






# nPoint LC.400 Series DSP Controller



## EPICS Developers Manual

Issue:	1.0
Date:	13 <sup>th</sup> September 2013

	NAME	DATE	SIGNATURE
Prepared by	Philip Taylor, Observatory Sciences Ltd.	10 September 2013	
Checked by	Alan Greer, Observatory Sciences Ltd.	12 September 2013	
Released by	Philip Taylor, Observatory Sciences Ltd.	13 September 2013	

# TABLE OF CONTENTS

## Contents

1	Scope.....	3
2	Reference Documents .....	3
3	Introduction.....	4
4	Requirements .....	4
4.1	Hardware Requirements.....	4
4.2	Software Requirements .....	4
5	FTDI Library for USB .....	4
6	nPoint EPICS Software Installation.....	5
6.1	Installation on Linux .....	5
6.2	libFTDI Installation .....	5
6.3	Enabling Access to the USB Device.....	5
6.4	nPoint Software Installation.....	6
6.5	Installation on Windows .....	7
6.6	EPICS on Windows .....	7
6.7	libFTDI Installation .....	7
6.8	nPoint Software Installation.....	7
6.9	Using libFTDI under Windows .....	8
7	Running the Test Application .....	10
7.1	Running the EPICS Database .....	10
7.2	Running the EDM Screens.....	11
7.3	Digital I/O Control Screen .....	11
7.4	Servo Loop Control & Tuning.....	12
7.5	Static Digital Positioning .....	12
7.6	Digital Trigger I/O Configuration.....	12
7.7	Wavetable Data Control Screen .....	14
8	Using the Device Support Code.....	17
8.1	Supported EPICS Records .....	17
8.1.1	Single Data Item I/O .....	17
8.1.2	Array Data I/O .....	17
8.2	Configuration of EPICS Records and Application .....	17

## 1 Scope

This document describes the installation and use of EPICS device support code for the nPoint LC.400 Series DSP Controller. The LC.400 series DSP controllers are specially designed to control nPoint nanopositioners. They are available in 1-channel, 2-channel and 3-channel configurations. A basic nanopositioning system consists of the nanopositioner, the controller and software.

The controller can be controlled through a USB interface and this document describes the EPICS software written to control the unit via this interface.

The Windows kernel drivers for the LC.400 USB interface are provided by Future Technology Devices International (FTDI) Ltd. The FTDI chip in the LC.400 series controllers has two ports (A and B), however the controller only uses Port A. The FTDI drivers provide both a “Virtual COM Port” interface and a “direct” interface (FTDI refers to the direct interface as “D2XX”). The FTDI D2XX interface provides the fastest communication speed for applications where communication speed is critical and this interface is the one used by the EPICS software described here.

With Linux (kernel version 2.6) the FTDI\_SIO driver is used, which should be automatically loaded when the device is connected.

## 2 Reference Documents

- [RD1] IOC Application Developer’s Guide, *Marty Kraimer et al.*
- [RD2] EPICS R3.14 Channel Access Reference Manual, *Jeffrey O. Hill.*
- [RD3] LC.400 Series DSP Controller manual (V1.10). *nPoint Inc.*
- [RD4] asynDriver: Asynchronous Driver Support, *Marty Kraimer, Eric Norum and Mark Rivers.*

### 3 Introduction

The nPoint EPICS software module contains driver and device support code for EPICS records to be used with the nPoint LC.400 motion controller.

The module contains an EPICS database template file containing a set of records providing the features provided over the LC.400 command interface described in document [RD3]. Also contained are EDM screens that can be used to interact with a database generated by the template database files. A demonstration application is provided, configured to work with an LC.400 controller with up to three nanopositioner channels.

### 4 Requirements

This section details the hardware and software requirements for using the NPOINT EPICS device support code.

#### 4.1 Hardware Requirements

The following hardware is required to run the device support code:

- Standard PC with a USB port.
- An nPoint LC.400 controller connected to the USB port. Note that the EPICS software currently supports only a single controller connected to the PC. If multiple controllers are connected to the PC, the first controller detected will be used.

#### 4.2 Software Requirements

The following software is required to run the driver and device support code:

- Linux or Microsoft windows 32 bit operating system. The device support code has been written using general asyn, StreamDevice and EPICS base function calls and methods and as a result will be fully functional on any system that can compile the required versions of EPICS base and the asynDriver module. The device support code was written on CentOS 6.2 and tested on CentOS 6.2 and Windows XP.
- libFTDI and libusb (version 1.x), see below for further details.
- EPICS (version 3.14.12.x)
- EPICS module Asyn (version 4.18)
- EPICS module StreamDevice (version 2.6)
- EDM to run engineering screens, if they are required.

### 5 FTDI Library for USB

The nPoint LC.400 controller uses the FTDI chip for its USB interface. The libFTDI open source library is used by this EPICS software module to talk to the FTDI chip hardware. For details and software downloads see:

<http://www.intra2net.com/en/developer/libftdi>

Note that this driver is open source and is different to that supplied by the FTDI chip manufacturer.

The libFTDI library can be easily built from source under Linux, see section 6.2 below for further details.

A pre-built binary for libFTDI on Windows is available See section 6.9 below for details of where this can be obtained and how to use the libFTDI library under Windows.

## 6 nPoint EPICS Software Installation

### 6.1 Installation on Linux

#### 6.2 libFTDI Installation

The FTDI library requires that the libusb 1.x library is installed. This may already be available on your Linux system, if not ensure that version 1.0 (or later) of libusb is installed using your package management utility (yum on CentOS). The library package may be called `libusb1-devel`.

The libFTDI source code may be downloaded from the website listed above either as a Source TGZ or Source RPM file. Having unpacked the source code into a suitable directory, consult the README and README.build instruction files. Additional utility packages may be needed for the build (e.g. `cmake`). Follow the supplied instructions to build and install the FTDI library: this should just consist of issuing the commands '`cmake`' followed by '`make install`'.

The libusb1 and libftdi libraries should be installed in directory `/usr/lib` on a Linux system.

#### 6.3 Enabling Access to the USB Device

To check whether the nPoint device has connected correctly to the Linux PC, type the following command at a shell command prompt:

```
% lsusb | grep -i ft
```

```
Bus 001 Device 006: ID 0403:6010 Future Technology Devices International, Ltd FT232C Dual USB-UART/FIFO IC
```

The Vendor ID should be shown as 0403 and the Product ID as 6010.

The default access permissions for the nPoint device mean that a Linux user (not logged in as root) will be unable to access the device. Although the permissions may be manually set each time the device is connected, a more robust solution is to define a udev rule to set this automatically (udev is the Linux system for managing devices).

A udev rule is written in a file which is installed in the directory `/etc/udev/rules.d`. A suitable access setting rule for this device would be:

```
ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6010", ATTRS{serial}=="7040008",  
MODE="666"
```

Note the Vendor ID used is 0403 and the Product ID is 6010. This would be put into a file called `99-ftdi.rules` and copied into directory `/etc/udev/rules.d`. The effect would be that whenever the specified USB device is plugged in, its Linux device permissions would automatically be set to R/W for all users.

## 6.4 nPoint Software Installation

This section will cover installation of the device support code for the nPoint LC.400 on a Linux operating system. Before performing the following installation, ensure that EPICS version 3.14.12.x has been installed and the EPICS software modules asynDriver V4.18 and StreamDevice V2.6 have been installed.

Unpack the NPOINT tar file in a suitable location in your file system and cd into the top-level directory that has been created. In there you will see the following files and directories:

Directory	Description
nPointApp	This directory contains the device support code in the src directory
nPointTestApp	This directory defines a test application: a template database for a single channel, along with a substitutions file to generate a database configured for an NPOINT with three channels. EDM screens are provided to use with the database
configure	The configuration directory. The RELEASE file must be edited to reflect the local configuration
iocBoot	The boot directory for the test application. A directory below this contains the startup script
Docs	Documentation files
proto	This directory contains the protocol definition file used by the Streams software module
Makefile	The top-level Makefile. Each directory in the application contains a Makefile
runGUI	Linux only. A Bash script to start the EDM application and open the test EDM screens

Before making the application it is necessary to add references to the EPICS installation, as well as the Asyn and StreamDevice installation. Cd into the configure directory and edit the RELEASE file. Update the lines that define the location of the Asyn and StreamDevice modules to point to wherever these modules are installed on your system. For example

```
ASYN=/usr/software/epics/R3-14-12-3/asyn4-18
STREAM=/usr/software/epics/R3-14-12-3/stream2-6
```

Also update the line that defines the location of your EPICS installation. For example

```
EPICS_BASE=/usr/software/epics/R3-14-12-3
```

Save any changes and exit. Cd back up to the top level. Finally the host architecture must be defined before the build can commence. From a terminal enter the following line:

```
export EPICS_HOST_ARCH=linux-x86
```

cd to the top level and type make to build the device support code and the test application. The build should complete with no errors.

Note that the Makefile in directory nPointApp/src defines a fixed location for the libftdi library (this is /usr/lib for a Linux build).

## 6.5 Installation on Windows

### 6.6 EPICS on Windows

EPICS installation and build on a Windows platform can be done using various tool-sets. For this software module, this was done using the Linux compatible tools provided by the MinGW/MSYS package. Refer to the EPICS website page <http://www.aps.anl.gov/epics/base/win32.php> for further information about building EPICS on Windows.

For building and testing this EPICS module under Windows, EPICS version R3-14-12-3 was built with the MinGW/MSYS toolchain. Tools used for the EPICS build included:

- MinGW gcc/g++ compilers (V4.7.2 used)
- MinGW Make utility (V3.81 used)
- MSYS bash shell (V3.1.17)
- Strawberry Perl (V5.16.2.1-32bit used)

### 6.7 libFTDI Installation

A prebuilt libFTDI development kit for MinGW is available directly as a zip file in the Google Code “picusb” project Downloads page at

[http://code.google.com/p/picusb/downloads/detail?name=libftdi1-1.0\\_devkit\\_mingw32\\_17Feb2013.zip](http://code.google.com/p/picusb/downloads/detail?name=libftdi1-1.0_devkit_mingw32_17Feb2013.zip)

The required .lib and .h files from this package are included in the nPoint EPICS software module (in folder nPointApp), so downloading and installing using this zip file is not necessary.

### 6.8 nPoint Software Installation

First ensure you have installed the ‘tar’ utility. Unpack the NPOINT tar file in a suitable location in your file system and cd into the top-level directory that has been created. Cd into the top folder. In here you will see the following files and folders:

Folder	Description
nPointApp	This folder contains the device support code in the src folder. For Windows, an FTDI/libusb library folder is provided, called mingw-lib.
nPointTestApp	This folder defines a test application: a template database for a single channel, along with a substitutions file to generate a database configured for an NPOINT with three channels.
configure	The configuration folder. The RELEASE file must be edited to reflect the local configuration
iocBoot	The boot folder for the test application. A folder below

	this contains the startup script
Docs	Documentation files
proto	This folder contains the protocol definition file used by the Streams software module
Makefile	The top-level Makefile. Each folder in the application contains a Makefile

Before making the application it is necessary to add references to the EPICS installation, as well as the Asyn and StreamDevice installation. Cd into the configure folder and edit the RELEASE file. Update the lines that define the location of the Asyn and StreamDevice modules to point to wherever these modules are installed on your system. For example

```
ASYN=C:/epics/R3-14-12-3/asyn4-18
STREAM=C:/epics/R3-14-12-3/stream2-6
```

Also update the line that defines the location of the EPICS installation.

```
EPICS_BASE=C:/epics/R3-14-12-3
```

Save any changes and exit. Cd back up to the top level. Finally the host architecture must be defined before the build can commence. From a terminal enter the following line:

```
export EPICS_HOST_ARCH=win32-x86-mingw
```

There is a test application supplied with the device support code. If this is to be used then the startup script should be altered. Cd into the iocBoot/iocnPoint directory. Edit the Makefile and ensure the ARCH variable is set to win32-x86-mingw (by default it will be defined for a Linux build). Now cd back to the top level and type make to build the device support code and the test application. The build should complete with no errors.

Note that the Makefile in folder nPointApp/src defines a fixed location for the libftdi and libusb1 libraries (this is the mingw-lib folder in nPointApp/src, for a Windows build).

## 6.9 Using libFTDI under Windows

Getting libFTDI to work under Windows is not straightforward, due to possible conflicts with incompatible Windows USB driver software. To ensure the correct USB Windows driver software is installed, the following instructions should be followed (which have only been tested under Windows XP):

1. While it is switched on, connect the nPoint controller to the Windows PC. If it requests installation of driver software, Cancel this operation (this may need to be repeated). This is to avoid incorrect driver software being installed.
2. Manually install the FTDI VCP Windows driver from the following website:  
<http://www.ftdichip.com/Drivers/VCP.htm>



Download and install the Windows x86 (32-bit) driver. As of September 2013, the latest version is 2.08.30. This is a WHQL Certified driver. The installation of this driver is necessary to allow the correct USB driver to be installed (see below).

3. Using the Windows Device Manager, check that a device called "USB Serial Converter" is available. There will probably be more than one such device shown.
4. Download the utility program called "Zadig" from the Windows Driver Installer library for USB devices Sourceforge website at <http://sourceforge.net/projects/libwdi/files/zadig/>

*Important:* If you are using Windows XP ensure that you download the file `zadig_xp_v2.0.1.160.7z`. For later versions of Windows, download the file `zadig_v2.0.1.160.7z`. These files need to be uncompressed using the 7-Zip utility.

5. Run the Zadig utility, Click on Options->List all Devices, and select a "USB Serial Converter" device from the drop down menu. You should see that the USB ID is 0403 (Vendor ID) and 6010 (Product ID).
6. Select "libusbK (v3.0.5.16)" from the drop down menu of drivers, and press the "Replace Driver" button. Perform this operation twice, for the "USB Serial Converter A" and "USB Serial Converter B" devices.
7. When finished, you should see the change using Windows Device Manager, where it should be listed under section "libusbK USB Devices" as devices "USB Serial Converter A" and "USB Serial Converter B".

## 7 Running the Test Application

Once the system has successfully compiled on either a Windows or Linux OS the test application should be executed to ensure a connection to the NPOINT is completed.

### 7.1 Running the EPICS Database

To run the IOC in either Windows or Linux cd into the iocnPoint directory and execute the st.cmd script. For windows the st.cmd script should be passed into the executable if it is to be run from a batch file. Below is an example of the output generated when the system is started on Linux.

```
#!/../../bin/linux-x86/asynNpointTest
< envPaths
epicsEnvSet("ARCH","linux-x86")
epicsEnvSet("IOC","iocnPoint")
epicsEnvSet("TOP","/home/pbt/nPoint")
epicsEnvSet("ASYN","/home/epics/R3.14.12.2/asyn4-18")
epicsEnvSet("STREAM","/home/epics/R3.14.12.2/stream2-6")
epicsEnvSet("EPICS_BASE","/home/epics/R3.14.12.2/base")
epicsEnvSet("STREAM_PROTOCOL_PATH","../../proto")
epicsEnvSet("EPICS_CA_MAX_ARRAY_BYTES","500000")
cd /home/pbt/nPoint
## Register all support components
dbLoadDatabase("dbd/asynNpointTest.dbd")
asynNpointTest_registerRecordDeviceDriver pdbbase
drvAsynFTDIPortConfigure("LC400", "0x403", "0x6010", "3000000", "2",
"0", "0")
Initialized libftdi 1.0 (major: 1, minor: 0, micro: 0, snapshot ver:
unknown)
FTDI chipid: A5B3D3D1
# When the controller is first powered on, after the control loop
begins
# running the controller still needs up to 2 seconds to initialize
values in memory
epicsThreadSleep 2.0
## Load record instances
dbLoadTemplate("db/nPointTest.substitutions")
# asynSetTraceMask("LC400", -1, 127)
# asynSetTraceIOMask("LC400", -1, 127)
cd /home/pbt/nPoint/iocBoot/iocnPoint
iocInit
Starting iocInit
#####
#####
## EPICS R3.14.12.2 $Date: Mon 2011-12-12 14:09:32 -0600$
## EPICS Base built Jan 23 2013
#####
#####
iocRun: All initialization complete
epics>
```

Once the startup is complete and the records loaded the LC.400 can be controlled by setting various records present in the database. These can be changed by editing the st.cmd file.

## 7.2 Running the EDM Screens

To allow testing some EDM engineering screens have been provided; these can be started on Linux by executing the script `runGUI` present in the top-level directory. The screens could be converted to other formats if required to run on Windows, but this has not been done within the nPoint module. The EDM screens can be run on a Linux host, connecting to an IOC running on Windows.

The EDM screen definition files (.edl) are installed in the data directory. Using the default record name prefix as defined for the test application, the startup command for the screens (issued from the top-level directory) is:

```
% edm -m "P=NPOINT" -x ./data/nPointTop.edl &
```

Once started the user is presented with a top-level startup screen, shown below. The test application contains records for three positioned channels and each is represented on this display. Only the channels physically present are available to use on the display – in the example shown, only channels 1 and 2 are usable, with Channel 3 shown as Unavailable.

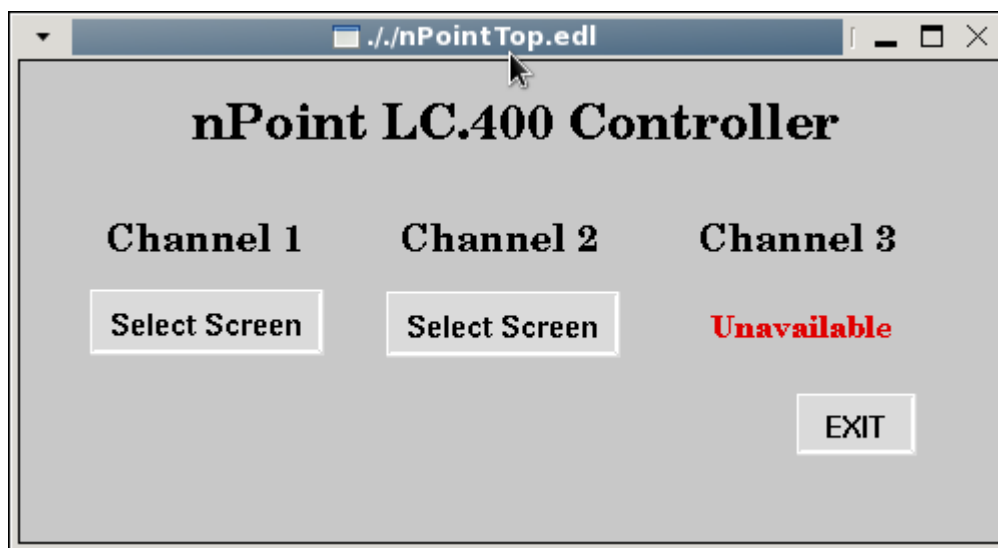


Figure 1 Top-level edl screen

Left-clicking on a Select Screen button for a channel displays a menu with two sub-screens: these are Digital I/O Control and Wavetable Data Control. Click on the menu entry to open the required screen. A similar Digital I/O Control or Wavetable Data Control screen is opened for each channel as required: multiple channel screens may be open simultaneously.

## 7.3 Digital I/O Control Screen

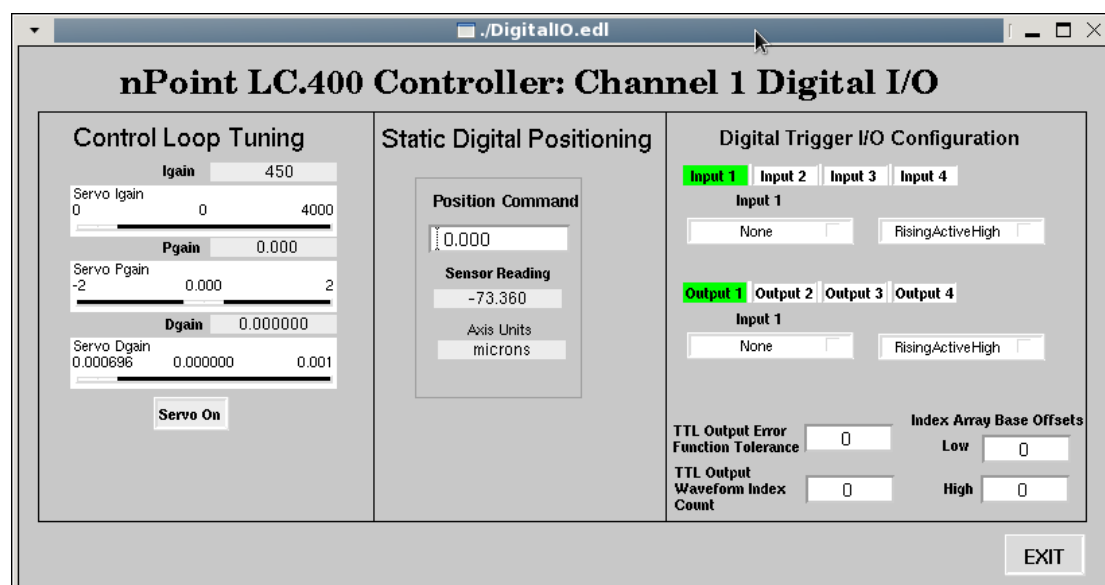


Figure 2 Digital I/O Control screen

The Digital I/O Control Screen provides facilities to display and set the static position directly, adjust the servo PID parameters and configure Digital Trigger I/O.

## 7.4 Servo Loop Control & Tuning

The left-hand part of the Digital I/O Control Screen allows the user to set the servo loop PID parameters and switch the servo loop on or off.

The three sliders allow (from the top) the Integral, Proportional and Derivative Gain of the servo loop to be individually adjusted. The button at the bottom switches the Servo Loop on or off.

## 7.5 Static Digital Positioning

The central part of the Digital I/O Control Screen allows the user to set the static position and monitor the current position reading. The Axis units are displayed at the bottom of this section.

## 7.6 Digital Trigger I/O Configuration

The right-hand part of the Digital I/O Control Screen provides Digital Trigger I/O Configuration. The LC.400 series controllers are equipped with programmable digital I/O capabilities through the 9-pin D-sub connector located on the back panel. Up to four input and four output trigger configurations can be programmed. Refer to Section 3.3 of document [RD3] for a detailed description of Digital I/O configuration.

There are three sections in the Digital Trigger I/O Configuration area on this screen:

- a) **The top section of the Digital Trigger I/O Configuration area** allows one of Inputs (Pins) 1-4 to be selected using the top row buttons (labelled Input 1...Input 4). The left hand pull-down menu specifies the Trigger type for the selected input pin of the TTL I/O connector. It is recommended that the user does not program different functions (other than None) for different channels on the same pin. The Trigger type options available are shown below. The left hand strings (in brackets) are the abbreviated descriptions actually shown on the pull-down menu.

- 0 = None
- 1 = Edge Triggered Start (EdgeStart)
- 2 = Level Triggered Start (LevelStart)
- 3 = Edge Triggered Stop (EdgeStop)
- 4 = Level Triggered Stop (LevelStop)
- 5 = Level Triggered Start and Stop (LevelStartStop)
- 6 = Edge Triggered Pause and Resume (EdgePauseResume)
- 7 = Level Triggered Pause and Resume (LevelPauseResume)

The pull-down menu to the right is used to specify the polarity for the selected input pin of the TTL I/O connector. Note that if the user wants multiple channels to have a function for TTL Input Pin 1 with the same polarity, the polarity must be programmed for each channel individually. The Polarity options are as follows, with the left hand strings (in brackets) being the abbreviated descriptions actually shown on the pull-down menu.

- 0 = Rising Edge/Active High (RisingActiveHigh)
- 1 = Falling Edge/ Active Low (FallingActiveLow)

- b) **The middle section of the Digital Trigger I/O Configuration area** allows one of Outputs (Pins) 1-4 to be selected using the top row buttons (labelled Output 1...Output 4). Outputs 1-4 correspond to connector Pins 6-9.

The left hand pull-down menu specifies the Output type for the selected output pin of the TTL I/O connector. The Output type options available are shown below. The left hand strings (in brackets) are the abbreviated descriptions actually shown on the pull-down menu.

- 0 = None
- 1 = Control Loop Error (ControlLoopErr)
- 2 = Waveform Index Level (WavefIndexLevel)
- 3 = Waveform Index Pulse (WavefIndexPulse)

The pull-down menu to the right is used to specify the polarity for the selected output pin of the TTL I/O connector. The Polarity options are as follows, with the left hand strings (in brackets) being the abbreviated descriptions actually shown on the pull-down menu.

- 0 = Rising Edge/Active High (RisingActiveHigh)
- 1 = Falling Edge/ Active Low (FallingActiveLow)

- c) **The lower section of the Digital Trigger I/O Configuration area** provides input values for the following four integer values:

1. *TTL Output Error Function Tolerance*. Specifies the control loop error threshold for the TTL Output Error function type. The threshold value is a 20 bit number with the same bits per distance scale factor as the Digital Position Command.
2. *TTL Output Waveform Index Count*. Specifies the number of low and high index pairs. For example, if two low indexes are specified and two high indexes are specified, the Index Count value should be 2 (not 4).

3. *TTL Output Low Index Array Base Offset.* The base offset for an array of up to 16 waveform indexes. At the specified indexes, an output set to type Waveform Index Level will transition to level low. At the specified indexes an output function set to type Waveform Index Pulse will output a single pulse.
4. *TTL Output High Index Array Base Offset.* The base offset for an array of up to 16 waveform indexes. At the specified indexes, an output set to type Waveform Index Level will transition to level high. At the specified indexes an output function set to type Waveform Index Pulse will output a single pulse.

## 7.7 Wavetable Data Control Screen

The Wavetable Data Control Screen provides facilities to load data into the Npoint controller to generate a Waveform. The screen provides two methods for loading waveform data into the nPoint controller's Wavetable and controls to configure using and displaying the data.

The maximum Wavetable data buffer size is 83,333 points, corresponding to 2 seconds of data at full loop speed (1 clock cycle every 24  $\mu$ sec). The period can be extended using the Wavetable Cycle Delay value.

At the top of the left-hand part of the screen, two buttons are provided:

- WT Enable: Pressing this button will enable Wavetable scanning for this channel.
- WT Active: Pressing this button sets the software trigger to start the wavetable output (if Wavetable is also Enabled).

Below these buttons there is a menu which selects the source of the Wavetable data:

- Autogenerate. The Wavetable data is generated internally using specified function. The function is chosen using the menu below and may be Sinewave or Triangle.
- Read from file. Wavetable data is read from a text file. The data file contains a list of positions, one position per line, floating-point format in engineering units (usually microns). The file is chosen using a file selector menu (see below).

Below these buttons is a set of controls which varies depending on the data Source selected.

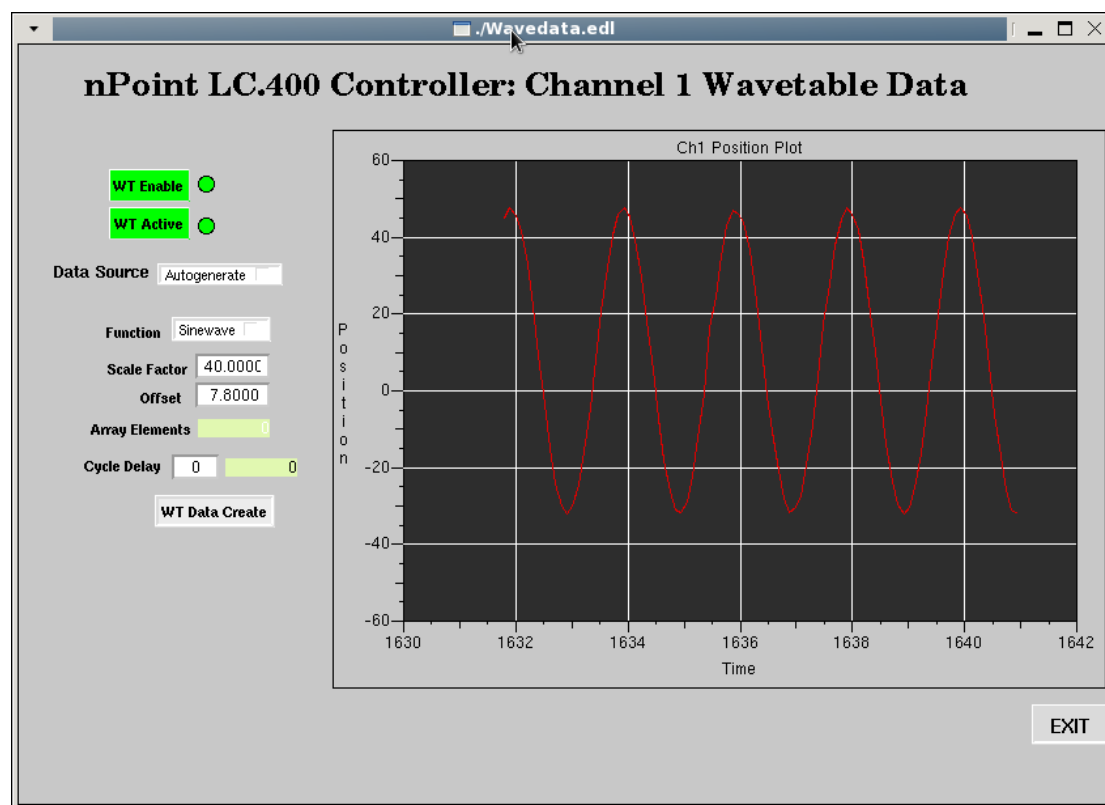


Figure 3 Wavetable screen showing Autogenerate data

- a) **Autogenerate.** See screenshot above. In this mode the following controls and monitors are provided:
- **Function:** Select the Sinewave or Triangle function to generate Wavetable data.
  - **Scale factor:** By default, the generated data is in the range  $+1/-1$ . The amplitude can be multiplied by the value input here (default 1.0000). In the example screenshot shown above, the Scale factor has been set to 40.0.
  - **Offset:** The fixed offset input here is added to the generated data (default 0.0000). In the example screenshot shown above, the Offset has been set to 7.8.
  - **Array Elements:** the number of data elements generated.
  - **Cycle Delay:** The (integer) number of clock cycles to wait before the next wavetable point is output. This can be used to achieve waveform shapes that are longer than two seconds. For example a Wavetable Buffer Size of 83,333 with a Wavetable Cycle Delay of 1 will have a period of 4 seconds. The value can be input on the left and the current value is displayed on the right (default 0).
  - **WT Data Create.** Pressing this button creates the Wavetable data using the selected function.

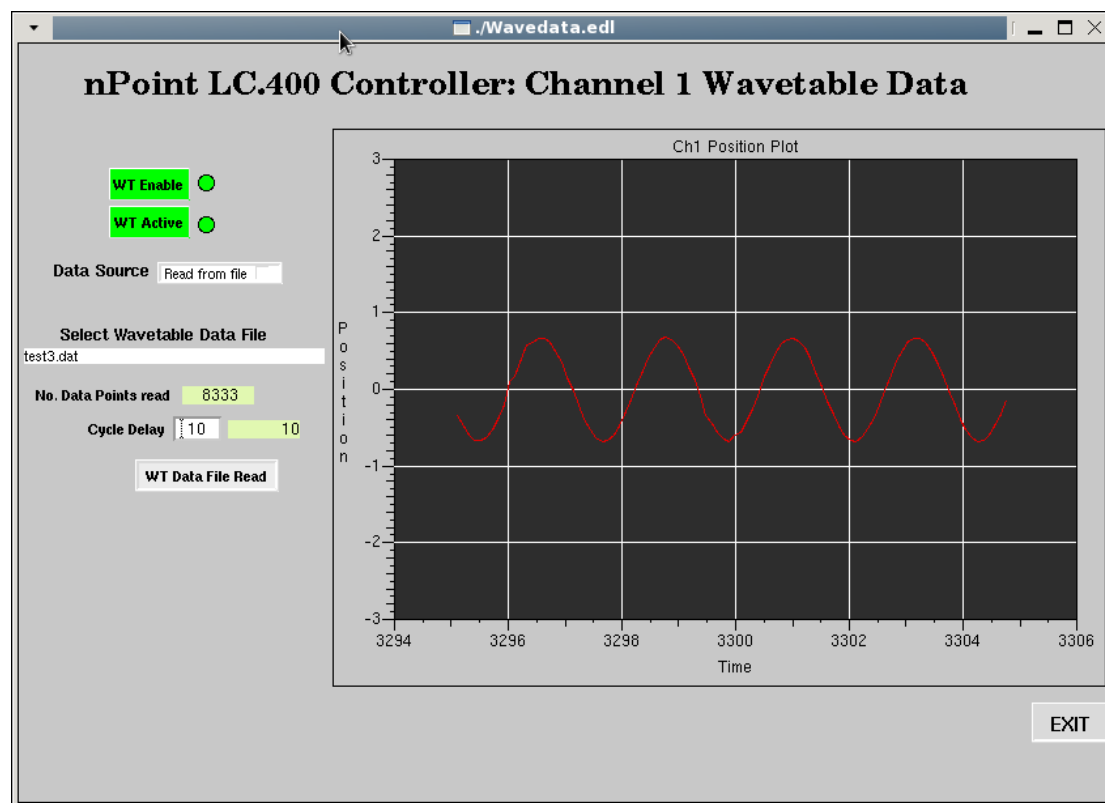


Figure 4 Wavetable screen showing file selection

b) **Read from file.** See screenshot above. In this mode the following controls and monitors are provided:

- **Select Wavetable Data File.** A file selector menu allows navigation through the file system and selection of the required data file.
- **No. Data Points Read:** The total number of data positions read from the file (one position per line).
- **Cycle Delay:** The (integer) number of clock cycles to wait before the next wavetable point is output. This can be used to achieve waveform shapes that are longer than two seconds. For example a Wavetable Buffer Size of 83,333 with a Wavetable Cycle Delay of 1 will have a period of 4 seconds. The value can be input on the left and the current value is displayed on the right (default 0).
- **WT Data File Read.** Pressing this button reads the data from the selected file and loads it into the Wavetable.



## 8 Using the Device Support Code

The simple test application described in section 7 presents all of the possible commands and status items available using the NPOINT motion controller. However, application developers can easily create their own databases of records; this section explains what is required to use the device support for the NPOINT.

### 8.1 Supported EPICS Records

The software allows all commands and status requests to be issued using EPICS records, using two different types of support code:

#### 8.1.1 Single Data Item I/O

Input and output of single (32 bit) data items are supported for a range of EPICS records types by using the EPICS StreamDevice support module. StreamDevice is a generic EPICS device support for devices with a byte-stream communication interface so that they can be controlled by sending and receiving strings. StreamDevice comes with an interface to asynDriver, which implements the low-level support for those communication interfaces. Asyn device support for the NPoint LC.400 controller has been written (using the FTDI direct interface library) to enable the use of StreamDevice software.

The EPICS input record types used for reading status from the controller are longin, mbbi and bi. Output record types used for setting values in the controller are longout, mbbo, and bo.

StreamDevice uses a protocol file which defines commands and expected reply output messages. For this device, only 2 commands are defined: get and put a single 32-bit data value. The format for these messages are defined in reference [RD3], along with those used for data array read/write operations implemented in device support code.

#### 8.1.2 Array Data I/O

Specific device support has been written for reading and writing arrays of wavetable data from and to the nPoint LC.400 using an EPICS waveform record. The corresponding device type (DTYP) strings are “LC400ArrayRead” and “LC400ArrayWrite”. This device support is also used for reading and writing 64-bit floating point data values from and to the controller: these values are handled as an array of two 32-bit data items.

### 8.2 Configuration of EPICS Records and Application

The following records are provided in the template, and can be instantiated using the substitutions file for whichever channel number \$(CHAN) is required. The \$(P) macro is used to define a unique prefix for the EPICS records. A selected set of the most useful records is shown in the following table.

Name	Description
<b>Static Position Control</b>	
\$(P):CH\$(CHAN):GET_POSITION	Get the current sensor position

\$ (P):CH\$(CHAN):SET_POSITION	Set the position. This value is an offset if wavetable data is Enabled.
\$ (P):CH\$(CHAN):GET_RANGE	Get the current positioning range (in engineering units)
\$ (P):CH\$(CHAN):GET_RANGE_TYPE	Get the engineering units e.g. Microns
<b>Servo Loop Control</b>	
\$ (P):CH\$(CHAN):SET_SERVO_PGAIN	Set the servo loop Proportional Gain
\$ (P):CH\$(CHAN):SET_SERVO_IGAIN	Set the servo loop Integral Gain
\$ (P):CH\$(CHAN):SET_SERVO_DGAIN	Set the servo loop Derivative Gain
\$ (P):CH\$(CHAN):GET_SERVO_PGAIN	Get the servo loop Proportional Gain
\$ (P):CH\$(CHAN):GET_SERVO_IGAIN	Get the servo loop Integral Gain
\$ (P):CH\$(CHAN):GET_SERVO_DGAIN	Get the servo loop Derivative Gain
<b>Digital I/O Control</b>	
\$ (P):CH\$(CHAN):TTLINP_PINx	Set the Trigger type for the specified input pin (X=1..4)
\$ (P):CH\$(CHAN):TTLINP_PINx_POLARITY	Set the Polarity for the specified input pin (X=1..4)
\$ (P):CH\$(CHAN):TTLOUT_PINx	Set the Output type for the specified output pin (X=6..9)
\$ (P):CH\$(CHAN):TTLOUT_PINx_POLARITY	Set the Polarity for the specified output pin (X=6..9)
\$ (P):CH\$(CHAN):TTLOUT_ERRTOLERANCE	Set the TTL Output Error Function Tolerance
\$ (P):CH\$(CHAN):TTLOUT_LOWINDEX_BOFF	Set the TTL Output Low Index Array Base Offset
\$ (P):CH\$(CHAN):TTLOUT_HIGHINDEX_BOFF	Set the TTL Output High Index Array Base Offset
\$ (P):CH\$(CHAN):TTLOUT_WAVEF_INDEXCOUNT	Set the TTL Output Waveform Index Count
<b>Wavetable Data Handling</b>	
\$ (P):CH\$(CHAN):WDATAMETHOD_SEL	Define data source (Autogenerate or Read from file)
\$ (P):CH\$(CHAN):SET_WTENABLE	Enable Wavetable data
\$ (P):CH\$(CHAN):SET_WTACTIVE	Set Wavetable data Active
\$ (P):CH\$(CHAN):SET_WTCDELAY	Wavetable Cycle Delay
\$ (P):CH\$(CHAN):WRITE_WTARRAY	Write out data array to the controller. This is an EPICS waveform record using LC400ArrayWrite device support (see section 8.1.2 above). It is used to write both autogenerated data and data read from file, and so is triggered from both the CREATE_WAVEDATA and WAVEDATA_READFILE records (see below)

<b>Wavetable Data Handling: Autogenerate data</b>	
<code>\$(P):CH\$(CHAN):WDATAFUNC_SEL</code>	Select data generation function type (Sine/Triangle)
<code>\$(P):CH\$(CHAN):CREATE_WAVEDATA</code>	<p>Wavetable data creation record. This record calls the C subroutine createPeriodicData using the following fields</p> <p><b>Inputs:</b></p> <p>A = Number of output elements (LONG)</p> <p>B = Type of data to be generated: 0 = Sine, 1 = Triangle (LONG)</p> <p>C = Scaling factor (DOUBLE)</p> <p>D = Value Offset (DOUBLE)</p> <p>E = Full motion range (DOUBLE)</p> <p><b>Outputs:</b></p> <p>VALA = Array of values (LONG ARRAY)</p>
<b>Wavetable Data Handling: Read data from file</b>	
<code>\$(P):CH\$(CHAN):WAVEDATA_READFILE</code>	<p>Wavetable file data reading record. This record calls the C subroutine readWtdataFile using the following fields:</p> <p><b>Inputs:</b></p> <p>A = filename (CHAR ARRAY)</p> <p>B = Full motion range (for scaling) (DOUBLE)</p> <p><b>Outputs:</b></p> <p>VALA = Array of values (LONG ARRAY)</p> <p>VALB = Number of values (LONG)</p> <p>VALC = Error string (CHAR ARRAY)</p>