailed information about operations and methods.

BKM

Danger: Indicates operations or activities that may cause damage to equipment or injury to personnel.

1.2 RELATED PRODUCTS

The following table lists Nanomotion products which may have this software version. Refer to your hardware product user manual to verify installed software version.

Product	Part Number
XCD HR1™ Controller Driver	XCD-HR1-BD-04
XCD HR2™ Controller Driver	XCD-HR2-BD-04
XCD HR4™ Controller Driver	XCD-HR4-BD-04
XCD HR8™ Controller Drive	XCD-HR8-BD-04
XCD HR16™ Controller Drive	XCD-HR8-BD-04
XCD™ Component	IC000028

CHAPTER 2 OVERVIEW OF SERVO LOOPS

Nanomotion's XCD Controller/Driver boards supply dynamic servo control to ensure accurate position changes. The motor control circuit consists of three major building blocks.

- Linear or rotary stage mounted on a Platform
- HR series or Edge motor
- Encoder
- XCD Controller/Driver
- Remote computer to host the XCD Commander

The Controller provides a PWM signal to an AC converter. The output AC is applied to the motor driving the stage. As the stage moves, the encoder sends position information to the Controller. The Controller matches the current position against a calculated expected position and corrects the PWM drive signal to correct motor speed.

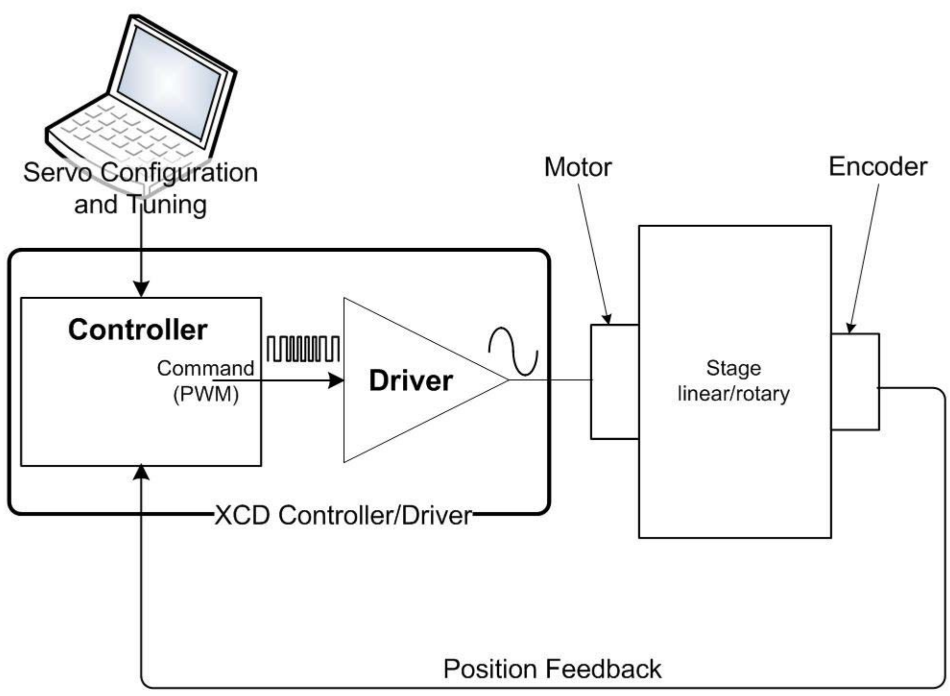

Figure 1: Servo Loop Block Diagram

2.1 POSITION PROFILE

When the Controller receives a MOVE command it calculates a position signal profile that defines the expected stage positions at 50 µsec intervals. The profile includes three phases:

- Acceleration (ACC)
- Constant velocity (VEL)
- Deceleration (ACC)
- Position (RPOS)

The ACC parameter is a <userUnit>/sec2 value used for both acceleration and deceleration. Note that for a linear stage the value is mm/sec2 and for a rotary stage it is degree/sec². During acceleration (ACC) the Controller supplies a signal that increases motor speed. After the motor reaches the configured velocity (VEL) the Controller maintains velocity by changing the drive command. As the stage approaches the target position the Controller begins to reduce the drive command.

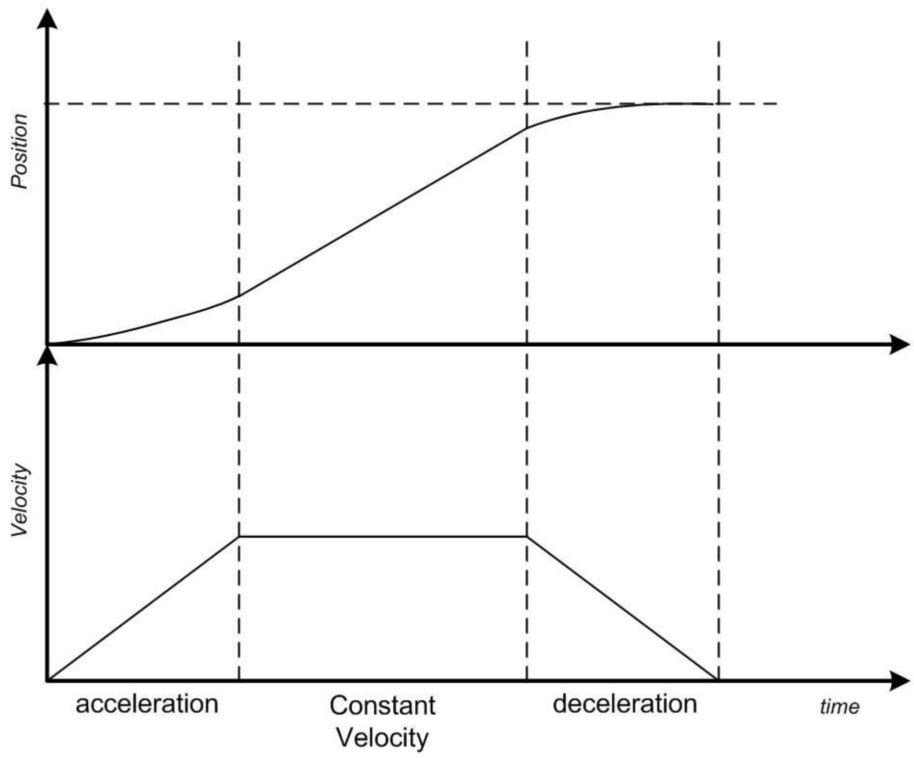

Figure 2: Position and Velocity Profile changes over time

2.2 CONTROL ALGORITHM

The XCD Controllers are based on a standard PIV (position/velocity loop) controller with a non-linear mechanism. The Controller uses several mechanisms to control movement of the stage.

- Offset mechanism
- Zero FeedForward mechanism
- Dead Zone mechanism

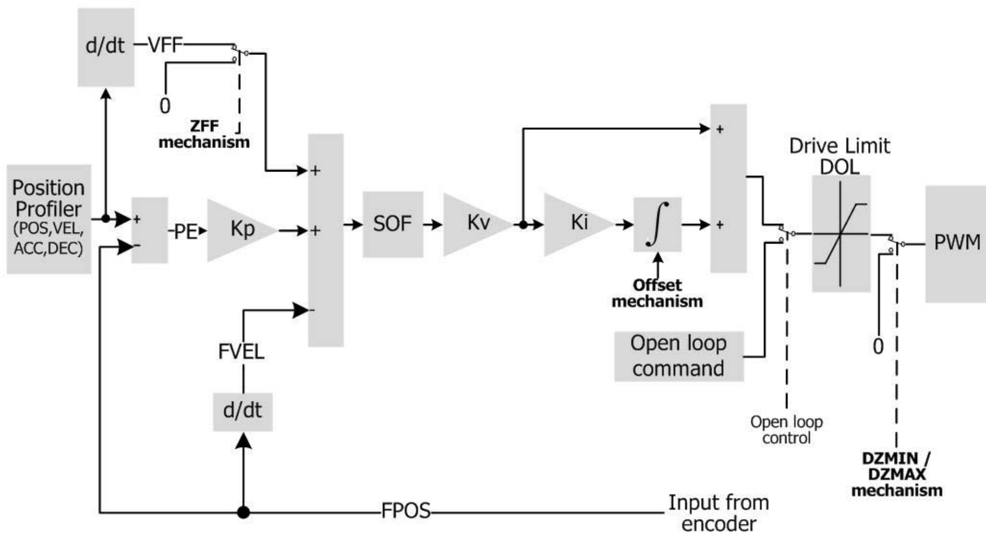

Figure 3: Controller Block Diagram

The discussions in the following sections refer to this diagram.

2.2.1 OFFSET MECHANISM

The offset mechanism provides an initial starting value to overcome the natural friction of the motor by initializing the integrator. The offset value is set using two parameters:

- Friction Positive (FRP) - compensates for friction in a positive direction.
- Friction Negative (FRN) - compensates for friction in a negative direction

These parameters are set at between 50-100% of the value required to cause the motor to move. This is determined during the initial tuning of the servo loop.

2.2.2 ZERO FEED FORWARD (ZFF) MECHANISM

The Zero feed forward mechanism improves the motors settling time. This is done by stopping the Velocity Feed Forward (VFF), reducing the speed at which the motor approaches the Target Position. As shown in the block diagram, VFF provides a value derived from the position profile. The ZFF parameter is set to a value of between 30 and 50 microns from the target position. The value of ZFF is dependent on the moving mass. For a larger mass ZFF should be increased.

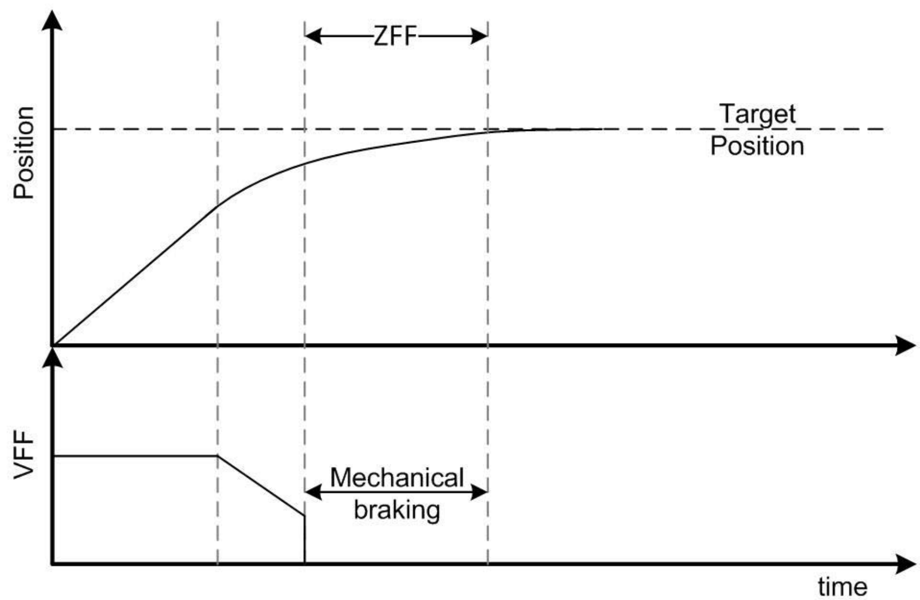

Figure 4: Motor braking on approach to Target Position

2.2.3 DEAD ZONE MECHANISM

The Dead Zone mechanism takes advantage of the motor's intrinsic friction to prevent jitter and improve the settling time. The Dead Zone is defined by two areas on either side of the motor.

- Dead Zone Minimum (DZMIN) - a range around the Target Position that is the desired final position of the motor.
- Dead Zone Maximum (DZMAX) - defines the maximum allowable final position of the motor.

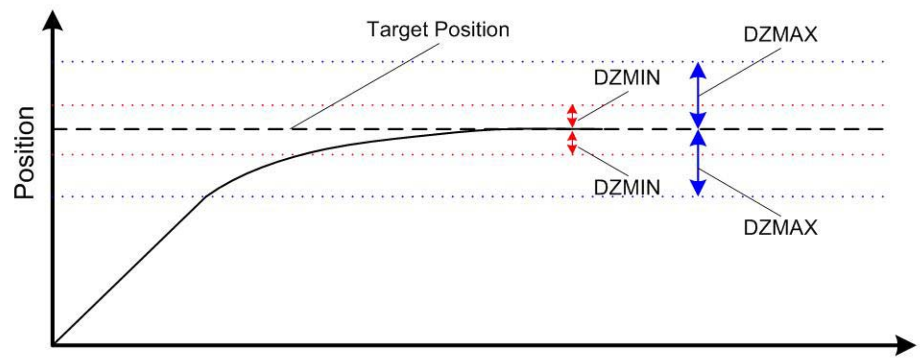

Figure 5: Dead Zone Mechanism

DZMIN is the distance from the Target Position at which the Dead Zone mechanism drops the drive command to **0**. At this point the motor's intrinsic friction applies mechanical braking to the motor.

After the motor enters DZMIN, the Controller stops monitoring the motor's position, referred to as Blackout. The default blackout time is 0.05 msec (***DZMIN Blackout, page 36*** ***). After Blackout is lifted, the Controller checks PE.***

If absolute value of PE is greater than DZMAX the Controller applies a signal to return the motor toward the Target Position (servoing again).

CHAPTER 3 XCD COMMANDER

The XCD Commander™ is an application that resides on an external Host computer and provides monitoring of the operation of the user developed motor control program. XCD Commander gives the user the ability to:

- Configure and tune the XCD Controller Motion system
- Edit and save XMS scripts
- Evaluate and troubleshoot motion control program operation
- Edit program parameters
- Save the final motion control program on the XCD Controller
- Save the final program to file on the Host and save from the Host to other Controllers

3.1 OVERVIEW OF XCD COMMANDER

The XCD Commander is a user interface that provides access to all operations involved in developing, testing, and using XMS scripts, as well as monitoring Controller and motor states.

The interface provides the following:

- Communication between XCD Commander and the XCD Controller/ Driver is based on a Host/Client relationship. XCD Commander resides on the Host (remote computer), and the XCD Controller/Driver is the Client. Communication is over either a UART cable or IIC of up to 400 kHz.
- XCD Motion Program™ (XMS Script) - The XCD Motion Program is a scripting language that gives the user the ability to customize and control Nanomotion motors. XMS uses both standard commands and variables to control motor position, stops, speed and loading.
- User interface - Provides a number of actions:
  - downloading and executing a motion program from the Host to the Controller.
  - access the Controller's flash memory to tune the servo loop parameters
  - editing of motion control programs written in XMS
  - saving programs from the Controller's flash memory to the Host for reuse on other Controllers.

3.2 WORK FLOW WITH XCD COMMANDER

the XCD Commander is typically used to tune the motion control system. The Commander is supplied with example XMS scripts that can serve as a starting point for program development. If a script is developed from scratch it must be saved with the extension (<fileName>.xms). After development open the file in the XCD Commander to verify the script. The file is opened with the parts of the script color coded. The next step is to check the operation of the script on the system by clicking Download and Execute. This downloads the script to the Controller's RAM and simulates a power on to start the program.

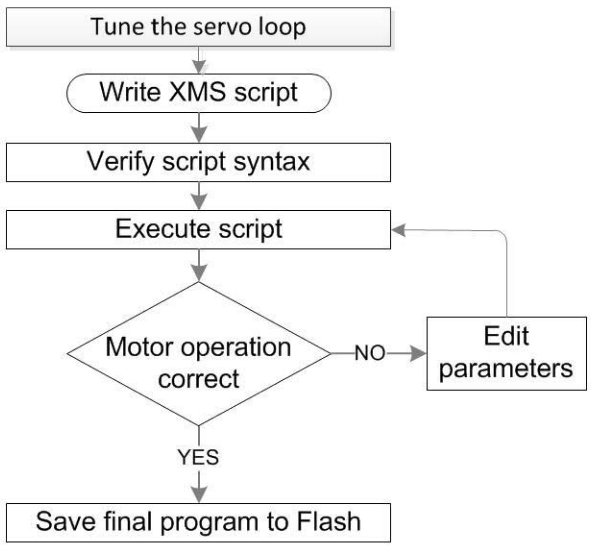

Figure 1: Basic Work Flow

If discrepancies are found in motor operation or there is a need to fine tune parts of the program specific commands can be applied and their response monitored. The parameters that require modification can be edited and the program rerun. All modifications of the program are implemented in the Controller's RAM. These changes are not saved unless they are saved to the Controller's Flash memory.

After development and testing are complete the final program is saved to the Controller's Flash. From here the program can be moved to a file (<fileName>.S19) on the Host computer. This file can be copied to other Controllers.

3.3 INSTALLING XCD COMMANDER

XCD Commander is on a USB drive or CD that is supplied with the Controller/driver. The Host computer OS must be Microsoft Windows XP, or Windows 7.

1. The XCD Software is delivered on either USB flash drive or CD. The XCD Commander will automatically launch.

   In case the installation does not launch navigate to the installation program (USB or CD) and double-click on the setup.exe file.

2. If **Security Warning** notice opens click **INSTALL**.

Figure 2: Application Install - Security Warning Window

   The installation process window shows install progress. When the installation finishes the XCD Commander GUI opens.

   If the XCD Commander GUI does not open select **Start > Programs > Nanomotion > XCD NanoCommander** to open the application.

3. If a Communication failure warning appears click **OK**.

4. In the **Communication** pane set the Address field to the correct Communication Protocol type.

   This is required because address formatting is different for different protocols. Nanomotion evaluation kits typically use RS232 communication.

BKM When using an IIC connector select the controller's IIC address.

When connecting with UART(RS232) select **0**. Set the following parameters in the Host computer's COM Port settings:

- Baud Rate: 115200
- Number of Data Bits: 8
- Parity: None
- Number of Stop bits: 1
- Flow Control: None

5. In the **Communication** pane **Port** field select the port connected to the Controller/driver from the dropdown menu.

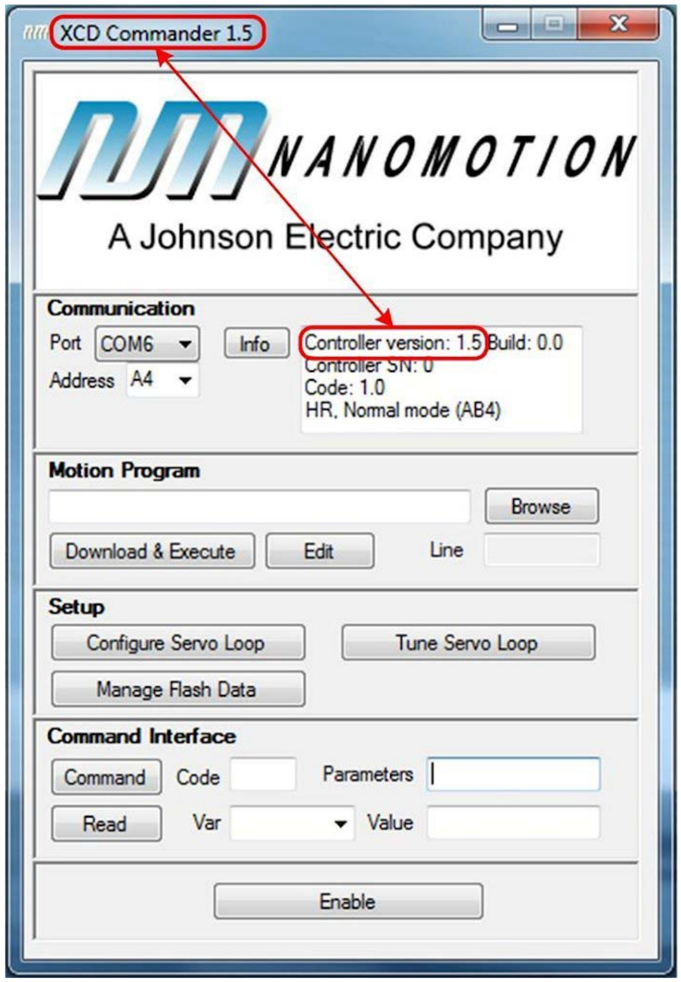

Figure 3: Commander Main Window

6. To confirm successful communication click **Info**.

   The Info field displays the Controller and application information:

   - Controller version and build
   - Controller serial number
   - Code
   - Controller/Driver type and mode

Ensure that the Controller version is the same as the XCD Commander version shown in the window header.

For example, Controller version 1.5.0.5 and XCD Commander 1.5

---

7. The XCD Commander application is ready for use.

   To uninstall XCD Commander application, use the Control Panel Program Remove feature.

3.4 QUICK START - LAUNCHING XCD COMMANDER

The XCD Software package has a folder with Nanomotion developed motion control programs. These programs provide recommended motor controls that can be used as-is or modified to meet specific user application requirements. The programs include basic servo loop tuning parameters. These parameters should be tested to ensure they provide optimal performance.

Changes to fields and selection boxes in the Configure Servo Loop and Tune Servo Loop dialogs are passed to the Controller's RAM when the field selection changes.

1. Connect the system components.

   If using a Nanomotion Evaluation kit connect the following components.

   - Motor to Controller
   - Encoder to Controller
   - Controller to the Host computer
   - Controller to external power supply

Ensure that the stage is securely mounted to the workbench. Stage movement can be extreme until properly tuned.

2. Launch XCD Commander.
3. Click **CONFIGURE SERVO LOOP**.
4. In the lower pane of the Configure Servo Loop window ensure that the four indicators are green.

   The following figure shows a Positive Limit showing red in the upper panel of the window. Change the configuration from Active Low to Active High to clear the problem.

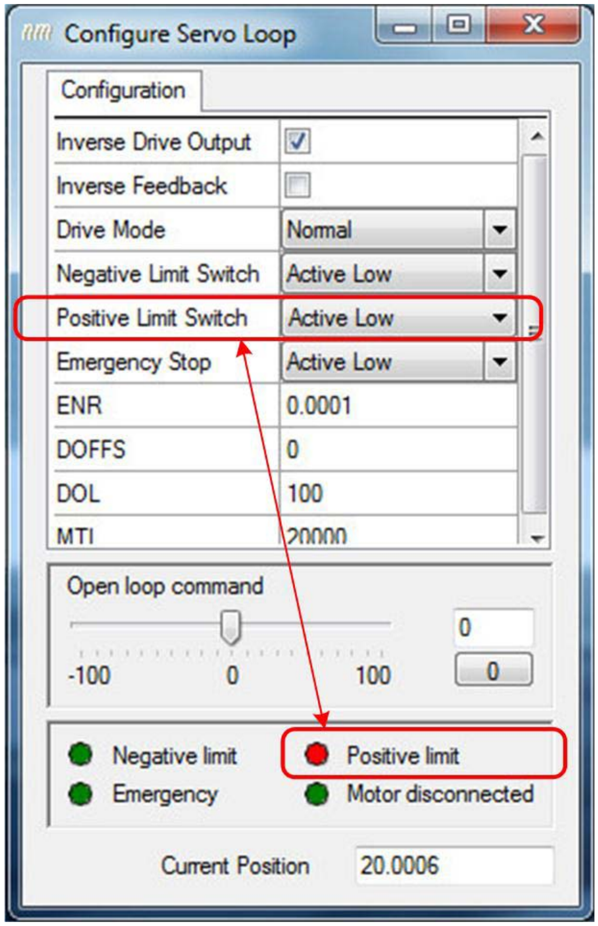

Figure 4: Example of incorrect limit switch configuration

5. Enter the ENR value.

   ENR is the encoder's resolution. For example:

   - Linear applications - If the encoder resolution is listed as 1000 counts per mm, enter 0.001 in the ENR field.
   - Rotary applications - If the encoder resolution is listed as 4096 counts per revolution, enter 360/4096=0.08789 (degrees per count) in the ENR field.

!

The maximum encoder frequency is 15M counts.

stage velocity * encoder resolution< 15M

---

6. In the **Open Loop Command** pane move the slider slowly right (positive PWM signal). When the motor begins to move, stop and record the value shown in the **Open loop command** field.

If the motor movement is in the opposite direction (as shown by a negative number in the **CURRENT POSITION** field), select or deselect **INVERSE DRIVE OUTPUT** and recheck motor action.

! The slide displacement is proportional to the PWM duty cycle.

In the Info pane, check that the Controller is in Normal mode. In this mode the motor shows a sensible Dead Zone and normally starts moving at 15-30% of PWM duty cycle.

If the motor hasn't moved at 50% of PWM, do not continue. The motor apparently has a problem that must be corrected.

7. Continue in the positive direction and record the value shown in the **CURRENT POSITION** where the motor stops.
8. Move the slider slowly left (negative PWM signal) and when the motor begins to move record the value in the field.
9. Continue in the negative direction and record the value shown in the **CURRENT POSITION** where the motor stops.
10. Click **TUNE SERVO LOOP** button in the Commander main window.
11. In the lower pane of the **Tune Servo Loop** dialog, enter values for **Position 1** and **Position 2** obtained in steps 7 and 9.

! The values should be at least 1 mm off the hard stops noted in the above steps.

12. In **BACK-FORCE WITH DELAY** enter 500 milliseconds.
13. Click **POSITION 1** and ensure that the motor moves to that position.
14. Click **POSITION 2** and ensure that the motor moves to that position.
15. If movement in both directions is ok, click **BACK-FORCE WITH DELAY**.

    The motor will begin moving between Position 1 and Position 2. Any of the following indicate a need to tune the servo loop:

    - Noisy motion
    - unstable movement
    - intolerable overshoot
    - position error
    - other undesirable motor action

3.5 TUNING OF THE XCD SYSTEM SERVO LOOP

After installing XCD Commander it is necessary to tune the connected servo loop. This operation sets the drive and feedback loop gain to ensure smooth motor operation.

! This procedure provides tuning instructions for a linear stage. Procedures for tuning a rotary stage will be added in a later revision of this document.

1. In the Commander main window click **CONFIGURE SERVO LOOP** and select the **Drive Mode**.

BKM **Normal** mode is recommended for most applications.

Linear mode may be useful for some applications with extreme requirements. However, Linear mode increases power consumption and motor wear.

2. In the Commander main window click **TUNE SERVO LOOP** and select the **Configuration** tab.
3. Set the parameters as follows:

   Press **Enter** after entering data in fields.

   - ENR (Encoder Resolution) - units appropriate for application (millimeters, degrees, radii etc.). For example:

     Linear applications - If the encoder resolution is listed as 1000 counts per mm, enter 0.001 in the ENR field.

     Rotary applications - If the encoder resolution is listed as 4096 counts per revolution, enter 360/4096=0.08789 (degrees per count) in the ENR field.

! Display of position-related variables in the XCD Commander is formatted according to the ENR value.

- DZMIN = Normal mode=2*ENR (0.02 micron)
- DZMAX = 10*ENR (0.1 micron)

! If the required Drive Mode is **Linear (AB5)**, set:

- DZMIN to 0
- DZMAX to 1*ENR
- FRP and FRN to zero

- ZFF = 0.03 - 0.05 (30-50 micron)
- FRP = 60% of value recorded in paragraph 3.4 step 5 (default value is 10)
- FRN = 60% of value recorded in paragraph 3.4 step 7 (default value is -10)
- SLN = 0 (disables negative software limit)
- SLP = 0 (disables positive software limit)
- PEL = 0 - (disables critical position error)
- TEL = 0 - (disables Temperature Error Limit)
- MTL = 20000 (Sets Motion Time Limit to 20 sec). A value of 0 disables the Motion Time Limit parameter.

The tuning values entered in step 3 are for the purpose of tuning the stage only. After tuning is completed the user should adjust these values according the application's requirements.

4. Select **SERVO LOOP PARAMETER** tab and set the following parameters:
   - Biquad 1 = Selected
   - Type = Low Pass
   - Bandwidth (Hz) = 700
   - Damping Ratio = 0.7
   - KP = 20
   - KV = 0.1
   - KI = 300
   - LI = 90
5. Open the **CONFIGURE SERVO LOOP** window.
6. Ensure that the status indicators at the bottom of the window are all green.

   If not reset the related logic selections in the dropdown menus in the top portion of the window. For example, if the Positive Limit indicator is red select Active High in the Positive Limit Switch dropdown.
7. Manually position the stage to its midpoint.
8. Move the Open loop command slider towards +100 and verify that the Current Position shows a positive value (counts positive).

   If the value was negative select the option box for Inverse Drive Output, press **ENTER** and recheck the action.

Standard orientation of the motor is when looking at the stage, and the Drive Strip is visible, motor movement to the right is positive, and movement to the left is negative.

9. Close the **Configure Servo loop** and open **Tune Servo Loop**.
10. In the lower section of the window select the Motion Parameter tab and set the parameters:
    - VEL to 50 (mm/sec)
    - ACC to 1000 (mm/sec2).
11. Manually move the stage in a positive direction until you feel the HW physical limit.
12. Enter a value that is 1mm less than the HW physical limit in the **Move to Position 1** field.
13. Manually move the stage in the negative direction until you feel the HW physical limit.
14. Enter a value that is 1mm less than the HW physical limit in the **Move to Position 2** field.
15. Click **MOVE TO POSITION 1** and verify that the stage moves to that position.
16. Click **MOVE TO POSITION 2** and verify that the stage moves to that position.
17. Enter 500 in the **Back-Force with Delay** field.
18. Click **BACK-FORCE WITH DELAY** to start the back and forth movement.
19. Ensure that the stage moves smoothly in both directions.

    The motor should move between the two entered positions in a smooth repetitive action.
20. While the motor is moving increase KV until an audible sound appears.

    Each increase should be twice the previous increment. The noise will typically be heard at the extremes of the stage movement.
21. Return the KV value to the last value before the noise began.
22. Raise the KV value very gradually (1mm increments).
23. Divide the value where the noise starts again by 2 and enter as KV.

    The stage runs smoothly with no audible noise.
24. Reset KP to 200. If the stage physically oscillates (low frequency movement), reduce KP to 100.

25. Stop the motor movement.

26. In the Commander's main window select **MANAGE FLASH DATA** and click **RAM>Flash** to save the tuning settings.

3.6 WORKING WITH XMS SCRIPTS

The user can develop a motion control program from scratch, or run a Nanomotion sample script that provides a starting point for program development. The sample scripts are in the evaluation kit software package folder titled Scripts. The folder contains two XMS scripts.

- <productName>_DEMO_KIT_Cond.xms - sets general defaults
- <productName>_DEMO_KIT_Steps.xms - sets defaults for step operation.

3.6.1 EDITING A SCRIPT

The edit function can be used for editing existing XMS scripts. By using one of the supplied sample scripts the user can experiment with the system, develop a unique script

1. In the **Motion Program** pane click **BROWSE**.
2. Browse to the folder containing XMS scripts and select a script and click **OPEN**.
3. The path to the script appears in the **Motion Program** field.
4. Click **EDIT**.
5. The Script edit panel opens. Commands and parameters are color coded.

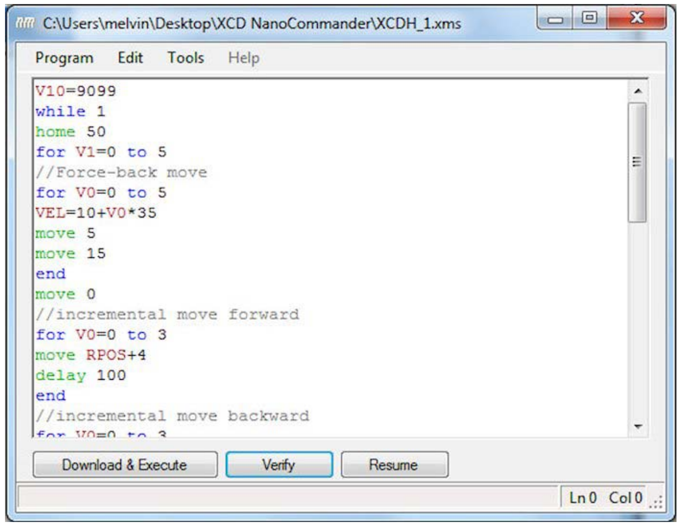

Figure 5: XMS edit window

6. Click **VERIFY** to check script syntax.

   If there are errors in the script syntax, an error message appears and the line with the error is highlighted.

7. Edit the script as required.

   The editor verifies the script syntax. An error message is displayed with the line number of the error and marks the error in the script.

   If no error is found no message is displayed

8. To test the script click **DOWNLOAD & EXECUTE** button to execute the script.

   The script is loaded to the Controller's RAM and a power on condition is simulated to start the motor. The button name changes to **STOP**. The Line field indicates the currently running script line.

9. To save the program select **Program > Save** or **Save As**.

3.7 EXECUTING AN XMS SCRIPT

To execute an XMS script

1. In the Motion Program pane click **Browse**.
2. Browse to the folder containing XMS scripts and select a script.
3. The selected script's full path is displayed in the Motion Program field.

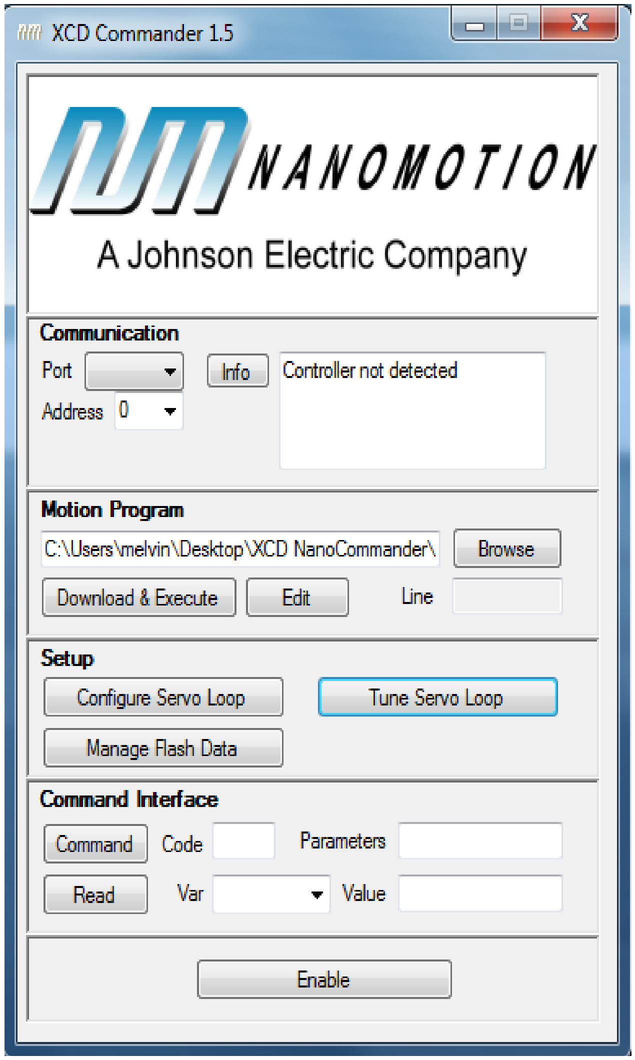

Figure 6: Commander user interface

4. To stop the script click **STOP**.

   The **STOP & DISABLE** button stops the script execution and disables the motor.

3.8 MANAGING FLASH DATA

The XMS program and parameters reside in the Controller's RAM. Changes that were made during editing will be lost on power down. They must be saved either to the Controller's Flash memory, an XMS file or as an S19 file. the S19 file can be loaded directly to the memory of multiple controllers.

After executing the program, the program's parameters are displayed in the Configure Servo Loop and Tune Servo Loop dialogs. These parameters can be edited to improve and fine tune the motor's operation. After editing these parameters the program can be rerun to test the changes.

Editing of the program cannot be done while the motor is operating. Parameter edits are copied to the Controller's RAM and affect the motor's current operation.

1. In the Commander's Setup pane click **MANAGE FLASH DATA**.

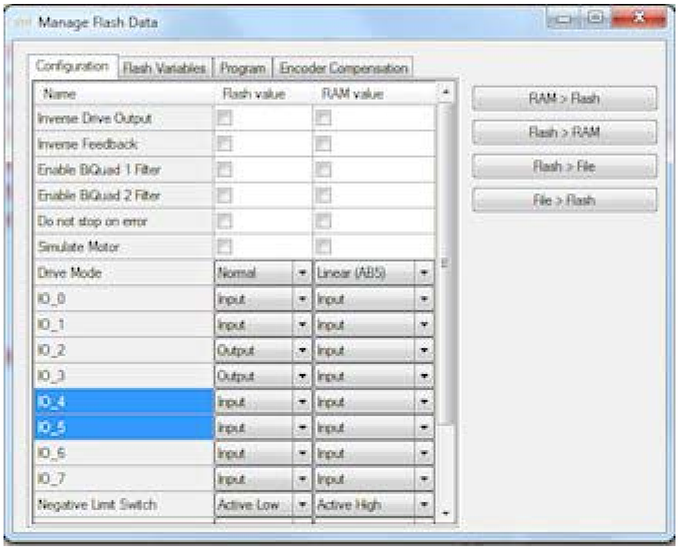

Figure 7: Manage Flash Data dialog

The dialog has three tabs that display the contents of the Flash memory. Each tab displays the contents of both the Controller's Flash and RAM.

- Configuration
- Flash Variables
- Program - shows Program size and checksum in the Flash drive and RAM

The Encoder Compensation tab provides the ability to import or export an encoder compensation value.

2. Save and upload data as follows:

- **RAM > FLASH** - saves the Controller's RAM to its Flash memory. The process overwrites the contents of the Flash memory
- **FLASH > RAM** - copies the contents of the Flash memory to RAM. This returns the RAM to its previous state.
- **FLASH > FILE** - saves the Controller's Flash data to a *.S19 file on the Host computer.
- **FILE > FLASH** - Copies an S19 file from the Host computer to the Controller's Flash memory.

3.9 Error Messages

When the Controller driver detects an error, the following actions take place:

- An error code appears on the screen.
- The currently running XMS program is aborted, and motor action is stopped.

Table 3-1: Error Codes

Error Code	Error Description
101	Position error
102	Software limit switch
103	Hardware limit switch
104	Emergency
105	Motor not connected
106	Encoder Error Two encoder signals show wrong phasing. The error may occur due to improper installation of the encoder, or because or too fast motion.
115	Motion Timeout A motion continues more than MTL milliseconds.
120	Operation Failure Special operation (homing or calibration) failed.
121	AIN protection mechanism - Overvoltage on power transistors
122	AIN protection mechanism - Overcurrent in supply circuit Limits 0-8 A, Dwell 1 ms
123	AIN protection mechanism - Voltage out of range Limits 21.5-26.5 V, Dwell 1 ms
202	The error may occur if non-waiting operations (nmove, nhome) are used in XMS script. The error occurs in attempt to command a motion while one motion is executed and another one is waiting in motion queue.
204	Mathematical Error Error occurred in expression calculation, E.g., function argument is out range.
301	Unsupported Method Unsupported method in special operation (homing or calibration) is requested.

Table 3-1: Error Codes

Error Code	Error Description
302	Timeout Timeout in special operation (homing or calibration) occurred.

CHAPTER 4 HOST COMMUNICATION PROTOCOL

4.1 COMMUNICATION CHANNELS

Communication with the host computer is provided through the following physical channels:

- UART (RS232) 115000 baud
- I2C up to 400 kHz

4.2 COMMUNICATION ADDRESS

Each controller stores its communication address, which is a number within the range of 0-254. Factory default is zero. The user can change controller's address by using the command Set Address (16). A user defined address can be stored in controller's flash memory by using command Save (13). This user defined address is retrieved at power up.

Each host command includes destination address. The destination address must be in a range of 1-254. The controller accepts host's command and responds to it, only if controller's address matches host's command destination address. Zero destination address defines broadcasting, i.e. any connected controller accepts and responds to the command.

4.3 COMMUNICATION PROTOCOL

The controller is a communication client and plays a passive role. Other side (customer processor or PC) is a communication host and plays an active role.

The communication is performed in a ping-pong manner. Each communication session includes two events:

- The host initiates communication by sending a command.
- The controller sends reply; in many cases, the reply is simply a prompt, which reports whether the command is accepted or rejected.

The host commands and the controller replies are similar in all supported communication channels.

Each host command consists of the following parts.

Table 1: Host's Command Structure Description

Part	Description
Command prefix	Command prefix depends on the communication channel. Command prefix is the same for all commands.
Command body	Command body does not depend on the communication channel. Command body is specific for each command.

Controller's reply has similar parts (see Table 3)

Table 2: Controller's Command Structure Description

Part	Description
Reply prefix	Reply prefix depends on the communication channel.Reply prefix is the same for all commands.
Reply body	Reply body does not depend on the communication cannel.Reply body is specific for each command.

4.4 PREFIXES

4.4.1 UART (RS232)

Command prefix and reply prefix are identical and consist of 4 bytes

Table 3: UART Command/Reply Prefixes Structure Description

Byte offset	Size in bytes	Content
0	1	Constant 0xE4 (228).
1	1	Constant 0xA5 (165).
2	1	Destination address.
3	1	Length of command/reply body in bytes.

4.4.2 IIC

The following table shows the Command prefix consisting of 2 bytes.

Table 4: I²C Command Prefix Structure Description

Byte offset	Size in bytes	Content
0	1	Destination address (write address).
1	1	Length of command body in bytes.

The following table shows the Reply prefix consisting of 2 bytes.

Table 5: I²C Reply Prefix Structure Description

Byte offset	Size in bytes	Content
0	1	Destination address plus one (read address). This byte is sent by the host.
1	1	Length of reply body in bytes. This byte is sent by the controller.

4.5 COMMAND BODY

4.5.1 GENERAL FORMAT

Command body is a sequence of bytes in the following order

Table 6: Command Body Structure Description

Byte offset	Byte size	Content
0	1	Command code.
1	Up to 49	Parameters.

If a command requires no parameters, the whole command body includes only one byte - the command code.

In most commands, the command code is followed by parameters. Each parameter occupies one or several bytes. Delimiting bytes are omitted between the command code and parameters or between the parameters.

Each parameter is a numerical value. Each command requires a specific format for each of its parameters. All formats are binary; the least significant byte appears first. The following formats are used.

Table 7: Command Parameters' Formats

Format	**Number of bytes**	**Range**
Int8	1	-128 to +127
Int16	2	-32768 to +32767
Real	4	-3.4*1038 to +3.4*1038 approximately (complying with IEEE 754)
ID	2	0 to 65535

Some commands require a parameter that specifies a controller variable. The variable is referenced by its numerical ID. See section 6.1.4 for variable IDs, Table 17.

4.5.2 COMMANDS TABLE

In the “Format” column, the number in parenthesis specifies the size in bytes for each parameter.

Table 8: Commands Table

Command	**Code**	**Parameters**	**Format**	**Comments**
Move	1	position	Real (4)	Move to absolute position.position defines new target position in mm.
Assign Int16	2	variable value	ID (2) Int16(4)	Assignment. The value is assigned to the variable.
Assign	3	variable value	ID (2) Real (4)	Assignment. The value is assigned to the variable.

Table 8: Commands Table

Command	Code	Parameters	Format	Comments
Home	4	method origin (opt) velocity1 (opt) velocity2 (opt)	Int8 (1) Real (4) Real (4) Real (4)	Initiate a Homing operation. The command contains the homing method, point of origin, and optional velocity. origin - (optional) defines the position at the home point. If no value is entered the system assumes zero as the origin. velocity1 - (optional) defines the first stage velocity velocity 2 - (optional) defines the second stage velocity. If omitted, the system takes the velocity1 value.
Velocity Loop	6	velocity	Real	Execute velocity loop control. The velocity parameter defines the required velocity for the stage (linear or rotary units per sec). **NOTE**: To avoid jerky movements when changing velocity make changes in small increments.
Open loop	7	command	Real (4)	Execute open loop control. Parameter command defines command value in percent, from -100 to +100.
Save parameters	13	addr 90 (0x5A)	Int8 (1) nt8 (1)	Save parameter values into flash memory. At the next start-up, the controller reads the parameters from the flash and starts with the stored parameters instead of default values. The parameters are required to prevent unintentional use of the command. Addr specifies communication address of the controller. The second parameter is constant 90 (0x5A).

Table 8: Commands Table

Command	Code	Parameters	Format	Comments
Set address	16	addr 90 (0x5A) newaddr	Int8 (1) Int8 (1) Int8 (1)	Change communication address. The first two parameters are required to prevent unintentional use of the command. addr specifies the current communication address of the controller. The second parameter is constant 90 (0x5A). newaddr specifies a new communication address of the controller.
Enable	17	-	-	Enable servo loop. While servo loop is enabled, the motor actively keeps current position and fixes deviation provided by an external force.
Disable	18	-	-	Disable servo loop. While servo loop is disabled, the motor passively resists to an external force. However, the piezo motor provides a relatively high passive force that in many cases is sufficient to keep the position.
Read version	19			Read version. The command requests information about controller firmware. See section 5.6.2 for format of controller's reply.

Table 8: Commands Table

Command	Code	Parameters	Format	Comments
Monitor	20	channel variable scale	Int8 (1) ID (2) Real (4)	Monitor variable. Being commanded, the controller in each cycle converts the variable using the scale and passes it to analog output. channel defines analog output to use: 0 - AOUT0 1 - AOUT1. variable specifies variable to monitor. scale defines a conversion factor. In each cycle, the Controller retrieves the value of the variable, multiplies it by the scale and assigns a voltage between 0 and 3.3 V. V=Var*scale*(1.65/ 100)+1.65 While active, the analog output is disconnected from the corresponding AOUT variable. To stop monitoring apply Monitor Address (21) with all zeros in parameters.

Table 8: Commands Table

Command	Code	Parameters	Format	Comments
Monitor address	21	channel address scale	Int8 (1) Int16 (2) Real (4)	Monitor variable. Being commanded, the controller in each cycle converts the specified RAM address using the scale and passes it to analog output. *Channel* defines analog output to use: 0 - AOUT0 1 - AOUT1 *Address* specifies variable to monitor. *Scale* defines a conversion factor. In each cycle, the Controller retrieves the value of the address, multiplies it by the scale and assigns a voltage between 0 and 3.3 V. V=Var*scale*(1.65/ 100)+1.65 While active, the analog output is disconnected from the corresponding AOUT variable. To stop monitoring apply Monitor Address (21) with all zeros in parameters.
Kill	23	-	-	Kill motion. Current motion is terminated, and the controller provides deceleration using KDEC parameter.
Report Int16	25	var1 var2 (opt) ... var10 (opt)	Int16 (2) Int16 (2) ... Int16 (2)	The command requests current variable values. From 1 to 10 variables can be requested in one command. Additionally to XMS variables, the command accepts IDs of pseudo-variables.

Table 8: Commands Table

Command	Code	Parameters	Format	Comments
Report	26	var1 var2 (opt) ... var10 (opt)	ID (2) ID (2) ... ID (2)	Report variable values. The command requests current variable values. From 1 to 10 variables can be requested in one command. Additionally to XMS variables, the command accepts IDs of pseudo-variables.
Position Pulse Incremental	33	start increment count	Real (4) Real (4) Int32	Start position compare pulse generation (incremental). start - specifies the first position, once the motor feedback compares to t star, the Controller generate the first pulse. increment - specifies the distance between pulses in mm. The next position for compare is set as previous position plus increment. The value is either positive or negative. count - specifies the number o pulses. If count is one, single pulse at start point is produced.
Get Configuration Revision	55	--	--	The controller returns the revision string defined in the configuration file (up to 30 characters). Indication of Configuration string in the Info window of XCD Commander.

Table 8: Commands Table

Command	Code	Parameters	Format	Comments
config	config 80	PWMvalue PWMwidth Threshold	Real Real Real	Activates calibration. The following parameters can be added: PWMvalue - specifies PWM level in percent of the maximum. If omitted, default value of 50% is used. PWMwidth - specifies time in milliseconds of PWM pulse. Between PWM pulses, pause is PWMwidth/2. If omitted, default value is 10 msec. Threshold - specifies minimum motion in millimeters: if maximum motion at any frequency exceeds the Threshold, the operation considered successful; otherwise an error is reported. If omitted, default value is 0.01 mm. The operation uses internal frequency table with frequencies from 153 kHz to 165 kHz. Stepping between frequencies is 2 kHz.
getvar	960	ID Bit (optional)	Real Real	Returns information about the Variable identified by the ID argument.

4.5.3 PSEUDO-VARIABLES

Pseudo-variables have IDs, which are different from any XMS variable. Therefore, pseudo-variables are not XMS variables and cannot be used in the XMS script. The only command that accepts pseudo-variables is Report (26). In the parameters of the Report command, the pseudo-variable IDs can be used in any combination with the XMS variables' IDs. Corresponding 4 bytes in the controller reply are filled with special content, not necessary in Real format.

The following table shows available pseudo-variables along with the corresponding 4 bytes in controller's reply. The 4 Bytes in the reply contain special content and may not be in Real format.

Table 9: Pseudo-Variables with Corresponding 4 Bytes in Controller's Reply

ID	Pseudo-Variable	Comments
Required Motion Parameters		
900	Status	Corresponding 4 bytes in controller's reply contain a bit mask with the following bits: 0 - Script Run flag (1 - script is running, 0 - not running) 1 - S_QUEUE flag 2 - S_MOVE flag 3 - S_BUSY flag 4-7 - reserved 8 - Open Loop flag (1 - motor PWM is enabled, 0 - disabled) 9 - Velocity Loop flag (1 - velocity loop is enabled, 0 - disabled) 10 - Position Loop flag (1 - position loop is enabled, 0 - disabled) 11 - reserved 12 - BiQuad filter enabled 13 - Second Biquad filter enable (1=2nd Biquad enabled, 0= 2nd Biquad disabled) 14 - non-stop mode (1=enabled, 0=disabled). In non-stop node, motor error does not stop program execution. The program continues running; motion result can be tested with pseudo-variable 96 15 - reserved 16 - Low Resolution flag 17 - HR motor flag (always 0) 18 - Inverse Feedback 19 - Inverse Drive Output 20 - Simulation mode 21-23 - reserved 24 - Logical Motion flag (internal) 25 - Hold Position flag (internal) 26 - Kill flag (internal) 27-31 - reserved

Table 9: Pseudo-Variables with Corresponding 4 Bytes in Controller's Reply

ID	Pseudo-Variable	Comments
901	Program Status	Corresponding 4 bytes in controller's reply are interpreted as follows: Bits 0-15: Currently executed program line Bits 16-31: Status values: • 0 - program was not executed • 1 - execution is in progress • 2 - program finished successfully • 3 - program was terminated by a user • 100 and above - error code (program failed)
902	Safety Disable	Corresponding 4 bytes in the controller reply contain a bit mask with the following bits: Bit 0 - Negative Limit Bit 1 - Positive Limit Bit 2 - Emergency Stop Bit 3 - Motor Not Connected 0 = indicates enabled safety signal 1 = indicates disabled safety signal. Default setting is all faults enabled.
903	Safety Inverse	Corresponding 4 bytes in controller's reply contain a bit mask with the following bits: Bit 0 - Negative Limit Bit 1 - Positive Limit Bit 2 - Emergency Stop 0 = indicates that the safety signal is enabled 1 = indicates that the safety signal is disabled. Default setting is all signals are enabled.
904	Safety State	Corresponding 4 bytes in controller's reply contain a bit mask with the following bits: Bit 0 - Negative Limit Bit 1 - Positive Limit Bit 2 - Emergency Stop Bit 3 - Motor Not Connected The result shows a raw state of the safety inputs. 0 corresponds to low voltage level 1 corresponds to high voltage level.

Table 9: Pseudo-Variables with Corresponding 4 Bytes in Controller's Reply

ID	Pseudo-Variable	Comments
905	IO direction	Corresponding 4 bytes in controller's reply contains a bit mask. Bit 0 of the bit mask defines behavior of IO_0, bit1 - IO_1, and so on up to IO_7. If a bit is zero, corresponding IO is input; if a bit is one, corresponding IO is output.
906	DZMIN Blackout	Shows the blackout time as a number between 1 and 256. Each number is an increment of 0.05 msec. For example 20=1 msec.
907	PWM Limit	The return is a real value of absolute PWM limit in percent. For most XCD products the value is 100%. The value restricts PWM output (DOUT value) in all controller modes. If DOL value is below 100%, the PWM output is restricted to the lower of two limits.
908	AIN Protection	Corresponding 16 bit values. Bit 0- AIN0; Bit 1- AIN1 and so on up to Bit15-AIN15. If AIN within the limits, the bit reads zero; if violation occurs, the bit reads one.
921-924	SPI input integer	These four pseudo variables provide access to four 16-bit integers that are read through SPI interface. Commands **config** 301/302 read four 16-bit values from SPI interface. XMS program uses getvar function to access the values.
925-926	SPI Input Real	These two pseudo variables provide access to two 32-bit Real numbers that are read through SPI interface. Commands config 301/302 read four 16-bit values from SPI interface. XMS program uses getvar function to access the values. If getvar parameter is 925, the result is the first pair of 16-bit values interpreted as one 32-bit real number. If getvar parameter is 926, the result is the second pair of 16-bit values interpreted as one 32-bit real number.
950	XMS checksum	The return is a 32-bit integer checksum of XMS program in the controller memory calculated according to Adler-32 algorithm.
951	XMS Length	The return is a 32-bit integer length of XMS program in the controller memory in bytes.
960	Last error	The return is a real value of the last error in the controller. XMS program reads the value using getvar function. XMS program running in non-stop mode can use the value to check if a motion has terminated successfully.

Table 9: Pseudo-Variables with Corresponding 4 Bytes in Controller's Reply

ID	Pseudo-Variable	Comments
990	UART address	Controller address for UART communication. A number in the range 0 to 255. The default value is zero. If the controller address is zero, the controller accepts any destination address. If the destination address is zero (broadcasting), controller accepts the command irrespective of controller address. If both controller and destination addresses are non-zero, the controller answers only if the destination address matches the controller address.
991	IIC address	Controller address for IIC communication. An even number in range from 2 to 254. The default value is 164 (0xA4). If the destination address is non-zero, the controller accepts the command only if the destination address matches the controller address. If the destination address is zero (broadcasting), controller accepts the command irrespective of controller address.

4.5.4 CONTROLLER CONFIGURATION

XMS command **config** and host command **Configure** (32) allow changing different controller parameters. To reveal actual configuration settings, the host should use the **Report** command.

Both, the script and the host commands, supply two parameters: the first parameter defines a configuration code, and the second provides a configuration value. The following table lists available configuration codes and their corresponding configuration value description.

Table 10: Configuration Codes and Configuration Value Description

Code	Value
80	Calibration The controller executes a series of back and force motions at different PWM frequencies. Then the controller selects PWM frequency that provides maximum motor velocity. Additionally to code 80, the command specifies up to four parameters: • Calibration algorithm (currently, ignored) • PWM level % (optional, default 50%) • Time of one move in milliseconds (optional, default 10) • Required move in mm (optional, default 0.01) - the function fails, if all moves appear less than the value

300	SPI interface.
	Parameters:
	• Key: 300.
	• Master:
	  • 1 - the Controller is SPI master
	  • 0 - the Controller is SPI slave.
	• Clocking:
	  • 0 - output data on the rising edge; latch data on the falling edge.
	  • 1 - output data half-cycle before the rising edge; latch data on the rising edge.
	  • 2 - output data on the falling edge; latch data on the rising edge.
	  • 3 - output data half-cycle before the falling edge; latch data on the falling edge.
	Pins: bitwise variable specifying which pins are assigned to SPI. Non-zero bit assigned a pin to SPI function; zero bit doesn't change previous pin assignment:
	• Bit 0: SOMI pin (GPIO17)
	• Bit 1: SIMO pin (GPIO16)
	• Bit 2: SPISTE (CS) pin (GPIO19)
	The function always assigns clock pin (GPIO18) to SPI.
	Bits: number of bits in one word, from 1 to 16.
	Rate: clock frequency in kHz. E.g., for 1 MHz rate, parameter is 1000.

Table 10: Configuration Codes and Configuration Value Description

Code	Value
301	SPI transfer Real The controller reads data from and sends one or two real numbers to SPI channel. Additionally to code 301, the command specifies up to three parameters: • Wait - number of input values to wait for • V1 - first number to send • V2 - second number to send (optional) The controller sends real number as two 16-bit integers. If Value2 parameter is omitted, only one real number is sent. Before sending, the controller reads four 16-bit numbers from SPI interface and stores them internally. XMS program can access them using getvar function and pseudo-variables 921º926. Parameter Wait modifies behavior as follows: If Wait is zero, the function is not waiting and reads dummy data even if no data was received. If Wait is from 1 to 4, the function waits until the specified number of 16-bit words is actually received.
302	SPI transfer Integer The controller reads data from and sends up to four integer numbers to SPI channel. Additionally to code 302, the command specifies up to five parameters: • Wait - number of input values to wait for • V1 - first number to send • V2 - second number to send (optional) • V3 - third number to send (optional) • V4 - fourth number to send (optional) The controller sends each number as 16-bit integer. If some parameters are omitted, only the specified numbers are sent. Before sending, the controller reads four 16-bit numbers from SPI interface and stores them internally. XMS program can access them using getvar function and pseudo-variables 921º926. Parameter Wait modifies behavior as follows: If Wait is zero, the function is not waiting and reads dummy data even if no data was received. If Wait is from 1 to 4, the function waits until the specified number of 16-bit words is actually received.

Table 10: Configuration Codes and Configuration Value Description

Code	Value
400	Power save The controller enters a power save mode with consumption ~20 mA. In power save mode, the controller does not execute any function; most interfaces are disabled, except the encoder interface and UART interface. The controller continues counting encoder counts. The first character sent through UART interface wakes up the controller.
900	Servo Configuration The value is a bitmask; only the following bits are meaningful: 12 - BiQuad (1 - enabled, 0 - disabled) 13: Second BiQuad (1 - enabled, 0 - disabled) 14: Non-stop mode (0 - XMS program stops if motion error occurs, 1 - XMS program continues running if motion error occurs) 16 - PWM Resolution (1 - low, 0 - high) 18 - Inverse feedback direction 19 - Inverse drive output
902	Safety Disable The value is a bitmask; only the following bits are meaningful: 0 - Negative Limit Switch (1 - disabled, 0 - enabled) 1 - Positive Limit Switch (1 - disabled, 0 - enabled) 2 - Emergency (1 - disabled, 0 - enabled) 3 - Motor Not Connected (1 - disabled, 0 - enabled)
903	Safety Inverse The value is a bitmask; only the following bits are meaningful: 0 - Negative Limit Switch (1 - inverse, 0 - normal) 1 - Positive Limit Switch (1 - inverse, 0 - normal) 2 -Emergency (1 - inverse, 0 - normal)
905	IO direction Second parameter is a bitmask. Bit 0 of the bit mask defines behavior of IO_0, bit1 - IO_1, and so on up to IO_7. Zero bit defines IO as input; non-zero bit defines IO as output. Specific XCD hardware defines initial assignment of inputs and outputs and ability of IO reconfiguration. Request for IO configuration not supported in specific hardware is ignored.
907	PWM limit Second parameter specifies absolute PWM limit in percents. For most XCD products default value is 100%. The value restricts PWM output (DOUT value) in all controller modes. If DOL value is below 100%, the PWM output is restricted to the lower of two limits.

Table 10: Configuration Codes and Configuration Value Description

Code	Value
990	UART address The value specifies controller address for UART communication and can be any number in range from 0 to 255. The default value is zero; if the controller address is zero, the controller accepts any destination address. If the destination address is zero (broadcasting), controller accepts the command irrespective of controller address. If both controller and destination addresses are non-zero, the controller answers only if the destination address matches the controller address.
991	IIC address The value specifies controller address for I2C communication; use only even numbers in range from 2 to 254. The default value is 164 (0xA4). If the destination address is non-zero, the controller accepts the command only if the destination address matches the controller address. If the destination address is zero (broadcasting), controller accepts the command irrespective of controller address.

4.6 Reply Body

4.6.1 General Format

The Reply body is a sequence of bytes in the following order.

Table 11: Reply Body Structure Description

Byte Offset	Byte Size	Content
0	1	Command code - copied from replied command.
1	1	Result: 1 - Command accepted. 2 - Command rejected.
2	Up to 48	Extension.
Total	2 ÷ 50	

For most commands, the controller sends back only two bytes, omitting the extension.

4.6.2 REPLY BODY FOR SPECIFIC COMMANDS

Read Version (19)

See Table 13 for Reply body structure description for the Read version command:

Table 12: Reply Body Structure Description for the Read Version Command

Byte Offset	Byte Size	Content
0	1	Command code - copied from replied command.
1	1	Result: 1 - Command accepted. 2 - Command rejected.
2	4	Version.
6	4	Serial number.
10	2	Application code.
Total	12	

Report (26)

The requested XMS variable is reported in 4-byte Real format complying with IEEE 754.

The requested pseudo-variable is reported in 4-byte special format (see section 5.5.3).

The following table provides the Reply body structure description for the Report command.

Table 13: Reply Body Structure Description for the Report Command

Byte Offset	Byte Size	Content
0	1	Command code - copied from the replied command.
1	1	Result: 1 - Command accepted. 2 - Command rejected.
2	4	Variable 1 in Real format.
6	4	Variable 2 in Real format (if requested).
…	…	…
38	4	Variable 10 in Real format (if requested).
Total	6 ÷ 42	

4.7 XCD MOTION SCRIPT (XMS) DESCRIPTION

4.7.1 NUMBERS

Floating Point Values

All numbers in XMS program are floating point values complying with the IEEE 754 definition of single precision arithmetic.

The values range is from -3.4*1038 to +3.4*1038 approximately.

All calculations are performed using single, precision floating numbers.

Literal Constants

In the XMS program, literal constant can appear in different formats. The format of the literal constant has no affect on its internal presentation; the controller converts each constant to a floating point number before using it in calculations.:

Table 14: Literal Constants Formats in XMS

Format	Examples
Integer	1, 20, -1078
Real	0.1, 20.35, 0.000009
Scientific	1e-5, 2.3e10
Hexadecimal	0x07FF, 0x1E23

4.7.2 UNITS

The controller supports predefined measuring units for physical values. For example, position or distance in XMS program is always specified in millimeters.

Table 15: Measuring Units for Physical Values in XMS

Value	Example of Variables	Measuring Unit
Position, distance	POS, RPOS, FPOS, TPOS	Millimeter (mm)
Velocity	VEL, RVEL, FVEL	Millimeter per second (mm/sec)
Acceleration	ACC	Millimeter per second per second (mm/sec2)

Table 15: Measuring Units for Physical Values in XMS

Value	Example of Variables	Measuring Unit
Time	TIME	Millisecond (msec)
Scaled values	AIN0, AOUT1, DOUT	Percents of maximum (%)

4.7.3 EXPRESSIONS

Expression is a formula calculating numerical value.

In its simplest form, the expression consists of a single variable or literal constant.

General expression may include the following elements:

- Variables, such as: VEL, V10, IN_0
- Literal constants, such as: 10, -0.0001, 0x0FFF
- Parenthesis: (and)
- Arithmetic operations: +, -, *, /
- Compare operations: = (equal), <> (non-equal), < (less), <= (less or equal), > (greater), >= (greater or equal)
- Logical operations: & (and), | (or), ^ (exclusive or)

4.7.4 BUILT-IN FUNCTIONS

XMS language provides several built-in functions that can be used in expressions along with variables and constants.

Few examples of using standard functions in expressions:

```
V6 = sqrt(V4*V4+V5*V5)
while abs(PE)<0.1
if getvar(960)>=100
```

Here is a list of built-in functions:

- abs - calculates absolute value of an argument. The argument can be any number.
- sqrt - calculates square root of an argument. The argument can be any zero or positive number.
- sin - calculates sine of an argument. The argument specifies angle in radians. The argument can be any number.

- cos - calculates cosine of an argument. The argument specifies angle in radians. The argument can be any number.
- tan - calculates tangent of an argument. The argument specifies angle in radians. The argument can be any number.
- getvar - returns value of a variable or a bit from a variable. The function accepts one or two arguments. The first argument specifies ID of a variable. The second argument specifies bit in the variable. If the second argument is omitted, the function returns current value of the variable. If the second argument is in range from 0 to 31, the function returns 0 or 1 according to the corresponding bit in the variable.

4.7.5 COMMANDS

Command is the major building block of a motion program. Table 16 includes syntax definition statements, The statements are structured as follows:

- bold - text specifies literal terms, which appear in the script exactly as specified.
- Italic - specifies syntax units explained in the right column. Every syntax unit belongs to one of the following groups: variable - one of the variable names expression - arithmetical/logical expression command - any sequence of the controller commands

For example, in definition `move absolute_position`, **absolute_position** is an expression that generates variety of possible lines, for example:

```
move 750
```

```
move TPOS+225
```

```
move V19*300+600
```

Definition **variable = expression** generates assignment commands, for example:

```
V9 = V9 +1
```

```
VEL = V10*10
```

The following table describes the XMS command syntax.

Table 16: Command Syntax Description

Command Syntax	Comments
variable = expression	Assignment. Right-part expression is calculated and its result is assigned to variable in the left.

Table 16: Command Syntax Description

Command Syntax	Comments
move *position*	Move to absolute position. position is an expression that defines a new target position. If command enable was not executed before, command move also enables the servo loop. After motion execution, the servo loop remains enabled, and the motor keeps final position.
nmove *position*	Non-waiting move to absolute position. The command is equivalent to move, but the program does not wait for motion end, and continues executing the next command in parallel and asynchronously to motion progress. To synchronize again to motion execution, the program should wait for motion termination using an empty loop, such as: while S_MOVE end
kill	Kill motion Current motion is terminated, and the controller enables deceleration using KDEC parameter.
enable	Enable servo loop.While servo loop is enabled, the motor actively keeps current position and fixes deviation provided by an external force. Command enable is equivalent to command move FPOS.
disable	Disable the servo loop. While the servo loop is disabled, the motor doesn't resist actively to external force. However, the piezo motor provides relatively high passive force that in many cases is sufficient to keep the position.
home *method*, *position*, *velocity1*, *velocity2*	Homing.method selects one of the standard homing methods; see section 7.1 for more details. position (optional) is an expression that sets position value in the home point. If omitted, zero is taken. velocity1 is an expression that defines velocity at the first stage of homing procedure. If omitted, VEL value is taken. velocity2 is an expression that defines velocity at the second stage of homing procedure. If omitted, velocity1 value is taken.
openloop *command*	Open loop. The controller switches to open-loop operation; command is an expression that defines drive output value.
set *variable = expression*	Set special variable. The command resembles assignment, but unlike regular assignment causes special actions. The command applies to limited set of variables that are read-only and cannot be addressed in regular assignment. Only the following variables can be specified in the command: • set FPOS= Defines a new motor position. • set TIME=0 The command resets TIME to zero and restarts counting. Right-side value is ignored; counting restarts from zero. • set S_IND=0 The command resets S_IND to zero and restarts position23 latch function.

Table 16: Command Syntax Description

Command Syntax	Comments
delay *time*	Delay. time is an expression that defines the delay time in milliseconds.
if *expression* *commands1* **else** *commands2* **end**	Conditional statement. If the expression yields a non-zero value, commands1 are executed, else commands2 are executed. The <else command2> close can be omitted.
for *variable = initial to final step* *command* **end**	FOR loop. The commands within a loop are repeated specified number of times. The loop header defines the loop variable (one of user variables V0-V19), initial value of the loop variable, final value of the loop variable, and step. Loop variable is incremented by step on each repetition.
while *expression* *commands* **end**	WHILE loop. The commands within a loop are repeated while expression yields non-zero value.
config *code*, *value*	Configure the controller. The command provides configuration of the controller. See section 5.5.4 for more details.
pause	Pause program execution. The command effectively provides breakpoint functionality. When the command is encountered, the program stops execution and waits for host commands. Use the command for program debugging with XCD NanoCommander.
ppi *start*, *increment*, *count*	Start position compare pulse generation (incremental). start specifies the first position; once the motor feedback compares to start, the controller generates the first pulse. increment specifies the distance between pulses in mm; the next position for compare is set as previous position plus increment. The value is either positive or negative. count specifies the number of pulses; if count is one, single pulse at start point is produced. See Position Compare (section 6.1.4) for details.

4.7.6 VARIABLES

All variable names in the XMS program are predefined; i.e. the user can use only these predefined variable names.

The XMS variables are subdivided into two classes:

- System variables; each system variable has predefined meaning, for example:
  - •VEL - required motion velocity
  - •FPOS - feedback position
- User variables with predefined names: V0, V1, V2 ... V19. A user variable has no predefined meaning, and can store any number required in a program.

Table 17: Required Motion Parameters

ID	Name	Comments
1	VEL	Velocity
2	ACC	Acceleration
4	KDEC	Kill deceleration - used in fault conditions; e.g., if limit switch is activated.

Table 18: Instant Reference Motion Variables

ID	Name	Comments
5	TPOS	Target position
6	RPOS	Reference position
7	RVEL	Reference velocity
8	RACC	Reference acceleration

Table 19: Instant Feedback Motion Variables

ID	Name	Comments
9	FPOS	Feedback position
10	FVEL	Feedback velocity
12	PE	Position error
52	POSI	Position latched on index pulse. See section 7.2 for details.

Table 20: Time variable

ID	Name	Comments
38	TIME	Elapsed time in milliseconds. See section 7.1 for details.

Table 21: Position Compare Variable

ID	Name	Comments
54	PPW	Position compare pulse width in milliseconds.

Table 22: Servo Loop and Drive Configuration

ID	Name	Comments
13	KP	Position loop gain
14	KV	Velocity loop gain
16	LI	Velocity loop integrator limit
17	BQA1	First Bi-Quad filter parameters
18	BQA2	
19	BQB0	
20	BQB1	
21	BQB2	
67	BQ2A1	Second B-Quad filter parameters
68	BQ2A2	
69	BQ2B2	
70	BQ2B2	
71	BQ2B2	
22	ENR	Encoder resolution (millimeters per one encoder count)
23	MFREQ	Motor frequency (PWM frequency)
24	SPRD	Servo loop sampling period (milliseconds)
40	DZMIN	Dead zone min (Nanomotion proprietary algorithm)
41	DZMAX	Dead zone max (Nanomotion proprietary algorithm)
42	ZFF	Zero feed forward (Nanomotion proprietary algorithm)
43	FRP	Friction in positive direction
44	FRN	Friction in negative direction
45	DOUT	Instant drive output (% of maximal output)
67	B2QA1	Second Biquad parameter
68	B2QA2	Second Biquad parameter
69	B2QB0	Second Biquad parameter
70	B2QB1	Second Biquad parameter
71	B2QB2	Second Biquad parameter

Table 23: Safety

ID	Name	Comments
39	DOL	Drive output limit (% of maximal output)
53	DOFFS	Drive output offset (% of maximal output)
47	SLP	Software limit positive
48	SLN	Software limit negative
49	PEL	Position error limit
51	MTL	Motion Time limit

Table 24: Analog Inputs/Outputs

ID	Name	Comments
30-33	AIN0 - AIN3	Analog inputs (%) The analog input voltage V in the range of 0 to 3.3 volts is converted to AIN value in the range of -100% to +100%. The AIN value is calculated based on the formula: AIN=100*(V-1.65) /1.65 The analog input voltage V in the range of 0 to 3.3 volts is converted to AIN value in the range of -100% to +100%. The AIN value is calculated based on the formula: AIN=100*(V-1.65) /1.65
55-66	AIN4-AIN15	Analog inputs (%)
34-37	AOUT0-AOUT3	Analog output % The analog output value in the range of -100% to +100% is converted to an output voltage V in the range of 0 to 3.3 volts. The AOUT voltage is calculated based on the formula: V = AOUT*1.65/100 + 1.65

Analog Inputs are not supported in all SW versions or Driver cards. Refer to the user manual for your Driver card.

Table 25: User Variables

ID	Name	Comments
1000÷ 1019	V0 - V19	User variables

Table 26: Flags (accept values 0 or 1 only)

ID	Name	Comments
2000-2007	IO_0 - IO_7	Digital inputs/outputs 0 - 7
2008	S_QUEUE	The motion queue is full. While the motion queue is full (S_QUEUE is 1), the nmove command is disabled and returns error.
2009	S_MOVE	The motion is in progress. The flag toggles to **1** at motion start (e.g. once move command executes). The flag toggles to 0 at motion end, once RPOS achieves TPOS.
2010	S_BUSY	Servo loop is busy. The flag is 1 while the servo loop is active. After the move/nmove command, the flags toggles to 1 synchronously with S_MOVE, but toggles to 0 (zero) usually later than S_MOVE, once the FPOS enters ±DZMIN interval around TPOS.
2011	S_IND	Index position latched. The flag indicates if the encoder index was encountered and POSI variable latched valid index position. See section 7.2 for details.
2012	S_HOME	Homing is successful. The flag is 0 after controller power-up. The controller toggles the flag to 1 after a successful Home operation. The flag indicates that absolute feedback position is available. The flag also can be assigned either in XMS script or by host command Assign (3).
2013	S_INPOS	In-position flag. The flag toggles to **1** after: • a successful motion termination • position error PE enters ±DZMIN around the target position • Blackout time has elapsed (default 6 msec) The flag toggles to **0** when: • when a motion starts • position error PE is forced out of ±DZMAX interval.

CHAPTER 5 XMS SPECIAL FUNCTIONS AND EXAMPLES

This section contains XMS special functions and examples of XMS operations.

5.1 STAGE LOCATION INFORMATION

XCD Controller does not have Stage location information. The XCD Encoder provides velocity and direction information, but does not provide information about absolute position. In order for the Controller to determine the location of the Stage it must run a HOME command. The command has one required parameter, and three optional parameters.

HOME method, position, velocity1, velocity2

- method - required parameter that defines the Homing method.
  - 50 - home on the negative hard-stop
  - 51 - home on the positive hard-stop
  - 60 - Homing on the negative hard-stop and index pulse
  - 61 - Homing on the positive hard-stop and index pulse
- position - sets position value at the home point. The default value is zero.
- velocity1 - defines the velocity during the first stage of the homing procedure. The default value is VEL.
- velocity2 - defines the velocity during the second stage of the homing procedure. The default value is velocity1.

In the following image the HOME method is 50, Home on negative hard-stop. If the position is not included in the command the position is marked as 0. This Home position will be used in all motor movements from this point forward.

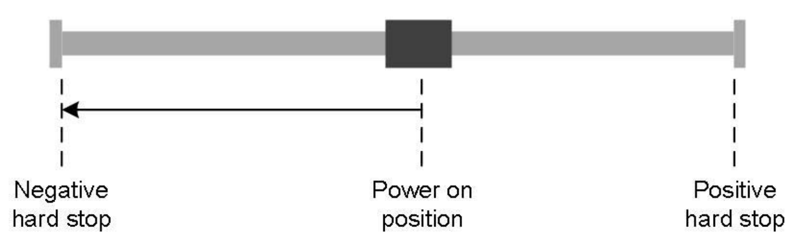

Figure 1: Homing on the Negative hard-stop

5.2 POSITION LATCH AND ENCODER INDEX

The encoder provides special input for position latch. This input should be connected to encoder index output. The function is used internally in Home functions based on encoder index.

Positive pulse on the input causes immediate latching of the current encoder position in variable POSI. The related delay is at nanosecond level and provides one-encoder-count resolution of the latched position.

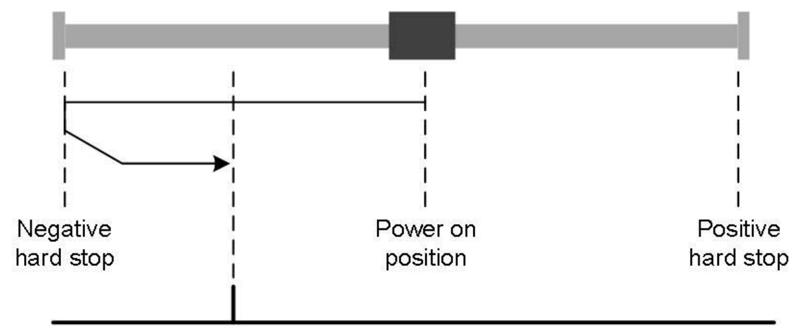

Figure 2: Home and Position Latch

Position latch function operates with two XMS variables:

- POSI - real variable
- S_IND - flag

S_IND is zero after Controller start up. Positive pulse on the latch input toggles S_IND to one and latches current encoder position in variable POSI. The value one in S_IND indicates that the latch input was activated and the value of POSI is valid.

As long S_IND remains one the next activation of the latch input will not be acted on. To repeat a position latch function, reset S_IND to zero (S_IND = 0).

5.3 XMS EXAMPLE

The following XMS example implements a sequence of actions similar to home 60,0,10 command, given the position latch input is connected to encoder index output.

Example for Home 60, 0, 10 Implementation

Command	Description
VEL=10	Move at velocity 10 mm/sec.
nmove -10000	Infinite move in negative direction.

Example for Home 60, 0, 10 Implementation

Command	Description
while PE>-1 end	Check if the motor stuck at negative hard-stop.
disable	Stop motion immediately.
set S_IND=0	Reset S_IND to zero, activate position latching.
nmove +10000	Infinite move in positive direction.
while S_IND=0 end	Wait for index pulse.
kill	Kill motion.
while S_BUSY=0 end	Wait for motion termination.
set FPOS=FPOS-POSI	Set axis origin to index position.

5.4 POSITION COMPARE

Position Compare function produces pulses on a Controller output once the motor feedback position compares to a predefined value. The function is supported in hardware; related delay is about 0.1 microseconds. Therefore, the pulse can be used for operating devices synchronously to motor position, e.g. for taking video frames.

5.4.1 POSITION COMPARE ACTIVATION

The function is activated with either XMS command ppi or Host command Position Pulse Incremental (33). In both cases, up to three parameters are specified:

- start specifies the first position; once the motor feedback compares to start, the Controller generates the first pulse.
- increment specifies the distance between pulses in mm; the next position for compare is set as previous position plus increment. The increment is rounded to integer number of encoder pulses.
- count specifies number of pulses; if count is one, single pulse at start point is produced.

Some of the parameters can be omitted. If increment and count are omitted, the Controller produces only one pulse once the motor achieves start position. If all parameters are omitted, no pulses are produced. The function without parameters effectively cancels currently active Position Compare function.

5.4.2 Position Compare Operation

Being activated with either XMS or Host command, the Position Compare function remains active until the requested number of pulse is produced. If a new ppi or Host (33) command is executed while a previous Position Compare is still active, the previous command is canceled immediately, and the new command starts operating. If the new command is issued with no parameters, the previous function is canceled, but no new function is activated.

Once a command activates Position Compare function, no pulse is produced immediately, but the Controller begins comparing feedback position with start value. The first pulse is generated once the compare result changes from not-equal to equal. The pulse width is defined by PPW value (pulse width in milliseconds). The Controller adjusts pulse width at the time of function activation; therefore, all pulses requested in a function have the same width. Assigning a new value to PPW has no effect on already active Position Compare function, but will affect the forthcoming commands.

Once a position compare occurs, the Controller produces the pulse, and verifies the remaining number of pulses. If all requested pulses have been produced, the Controller deactivates the function avoiding further pulse generation. Otherwise, the Controller adds increment to the current compare value and continues comparing.

Position Compare Timing

Table 1:

Parameter	Value	Unit
Delay from encoder pulse to output compare pulse	0.1	µsec
Minimal width of compare pulse	0.067	µsec
Maximal width of compare pulse	273.6	µsec
Maximal rate of compare pulses	20	kHz
Minimal period of the compare pulses (minimal time between adjacent pulses)	50	µsec

5.4.3 Position Compare Example

The following XMS fragment shows possible use of ppi command.

Table 2:

Command	Description
PPW=0.001	Set pulse width 1 microsecond.

Table 2:

Command	Description
move 2.1	Move to position 2.1 mm.
ppi 2.2,0.002,201	Request 201 pulses starting at position 2.2 mm with increment 2 µm.
move 2.7	Move to position 2.7 mm.

The following diagram shows the result:

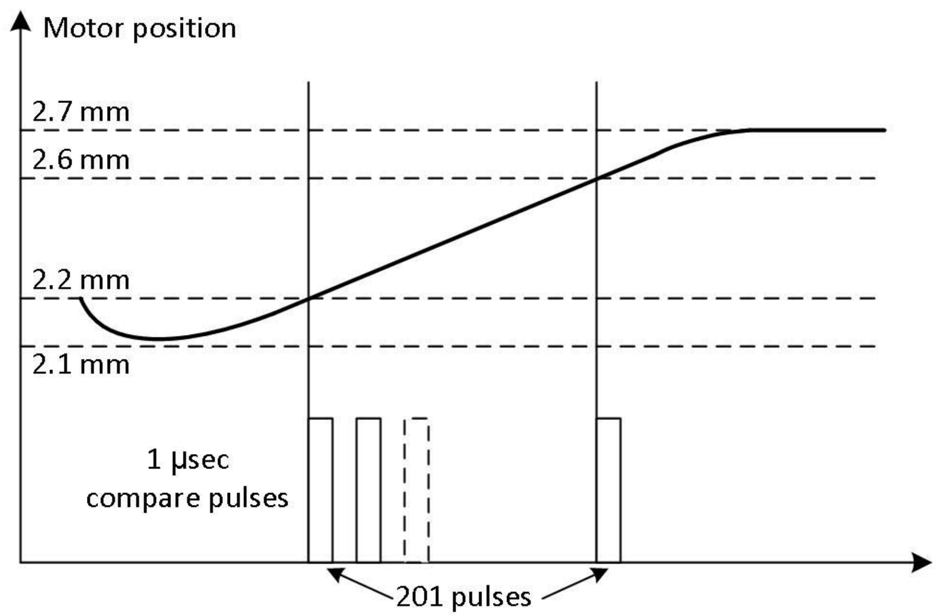

Figure 3: Motor position compare example

5.4.4 TIME VARIABLE

The Controller provides high resolution time service through the variable TIME. This variable provides time in milliseconds elapsed since the Controller start-up, or since the last executed command set TIME=0.

The resolution of time counting is 50 µsec. therefore the variable may read fractional values, such as 2345.05.

Time counting has the following limitations:

- After about two days of uninterrupted counting TIME variable rolls down to zero and starts new counting cycle.

- After about 12 minutes of uninterrupted counting the resolution of TIME begins to gradually deteriorate.

Command set TIME=0 resets the Time value to zero and restarts counting. To use full resolution, the command should be executed in the application periodically, so that using of TIME occurs not later than 12 minutes past the last set TIME=0.

5.4.5 Safe Time Measuring

The following example shows how to safely use TIME variable for measuring motion time. The XMS program repeats motion 100 times, measures full motion time (up to the motor physically enters DZMIN interval) and stores minimum motion time in V3 and maximum motion time in V4.

Example of TIME Variable for Measuring Motion Time

Command	Description
V3=10000	Initialize V3 with big value.
V4=0	Initialize V4 with small value.
for V19=0 to 99	Repeat 100 times.
move 5	Move to initial point.
while S_BUSY end	Wait for physical motion end.
set TIME=0	Reset TIME.
move 15	Move.
while S_BUSY end	Wait for physical motion end.
V0=TIME	Latch elapsed time.
if V0<V3 V3=V0 End	Update minimum time.
if V0>V4 V4=V0 end	Update maximum time.
move 5	Return to initial point
end	Repeat

5.5 XMS Script Examples

This section provides example XMS scripts for both linear and rotary applications.

5.5.1 XMS Script for a Linear Application

The following linear motion program executes a set of motions. Each motion is positioned to a point in the range of 0 to 20 mm

Linear Motion Program Example

Command	Description
V10=999	Seed of random number generator assigned to variable V10.
while 1	WHILE loop executes forever, as condition expression is always non-zero.
home 50	Homing, method 50 to the left hard-stop
for V1=0 to 5	FOR loop executes 6 times, loop variable V1 changes from 0 to 5.
// Force-back move	Comment.
for V0=0 to 5	Inner FOR loop executes 6 times, loop variable V0 changes from 0 to 5.
VEL=10+V0*35	Set a required motion velocity. Velocity starts from 10 mm/sec, then rises to 45, 80, 115, 150, and finally reaches 185 mm/sec.
move 5	Move to absolute position 5 mm.
move 15	Move to absolute position 15 mm.
end	End of inner FOR loop.
move 0	Move to absolute position 0 mm.
// Incremental move forward	Comment.
for V0=0 to 3	Inner FOR loop executes 4 times, loop variable V1 changes from 0 to 5.
move RPOS+4	Move to relative position, increment 4 mm.
delay 100	Delay for 100 milliseconds.
End	End of inner FOR loop.
// Incremental move backward	Comment.
for V0=0 to 3	Inner FOR loop executes 4 times, loop variable V1 changes from 0 to 5.
move RPOS-4	Move to relative position, increment -4 mm.
delay 100	Delay for 100 milliseconds.
End	End of inner FOR loop.
// Variable step forward	Comment.

Linear Motion Program Example

Command	Description
for V0=1 to 8	Inner FOR loop executes 8 times, loop variable V1 changes from 1 to 8.
move RPOS+0.5*V0	Move to relative position, increments 0.5, 1, 1.5, etc.
delay 100	Delay for 100 milliseconds.
End	End of inner FOR loop.
// Variable step backward	Comment.
for V0=1 to 8	Inner FOR loop executes 8 times, loop variable V1 changes from 1 to 8.
move RPOS-0.5*V0	Move to relative position, increments -0.5, -1, -1.5, etc.
delay 100	Delay for 100 milliseconds.
end	End of inner FOR loop.
end	End of outer FOR loop.
// Positioning to random points	Comment.
for V1=0 to 200	FOR loop executes 201 times, loop variable V1 changes from 0 to 200.
// random number generator (11 bits)	Comment.
V10=V10*993+1 V10=V10&0x07FF	At each cycle generate a random number between 0 and 2048. Symbol **&** designates logical **AND**. Literal 0x07FF is hexadecimal constant, equal to decimal 2047.
// V10 is random number in the range of 0 to 2048	Comment.
move 18*V10/2048	Move to random absolute position in the range from 0 to 18 mm.
delay 100	Delay for 100 milliseconds.
end	End of FOR loop.
End	End of WHILE loop.

5.5.2 XMS Script for a Rotary Application

The following motion program performs a series of tests on a rotary stage with hard-stops to determine the stage’s ability to operate correctly. The program locates the stage's hard stops, identifies the index, and then runs a four hour conditioning program.

Rotary Application with Hard-stop Motion Program Example

Command	Description
// Program HR2 motor Rotary Stage encoder renishaw 1184000 counts/rev scale in deg	
// setting motor controller configuration config 900,0x00091000	
//inverse output (bit19)High res (bit 16=0), enable biquad (bit 12)	
// setting controller motor parameters	
// home zero 0.5mm from HS	
ENR =360/1184000	define user units in degrees
V0=0	user variable for Home offset
DZMIN = ENR	DZMIN is set equal to Encoder resolution
DZMAX = 15*ENR	DZMAX is set to 15 times the Encoder resolution
PEL=5	Sets maximum position error to 5 degrees
ZFF = 0.003	Zero Feed Forward is set to 0.003 degrees
FRN = 10	Negative Friction is set to 10%
FRP = 10	Positive Friction is set to 10%
MTL = 60000	Maximum Time Limit is set to 60 seconds
KP = 300	
KV = 0.5	
KI = 300	
LI = 60	
VEL = 40	Velocity is set to 40 deg/sec
ACC = 500	Acceleration is set to 500 deg/second^2
KDEC = 1000	Kill deceleration at 1000 deg/sec^2
//Find the negative hard-stop	
nmove -500	move in a negative direction to a position that is larger than the hard-stop. In this case
while PE>-0.5 end	The loop continues as long as the position error (PE) is greater than -0.5 degree.
disable	When the PE is less than -0.5 stop the loop because the hard-stop has been found, and disable motor.
//Find the Position Index and set the Index Position to Home offset (V0)	
nmove 360	
set S_IND=0	
while S_IND=0 end	
kill	
delay 100	
set FPOS=FPOS-POSI-V0	For V0=0 set the index position to 0 position
//Find Home offset	
nmove -360	

Rotary Application with Hard-stop Motion Program Example

Command	Description
while PE>-0.5 end	
disable	
//	
V0=FPOS+185	This value is the distance of the Index from the middle of the stage
set FPOS=-5	Set the negative hardstop position to -5 degrees
VEL=120	set the velocity to 120 deg/sec
V2=0	is the maximum positive position error when the reference velocity is in the constant velocity range.
V3=0	is the minimum position error when the reference velocity is in the constant velocity range.
V4=0	is the maximum positive PWM (DOUT)
V5=0	is the minimum negative PWM (DOUT)
//Finds maximum and minimum position error, and maximum and minimum DOUT over the total stage travel during a four hour period	
V19=TIME	start time
while (TIME-V19) < (4*3600000) nmove 360	execute the following lines when the current time minus start time is less than 4 hours
delay 2	
while RVEL<VEL end	while in the acceleration stage (RVEL<VEL) the loop will continue
while RVEL=VEL if PE>V2 V2=PE end	Measures the maximum and minimum position error and the DOUT in the constant velocity range.
if PE<V3 V3=PE end	
if DOUT<V4 V4=DOUT end	
while S_BUSY end	
delay 1000	
nmove 0	
delay 2	

Rotary Application with Hard-stop Motion Program Example

Command	Description
while RVEL>-VEL end	while in the acceleration stage (RVEL>-VEL) the loop will continue
while RVEL=-VEL if PE>V2 V2=PE end	Measures the maximum and minimum position error and the DOUT in the constant velocity range.
if PE<V3 V3=PE end	
if DOUT>V5 V5=DOUT end	
while S_BUSY end	
delay 1000 end	

CHAPTER 6 COMMUNICATION EXAMPLES

This section provides examples of command and response sequences and Byte.

6.1 EXPLANATION OF COMMAND-RESPONSE SEQUENCE

This example shows the sequence for the REPORT [0x1A (26)] command with the variable STATUS [0x384 (900)]. The REPORT command can include up to ten variables. Each variable is identified by four Bytes.

In the command and reply examples below, the Bytes are shown in the order they are transmitted via RS232.

E4 A5 A4 03 1A 84 03

prefix body

Figure 1: Byte Sequence for REPORT STATUS Command

The Command prefix contains four Bytes. The first two are constant values. The third Byte is the address of the Controller and the fourth Byte is the length of the Command body, in this case three Bytes.

The fourth position in the sequence begins the Command body and defines the Command. In this case REPORT (26). The next Bytes define the variables for the Command. In this example the Variable code is 900 (0x384) and requires two Bytes, defined by the fifth and sixth positions. Because the Bytes that define the Variable are transmitted as little endian, the LSB (0x84) appears first in the stream and the MSB (0x03) appears second.

Table 1: Command Prefix and Body Contents

Byte Offset	Byte Value Hex (decimal)	Comments
0	0xE4 (228)	Constant 0xE4 (228)
1	0xA5 (165)	Constant 0xA5 (165)
2	0xA4 (164)	Destination address
3	0x03 (3)	Length of the command body
4	0x1A (26)	Command REPORT
5	0x84 (132)	LSB of variable STATUS (900)
6	0x03 (3)	MSB of variable STATUS (900)

The Reply also contains a prefix and body.

$\underbrace{\text{E4 A5 00 06}}_{\text{prefix}}\ \underbrace{\text{1A 01 0D 17 09 01}}_{\text{body}}$

Figure 2: Byte Sequence of REPORT STATUS Reply

The Reply prefix contains four Bytes. The first two are constant values. The third Byte is the destination address, the Host that send the Command. The fourth Byte is the length of the reply.

The Reply body begins at Byte offset 4. This position is the code of the command being replied to. Byte offset 5 indicates if the Command was accepted. Byte offsets 6 thru 9 provide the 31 bits available in the STATUS reply

Table 2:

Byte Offset	Byte Value Hex (decimal)	Comments
0	0xE4 (228)	Constant 0xE4 (228).
1	0xA5 (165)	Constant 0xA5 (165).
2	0x00 (0)	Destination address.
3	0x06 (6)	Length of the reply body.
4	0x1A ( )	Last command code.
5	0x01 (1)	Command accepted
6	0x0D ( )	Bits 0- 7
7	0x17 ( )	Bits 8-15
8	0x09 ( )	Bits 16-23
9	0x01 ( )	Bits 24-31

The 31 bits of the STATUS variable reply, Byte offsets 6-9, are shown below:

Table 3:

Bit	Function
0	Script Run flag
1	S_QUEUE flag
2	S_MOVE flag
3	S_BUSY flag
4-7	reserved
8	Open Loop flag
9	Velocity Loop flag

Table 3:

Bit	Function
10	Position Loop flag
11	reserved
12	BiQuad filter enabled
13	Second BiQuad filter enable
14	non-stop mode
15	reserved
16	Low Resolution flag
17	HR motor flag (always 0)
18	Inverse Feedback
19	Inverse Drive Output
20	Simulation mode
21-23	reserved
24	Logical Motion flag (internal)
25	Hold Position flag (internal)
26	Kill flag (internal)
27-31	reserved

6.2 RS232 EXAMPLES

The examples below show RS232 byte sequences for few typical commands. Destination and controller address for RS232 communication is zero.

6.2.1 MOVE MOTOR TO POSITION 2.5 (MILLIMETERS)

The Host sends the following sequence of bytes:

Table 4:

Byte Offset	Byte Value	Comments
0	0xE4 (228)	Constant 0xE4 (228).
1	0xA5 (165)	Constant 0xA5 (165).
2	0x00 (0)	Destination address.
3	0x05 (5)	Length of the command body.
4	0x01 (1)	Command MOVE
5	0x00 (0)	2.5 (Real format)

Table 4:

Byte Offset	Byte Value	Comments
6	0x00 (0)	
7	0x20 (32)	
8	0x40 (64)	

The Controller responds with acknowledge:

Table 5:

Byte Offset	Byte Value	Comments
0	0xE4 (228)	Constant 0xE4 (228).
1	0xA5 (165)	Constant 0xA5 (165).
2	0x00 (0)	Destination address.
3	0x02 (2)	Length of the reply body.
4	0x01 (1)	Last command code.
5	0x01 (1)	Acknowledge - command accepted.

6.2.2 SET MOTION VELOCITY 70 (MM/SEC)

The Host sends the following sequence of bytes:

Table 6:

Byte Offset	Byte Value	Comments
0	0xE4 (228)	Constant 0xE4 (228)
1	0xA5 (165)	Constant 0xA5 (165)
2	0x00 (0)	Destination address.
3	0x07(7)	Length of the command body
4	0x03 (3)	Command ASSIGN
5	0x01 (1)	ID of VEL variable
6	0x00 (0)	
7	0x00 (0)	70 (Real format)
8	0x00 (0)	
9	0x8C (140)	
10	0x42 (66)	

The Controller responds with acknowledge:

Table 7:

Byte Offset	Byte Value	Comments
0	0xE4 (228)	Constant 0xE4 (228).
1	0xA5 (165)	Constant 0xA5 (165).
2	0x00 (0)	Destination address.
3	0x02 (2)	Length of the reply body.
4	0x03 (3)	Last command code.
5	0x01 (1)	Acknowledge - command accepted.

6.2.3 READ FEEDBACK POSITION

The Host sends the following sequence of bytes:

Table 8:

Byte Offset	Byte Value	Comments
0	0xE4 (228)	Constant 0xE4 (228).
1	0xA5 (165)	Constant 0xA5 (165).
2	0x00 (0)	Destination address.
3	0x03(3)	Length of the command body.
4	0x1A (26)	Command REPORT
5	0x09 (9)	ID of FPOS variable
6	0x00 (0)	

The Controller responds with sending actual position (FPOS) value; assume actual position is 3.11 mm:

Table 9:

Byte Offset	Byte Value	Comments
0	0xE4 (228)	Constant 0xE4 (228).
1	0xA5 (165)	Constant 0xA5 (165).
2	0x1A (26)	Last command code.
3	0x01 (1)	Acknowledge - command accepted.
4	0x3D (61)	3.11 (Real format)

Table 9:

Byte Offset	Byte Value	Comments
5	0x0A (10)	
6	0x47 (71)	
7	0x40 (64)	

6.3 IIC EXAMPLES

The examples below show IIC byte sequences for few typical commands.

All examples imply default IIC controller address 0xA4. If actual controller address was changed, the destination and controller address fields should be changed accordingly.

6.3.1 MOVE MOTOR TO POSITION 2.5 (MILLIMETERS)

The Host sends the following sequence of bytes:

Table 10:

Byte Offset	Byte Value	Comments
0	0xA4 (164)	Destination address.
1	0x05 (5)	Length of the command body.
2	0x01 (1)	Command MOVE
3	0x00 (0)	2.5 (Real format)
4	0x00 (0)	
5	0x20 (32)	
6	0x40 (64)	

The Controller responds with acknowledge:

Table 11:

Byte Offset	Byte Value	Comments
0	0xA5 (165)	Destination address plus one (sent by the host).
1	0x02 (2)	Length of the reply body.
2	0x01 (1)	Last command code.
3	0x01 (1)	Acknowledge - command accepted.

6.3.2 SET MOTION VELOCITY 70 (MM/SEC)

The Host sends the following sequence of bytes:

Table 12:

Byte Offset	Byte Value	Comments
0	0xA4 (164)	Destination address.
1	0x07(7)	Length of the command body.
2	0x03 (3)	Command ASSIGN
3	0x01 (1)	ID of VEL variable
4	0x00 (0)	
5	0x00 (0)	70 (Real format)
6	0x00 (0)	
7	0x8C (140)	
8	0x42 (66)	

The Controller responds with acknowledge:

Table 13:

Byte Offset	Byte Value	Comments
0	0xA5 (165)	Destination address plus one (sent by the host).
1	0x02 (2)	Length of the reply body.
2	0x03 (3)	Last command code.
3	0x01 (1)	Acknowledge - command accepted.

6.3.3 READ FEEDBACK POSITION

The Host sends the following sequence of bytes:

Table 14:

Byte Offset	Byte Value	Comments
0	0xA4 (164)	Destination address.
1	0x03(3)	Length of the command body
2	0x1A (26)	Command REPORT
3	0x09 (9)	ID of FPOS variable

Table 14:

Byte Offset	Byte Value	Comments
4	0x00 (0)	

The Controller responds with sending actual position (FPOS) value; assume actual position is 3.11 mm:

Table 15:

Byte Offset	Byte Value	Comments
0	0xA5 (165)	Destination address plus one (sent by the host).
1	0x06 (6)	Length of the reply body.
2	0x1A (26)	Last command code.
3	0x01 (1)	Acknowledge - command accepted.
4	0x3D (61)	3.11 (Real format)
5	0x0A (10)	
6	0x47 (71)	
7	0x40 (64)	

INDEX

E

F

G

H

K

L

M

O

P