




Type: Programmer's
Title: GuideProgrammer's Guide

Page: 1 / 17
Version: 1.0
Date: July 27, 2011

FUZZY LOGIC SERVO CONTROLLER

viTestApp User Guide


| | | |
|---|--------------------------------|---------------------|
|  | Type: Programmer's | Page: 2 / 17 |
| | Title: GuideProgrammer's Guide | Version: 1.0 |
| | | Date: July 27, 2011 |

UPDATES TABLE

| Version Number | Date | Nature of modifications | Author |
|-----------------------|---------------|--------------------------------|---------------|
| 1.0 | July 21, 2011 | First draft | rob |

TABLE OF CONTENTS

| | | |
|-----------|-------------------------------------|----------|
| 1. | <u>TEST SOFTWARE</u> | 4 |
| 1.1 | DESCRIPTION | 4 |
| 1.2 | INSTALLATION | 4 |
| 2. | <u>OPERATION</u> | 5 |
| 2.1 | GATEWAY AND SERVO NETWORK DETECTION | 5 |
| 2.2 | GATEWAY TAB | 5 |
| 2.3 | SERVO TAB | 5 |
| 2.3.1 | SERVO CONTROL TAB | 6 |
| 2.3.1.1 | Hardware Settings | 6 |
| 2.3.1.2 | Motion Settings | 6 |
| 2.3.1.3 | Motion Start/Stop | 6 |
| 2.3.1.4 | Limit Control | 7 |
| 2.3.1.5 | Zero | 7 |
| 2.3.1.6 | Status | 8 |
| 2.3.1.7 | EEPROM Configuration | 8 |
| 2.3.1.8 | Network Suspend | 8 |
| 2.3.2 | FUZZY TUNING TAB | 8 |
| 2.3.3 | STATISTICS TAB | 8 |
| 3. | <u>TUNING</u> | 9 |
| 3.1 | FUZZY LOGIC CONCEPTS | 9 |
| 3.2 | HARDWARE AND SERVO RATE SELECTION | 10 |
| 3.3 | FUZZY CONTROLS | 12 |
| 3.4 | EXAMPLE TUNING PROCESS | 14 |

| | | |
|---|--------------------------------|---------------------|
|  | Type: Programmer's | Page: 4 / 17 |
| | Title: GuideProgrammer's Guide | Version: 1.0 |
| | | Date: July 27, 2011 |

1. TEST SOFTWARE

1.1 Description

viTestApp is a .NET application developed by Vera Ikona for Windows users to test our fuzzy logic servo control (FLSC) modules via the USB to CAN Gateway (UCG). The program exercises all features of the module including the ability to visually inspect servo feedback to facilitate tuning the fuzzy controls.


The application source code is available on GitHub. The code can be used as a guide for those who wish to create their own custom Windows applications using the viServoMaster dll provided with the installer software.

1.2 Installation

System software requirements for the application are a PC with a Windows OS running .NET framework 3.5.

Before the user's system can access the UCG the user must download and install the D2XX driver package from FTDI (<http://www.ftdichip.com/Drivers/D2XX.htm>). Take note of the driver installation path. When the UCG is plugged into the USB bus for the first time the operating system will ask for the location of this driver. If you have multiple UCG devices the operating system will repeat the process for each device.

The installer program (viTestAppInstaller.exe) copies the viTestApp.exe file to the directory path of the user's choice. The viServoMaster.dll and FTD2XX_NET.dll files are also copied to the same directory. If the user selects the default location the files will be installed to "C:\Program Files\viTestApp". The installer will create the directory if it doesn't exist and prompt the user to start the program or quit the installation after writing the files.

| | | |
|---|--------------------------------|---------------------|
|  | Type: Programmer's | Page: 5 / 17 |
| | Title: GuideProgrammer's Guide | Version: 1.0 |
| | | Date: July 27, 2011 |

2. OPERATION

2.1 Gateway and Servo Network Detection

If the FTDI driver is properly installed and a UCG device is attached to the USB bus then the viTestApp application will detect the UCG serial number and present it for selection when the user clicks on the “Open Servo Network” menu item. Select the device and click on the “Open FTDI Device” button to begin detection of the servo controller network. If the selector in the startup form doesn't display a UCG serial number then unplug and replug the USB cable to the device and restart the application. Network detection requires several seconds to poll the 127 possible network addresses. The servo controls are not accessible if no FLSC devices are detected.

2.2 Gateway Tab

The Gateway tab page shows the software version of the UCG and a “Status” group of controls for monitoring the CAN bus status and setting the CAN bit rate for the network. The textbox at the bottom is a CAN message turnaround timer. It shows the time that has elapsed from when the last master command was placed on the CAN bus until the slave response message returned.


The default CAN bitrate of the UCG and all FLSC modules is 125Kbps. Selecting another bitrate starts a process of synchronizing the network to the new bitrate. If after establishing a faster bitrate a FLSC module gets reset or the UCG unit is reset by unplugging the USB cable there will be a mismatch of CAN bitrates. CAN errors that cause a “bus off” condition within the FLSC module will cause it's LED diagnostic code to flash and it's bitrate to change to 125Kbps. In the case where CAN bitrates become unsynchronized it is the responsibility of the master software (viTestApp) to reset it's CAN bitrate to the default value.

2.3 Servo Tab

If only one FLSC is detected on the network its address and version number will be displayed in the “Slave List” selector on the top right of the “Servo” tab. If the network has more than one FLSC module the user will need to explicitly select an address from the “Slave List” to enable the servo controls.

The Servo tab contains three sub-tabs:

- Servo Control – All servo controls except the fuzzy controls
- Fuzzy Tuning – Controls for rules, input membership functions and output singleton
- Statistics – contains the visual feedback elements for servo tuning

| | | |
|---|--------------------------------|---------------------|
|  | Type: Programmer's | Page: 6 / 17 |
| | Title: GuideProgrammer's Guide | Version: 1.0 |
| | | Date: July 27, 2011 |

2.3.1 Servo Control Tab

The functionality of the controls within this tab is grouped into 8 categories:

2.3.1.1 Hardware Settings

The controls inside the 'Servo Control' group box set the servo rate (frequency of servo cycles) and encoder divider which are fundamental to fuzzy logic operation as well as the interface to various h-bridges.

The 'Move Done' radio buttons allow the user to control whether the h-bridge stays energized after completing a 'Position' or 'Trapezoidal' move.

The 'Kick' control provides a way to improve final positioning for a 'Trapezoidal' move, especially when the fuzzy controls have not been optimally tuned.

The GPIO controls affect pin 16 of the FLSC module. This pin can be used for either input or output. In the case of output, the pin is useful for switching external hardware and has the ability to toggle on network suspension. Typically this suspend toggle feature can control a MOSFET gate to the incremental encoder so that power consumption is minimized.

Changes to hardware settings are saved to RAM in the active FLSC when the local 'Set' button is pressed. The current hardware settings stored in the active FLSC RAM are obtained by pressing the 'Get' button.


2.3.1.2 Motion Settings

Motion trajectory is controlled by the 'Motion Control' group box. Position is a signed 32-bit variable which means that the valid range for the target position is -2147483648 to 2147483647 encoder counts. Velocity is a signed 20-bit variable allowing values to span the -524288 to 524287 encoder counts per second range. Acceleration is an unsigned 32-bit variable (encoder counts per second second) however its range is also constrained by position and velocity selections that cause an acceleration period less than 1 servo tick or greater than 65536 servo ticks. The PWM controls cause the h-bridge to function at a set pwm rate and direction without the use of any encoder feedback information.

The 'Motion Control' group box 'Set' button saves the trajectory values to the active FLSC module RAM. The 'Get' button will load the motion trajectory controls with the values stored in the active FLSC module RAM. The 'Go' checkbox, when checked, will cause the motion to begin when the local 'Set' button is pressed (motion type set by radio buttons in 'Start Motion' group box).

2.3.1.3 Motion Start/Stop

The 'Stop Motion' group box will stop the servo in two ways. The soft method is a de-energizing of the h-bridge causing the servo to coast to a halt via mechanical resistance. The hard stop is actually a command to servo to the current position.

| | | |
|---|--------------------------------|---------------------|
|  | Type: Programmer's | Page: 7 / 17 |
| | Title: GuideProgrammer's Guide | Version: 1.0 |
| | | Date: July 27, 2011 |

The 'Start Motion' group box contains a set of radio buttons to select the motion type and a 'Start' button to begin the motion trajectory set in the 'Motion Control' group box (assuming the 'Go' button was not selected when those settings were saved to the FLSC module). The 'pos limited' version of the PWM motion type is a way to test motor response within a mechanical system with finite travel limits. Motions using this mode must travel towards the position set in the 'Motion Control' group box and will stop at that value according to the stop type set in the 'Stop Motion' group box. The other buttons inside this group box are canned procedures to demonstrate the capability of the FLSC module to modify trajectory values while in motion. The 'Inv+St' button reads the target position currently stored in the FLSC module, inverts the value, writes it back and begins the motion type set by the radio buttons. Similarly the 'Dbl+St' and 'Half+St' buttons double or half the current position value stored in the FLSC module. If the checkbox inside the 'Group' group box is set then the 'Start Motion' buttons will initiate moves on all FLSC modules. The group motion start command will have no affect on modules that have already attained their goal positions if the motion type is position or trapezoidal.

The 'Random Test' uses a pseudo random number generator to modify the current position, velocity and acceleration values stored in the active FLSC (group mode is disabled) and begin 100 moves that vary in duration from .5 to 2 seconds.

2.3.1.4 Limit Control

This group box contains all possible limit sensing for the FLSC module. For each of the external limit switches and for the encoder index signal there are 4 controls. The 'Enable' checkbox is self explanatory; the 'Zero' checkbox will reset the encoder count to zero (only one of the three can be the zero limit); the 'Value' control determines whether a rising or falling edge triggers the limit; the 'Halt' type is how the servo will stop when it encounters the limit. Note that external pull resistors are should be used for external limits to prevent the inputs of the HC14 gates on the FLSC module from floating.


The 'ATD Limit' is for h-bridges that provide current feedback as a voltage that ranges from 0 to 5V. Setting the 'Limit Value' to zero disables the limit, otherwise the value corresponds to 255 discrete steps between 0 and 5V. The 'Dwell' setting is the duration in servo cycles that the current feedback must be equal or above the 'Limit Value' before de-energizing the h-bridge (no hard stop is available for this limit).

The 'IRQ Stop' is typically used as a deadman stop that is common to multiple FLSC modules. A low voltage level on this common line will cause all connected FLSC modules with their 'IRQ Stop' enabled to stop according to the associated stop type radio buttons.

The local 'Set' button transfers all the limit settings to the active FLSC module RAM and the 'Get' button copies all the limit settings from the active FLSC module RAM to the UI.

2.3.1.5 Zero

The 'Zero Encoder' button sets the current encoder count of the active FLSC module to zero. Note that the position stored by the Motion Control command is zeroed to prevent starting a motion that might cause a limit error.

| | | |
|---|--------------------------------|---------------------|
|  | Type: Programmer's | Page: 8 / 17 |
| | Title: GuideProgrammer's Guide | Version: 1.0 |
| | | Date: July 27, 2011 |

2.3.1.6 Status

This area is used to query the active FLSC for current state and error information. Usage of the 'Current Velocity', 'Peak Encoder Change' and 'Peak Change Change' textboxes are discussed in the fuzzy tuning section. The 'Servo State' textbox shows the current motion phase (ACCEL, SLEW, DECEL, REVERSE or DONE). The 'Limit' section displays the physical pin state for the external limits and the encoder index as well as a checkbox to indicate if that particular limit has been detected. The 'ATD Value' textbox displays the current voltage detected on the current sense input if a non-zero value is present in the 'Limit Value' textbox of the 'ATD Limit' section.

Errors are 'sticky' meaning that they can only be cleared by clicking on the 'Clear Error Flags' button.

2.3.1.7 EEPROM Configuration

Selecting the 'Write All' button will store 4 files in the active FLSC file system. These files are the 'Servo Control' settings, the 'Limit Control' settings, the fuzzy logic input membership functions and output singleton, and the fuzzy logic rules. Motion controls are not saved. Upon power up of the FLSC module, these saved settings will override the default settings. The 'Read All' button will reset the hardware, limit and fuzzy settings to the saved values. The 'Delete All' button will erase all stored files in the active FLSC module so that the factory default values will be restored on the next power up.

2.3.1.8 Network Suspend

This is a network wide command to place all FLSC modules into a sleep mode for power saving. Regardless of the wakeup method selected, encoder and limit events are not monitored, thus there is no guarantee that the FLSC module position will be where it was before entering the suspended state. As stated above in the hardware settings, the GPIO pin can be configured to toggle on entry/exit from the suspend state.

2.3.2 Fuzzy Tuning Tab

The controls in this tab are discussed in the 'Tuning' section.

2.3.3 Statistics Tab

The statistics tool is described in the 'Tuning' section.

3. TUNING

3.1 Fuzzy Logic Concepts

The “Fuzzy Logic Support“ chapter of the “S12XCPUV1” reference document from Freescale contains the details of the fuzzy logic internals of the module. The term “fuzzy logic” does not in any way imply that system inputs and outputs are uncertain or non-deterministic. The “fuzziness” of the system is the way in which crisp inputs are applied to a set of rules that are evaluated for the degree of their truthfulness. The rules themselves are formulated as linguistic expressions that non-experts can easily understand, such as, “if the current position is negative large (relative to the goal position) and the speed is zero then apply positive large energy”. Each rule has three components:

1. if clause (position error input)
2. and clause (speed input)
3. then result (output)

The 25 rules inside the servo module correspond to 5 possible “if” input expressions compared to 5 possible “and” input expressions (5x5) with 5 possible “then” output expressions.

| | | | | | |
|----------------|---------------|--------------|---------------|----------------|--------------|
| IF POSITION IS | NEGLRG | AND SPEED IS | NEGFST | THEN OUTPUT IS | POSHI |
| IF POSITION IS | NEGLRG | AND SPEED IS | NEGSLW | THEN OUTPUT IS | POSHI |
| IF POSITION IS | NEGLRG | AND SPEED IS | SAME | THEN OUTPUT IS | POSHI |
| IF POSITION IS | NEGLRG | AND SPEED IS | POSSLW | THEN OUTPUT IS | POSHI |
| IF POSITION IS | NEGLRG | AND SPEED IS | POSFST | THEN OUTPUT IS | POSLO |
| IF POSITION IS | NEGSML | AND SPEED IS | NEGFST | THEN OUTPUT IS | POSHI |
| IF POSITION IS | NEGSML | AND SPEED IS | NEGSLW | THEN OUTPUT IS | POSHI |
| IF POSITION IS | NEGSML | AND SPEED IS | SAME | THEN OUTPUT IS | POSLO |
| IF POSITION IS | NEGSML | AND SPEED IS | POSSLW | THEN OUTPUT IS | ZERO |
| IF POSITION IS | NEGSML | AND SPEED IS | POSFST | THEN OUTPUT IS | NEGLO |
| IF POSITION IS | OK | AND SPEED IS | NEGFST | THEN OUTPUT IS | POSHI |
| IF POSITION IS | OK | AND SPEED IS | NEGSLW | THEN OUTPUT IS | POSLO |
| IF POSITION IS | OK | AND SPEED IS | SAME | THEN OUTPUT IS | ZERO |
| IF POSITION IS | OK | AND SPEED IS | POSSLW | THEN OUTPUT IS | NEGLO |
| IF POSITION IS | OK | AND SPEED IS | POSFST | THEN OUTPUT IS | NEGHI |
| IF POSITION IS | POSSML | AND SPEED IS | NEGFST | THEN OUTPUT IS | POSLO |
| IF POSITION IS | POSSML | AND SPEED IS | NEGSLW | THEN OUTPUT IS | ZERO |
| IF POSITION IS | POSSML | AND SPEED IS | SAME | THEN OUTPUT IS | NEGLO |
| IF POSITION IS | POSSML | AND SPEED IS | POSSLW | THEN OUTPUT IS | NEGHI |
| IF POSITION IS | POSSML | AND SPEED IS | POSFST | THEN OUTPUT IS | NEGHI |
| IF POSITION IS | POSLRG | AND SPEED IS | NEGFST | THEN OUTPUT IS | NEGLO |
| IF POSITION IS | POSLRG | AND SPEED IS | NEGSLW | THEN OUTPUT IS | NEGHI |
| IF POSITION IS | POSLRG | AND SPEED IS | SAME | THEN OUTPUT IS | NEGHI |
| IF POSITION IS | POSLRG | AND SPEED IS | POSSLW | THEN OUTPUT IS | NEGHI |
| IF POSITION IS | POSLRG | AND SPEED IS | POSFST | THEN OUTPUT IS | NEGHI |

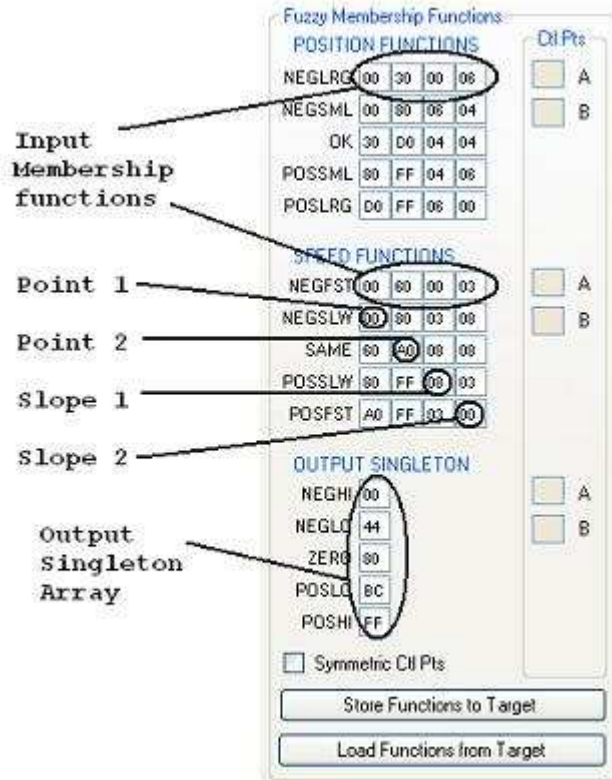
A servo iteration applies crisp input values derived from the encoder feedback to all 25 rules using min-max fuzzification to build a table of results which are defuzzified by a weighted average to generate a crisp output. The output is transformed to a PWM signal for various h-bridge circuits. All rules are considered equally true.

Each of the ten input expressions are described by a trapezoidal membership function that consists of two points and two slopes as described in the Freescale reference document. The 4 bytes of each trapezoidal function are in the following order:

1. point 1
2. point 2
3. slope 1
4. slope 2

The outputs correspond to a 5 byte array of singletons.

Inputs are signed 8-bit values and are thus restricted to the range of 127 to -128. Inputs outside of this range are clipped to the maximum 127 value or the minimum -128 value.



Fuzzy Membership Functions

POSITION FUNCTIONS

| | | | | |
|--------|----|----|----|----|
| NEGLRG | 00 | 30 | 00 | 06 |
| NEGSML | 00 | 80 | 06 | 04 |
| OK | 30 | D0 | 04 | 04 |
| POSSML | 80 | FF | 04 | 06 |
| POSLRG | D0 | FF | 06 | 00 |

SPEED FUNCTIONS

| | | | | |
|--------|----|----|----|----|
| NEGST | 00 | 60 | 00 | 03 |
| NEGSLW | 00 | 80 | 03 | 08 |
| SAME | 60 | 40 | 08 | 08 |
| POSSLW | 80 | FF | 08 | 03 |
| POSFST | A0 | FF | 03 | 00 |

OUTPUT SINGLETON

| | |
|-------|----|
| NEGH | 00 |
| NEGL | 44 |
| ZERO | 80 |
| POSLL | BC |
| POSH | FF |

Di Pts: A B


Symmetric Di Pts

Store Functions to Target

Load Functions from Target

3.2 Hardware and Servo Rate Selection

If the motor/encoder combination is not already set then the system designer must select an appropriate servo motor and power supply that can achieve the desired motion performance while remaining within the specifications of the h-bridge. Motor selection

| | | |
|---|--------------------------------|---------------------|
|  | Type: Programmer's | Page: 11 / 17 |
| | Title: GuideProgrammer's Guide | Version: 1.0 |
| | | Date: July 27, 2011 |

may end up being an iterative process if the user doesn't have precise understanding of the kinematic properties of the mechanical system. The choice of encoder is determined by the position and velocity accuracy required by the system. Note that gearing or transmission components will amplify the accuracy of an encoder mounted on the motor shaft. Note also that the optical incremental encoder might be rated in cycles per revolution which is actually $\frac{1}{4}$ of the total resolution because quadrature optical encoder signals generate 4 events per cycle.

An important consideration when selecting a motor/encoder combination is to restrict the rate of encoder feedback to the maximum .5 million (2^{19}) encoder counts per second that the servo controller is capable of reading. For example, a motor shaft mounted 1250 cpr encoder produces 5000 events per revolution which sets the upper limit of motor speed to 104.8 revolutions per second (6291 rpm). Controlling the motor speed greater than 6291 rpm would necessitate using an encoder with a lower resolution.

The servo rate is the number of times that the fuzzy inference engine is executed per second and ranges from 16Hz to 4096Hz. The most critical factor to enable proper tuning of the fuzzy logic servo controller is to select a servo rate that keeps the position and speed inputs within the signed 8-bit range during "trapezoidal" or "velocity" modes of motion. This means that from one servo cycle to the next the distance travelled must be less than 128 encoder events.

The servo rate can be derived by calculation or by experimentation. If the specifications of the system are known then a simple calculation can determine the minimum servo rate. As an example, if the motor is required to rotate at 80 rev/sec with a 512 cpr encoder then the encoder rate will be $80 * 512 * 4 = 163840$ encoder counts per second. If the servo rate is set to 2048 (2KHz) the encoder feedback rate will be 80 events per servo cycle. A servo rate less than 2KHz will overflow the signed 8-bit fuzzy input range unless the encoder divider is employed to reduce the feedback rate. If a divider rate of 4 is used then the encoder rate becomes 40960 encoder counts per second and a 512Hz servo rate is usable.

The simplest method to empirically derive the servo rate is to run "pwm" mode experiments. If the motor is required to operate near the maximum speed then set the pwm duty cycle to 100%, select a direction, start motion and observe the "Peak Encoder Change" feedback of status command reads. If the value exceeds 128 then the servo rate must be increased or the encoder divider increased. If the peak encoder change is a very small number at the top speed then the servo rate should be decreased or the encoder divider decreased.

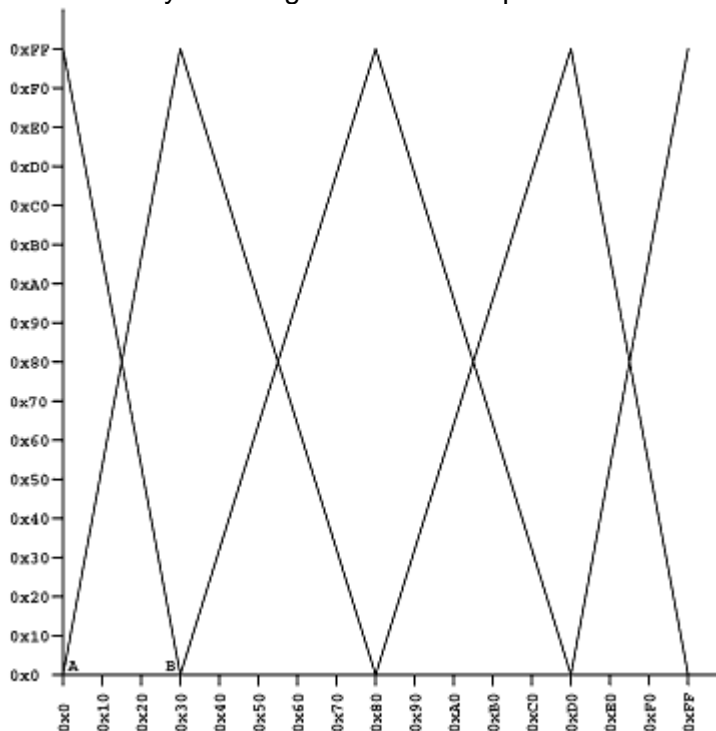
If the arrangement of the mechanical system limits travel such that the motor cannot freely rotate past a certain position then the "pos limited" version of the pwm motion mode can be used. Alternatively, "position" mode commands can be issued for short distances while observing the peak encoder change. The disadvantage of using "position" mode is the chicken-egg scenario where the fuzzy controls will affect motor performance before we get a chance to set the basic hardware configuration required in advance to setting the fuzzy controls. This method is satisfactory if the user has a basic understanding of the fuzzy controls.

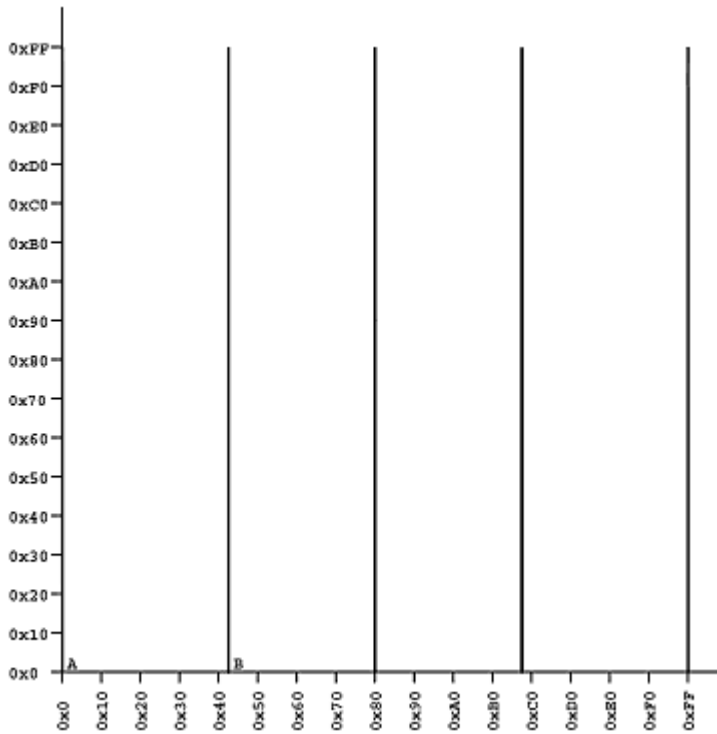
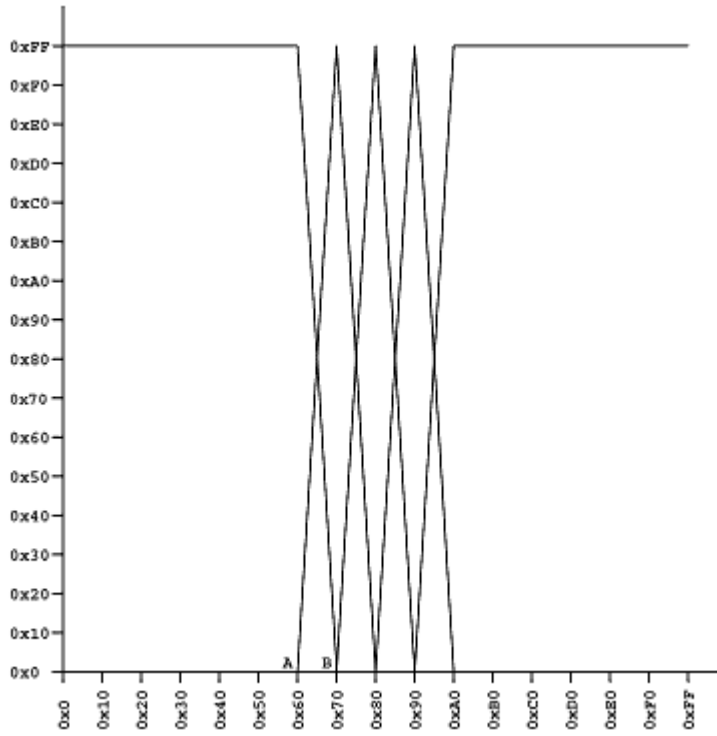
Note also that during the pwm test the “Current Velocity” feedback is the actual speed of the system (encoder counts per second). The “Peak Change Change” quantity is the acceleration when multiplied by the servo rate squared (ie: a value of 4 when the servo rate is 1KHz translates to $4 * 1024 * 1024 = 4194304$ encoder counts per secondsecond).

3.3 Fuzzy Controls


The default fuzzy control settings that come with a new control module are an acceptable starting point for fuzzy tuning. They may only require minor modification for systems with a low inertial load relative to motor capacity and with a servo rate that allows the encoder feedback to utilize the majority of the signed 8-bit range. Though the fuzzy rules are unlikely to need adjustment the input membership functions are likely to change via experimentation with the visual feedback provided by the stats command.

The quickest method of modifying the input membership functions is to check the “Symmetric Ctl Pts” checkbox and use the A and B control points. These controls have input filters to enforce properly constructed trapezoidal functions. 2D visual representations of the input membership functions with A:B values 0:30 and 60:70 and output singleton array with values 0:44 are depicted by the following three illustrations. Note that checking this control may destroy settings in the UI; the user can only undo this action by restoring the membership functions from the FLSC memory.





In the case of the input trapezoidal functions there may be small flats at the tops of the triangular shapes due to the integer (non-fractional) quantity of slopes 1 and 2.

| | | |
|---|--|--|
|  | Type: Programmer's Title: GuideProgrammer's Guide | Page: 14 / 17 Version: 1.0 Date: July 27, 2011 |
|---|--|--|

If a vertical line representing the crisp input is drawn on a graph of input membership functions, the y-axis intersection value (or non-intersection) with a trapezoidal function represents the degree of truth for the corresponding linguistic expression. In the case of the second graph, a crisp input of 0x68 would be 50% true for the expression described by the leftmost input function, 50% true for the expression represented by the next function to the right of it and 0% true for the other 3 expressions. Min-max rule evaluation of the inputs and weighted average defuzzification is detailed in the “Fuzzy Logic Support” chapter of the CPU12 reference manual and won’t be repeated here.

Unchecking the “Symmetric Ctl Pts” checkbox allows users to input asymmetric values into the membership functions. This permits the motor to operate differently in one direction than the other for asymmetric loads.

Input membership function and output singleton array values are always validated by the controller before being accepted. It is not possible to store membership functions and singletons that are improperly constructed.

3.4 Example Tuning Process

Tuning the servo is an iterative process of adjusting the input membership functions or output singleton control points, commanding a short distance motion, reading back the fuzzy statistics and comparing the generated graph with a previous set of statistics. The screen shots in this section illustrate the iterative process of tuning a servo using the membership functions and output singletons generated by the symmetric control points. Since the size of the statistics buffer in the controller is only 1000 points, data sets of a complete trapezoidal motion including the acceleration, slew and deceleration phases will be possible only for short distances, especially if the servo rate is high.

The DC servo used for the test is a Maxon A-max 32 powered by a 300mA 12 volt DC power supply in combination with the LMD18200 IO board from Vera Ikona. A 10:1 spur gearhead combined with a 500cpr encoder mounted on the motor shaft results in 20000 encoder counts per revolution of the output shaft.

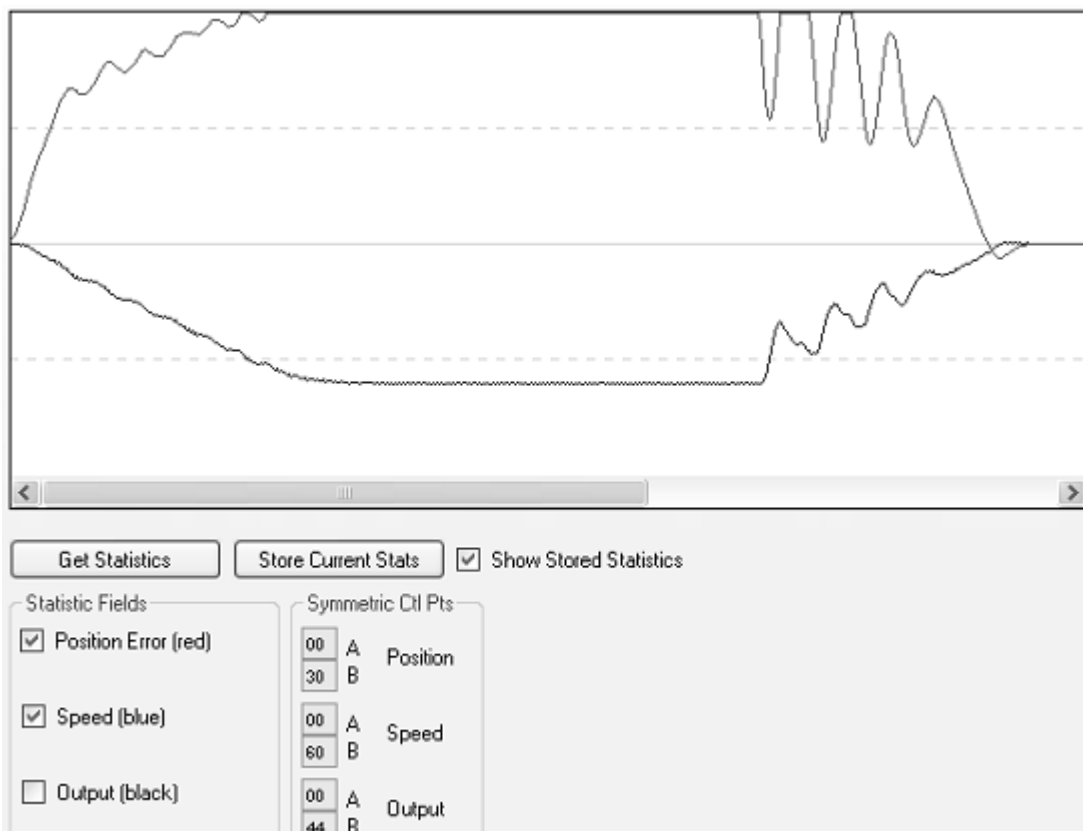
For this particular test the output shaft position is not limited by an attached mechanism so one is free to run pwm tests to determine the optimal servo rate for the maximum speed achievable by the system. Note that pwm motion will halt and return an error if the polarity of the encoder feedback is opposite to the commanded pwm direction (encoder value must increase for the INC direction). If there is a polarity error the user must swap the motor connections to the h-bridge or swap the encoder channel connections to the IO board. It is strongly suggested that motor/encoder polarity be determined with the motor unattached to a mechanical system.

In the case where a mechanism limits travel one can use the “pos limited” version of the pwm command to de-energize (soft stop) the motor when it has achieved the position set in the “Position” box of the “Motion Control” area. Note that a soft stop means that the motor will stop via the resistance of the mechanism combined with the internal resistance of the motor, so be careful to not set the limit too close to the end of travel.

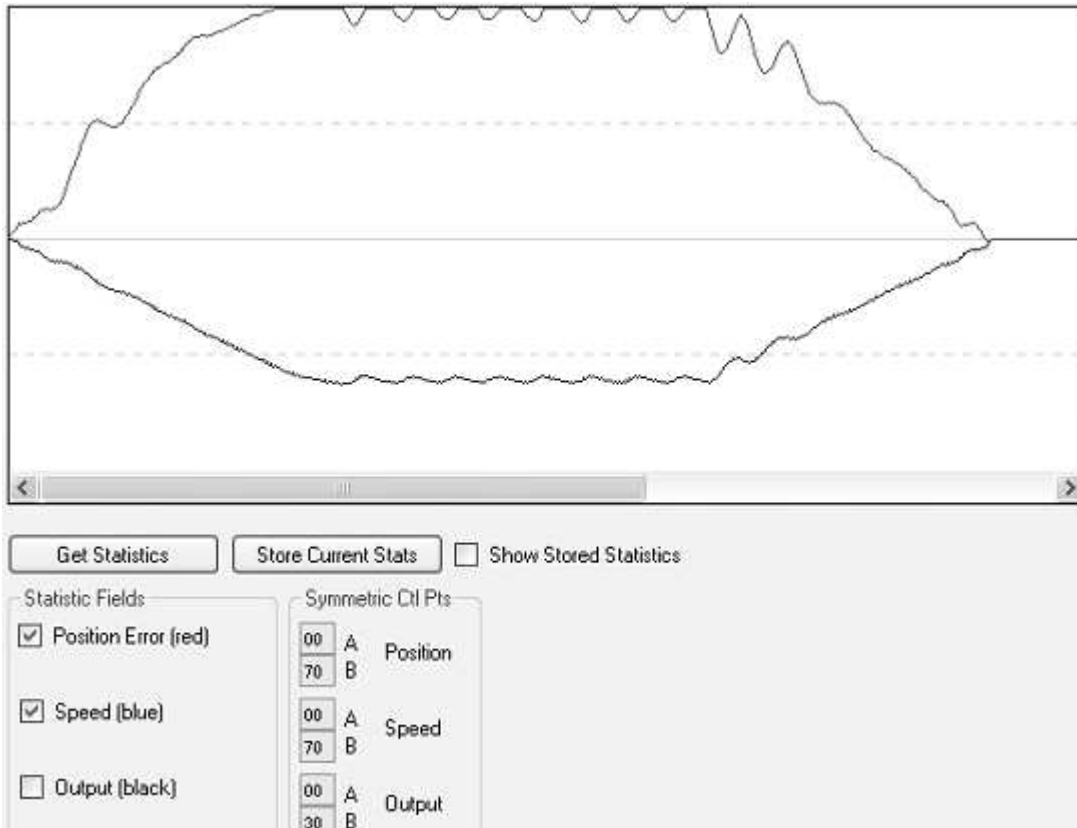
The “pos limited” pwm mode will return an error if the commanded pwm motion causes the motor to move in the wrong direction away from the limit position.

Pwm tests with the duty cycle set to 100% show that a “Peak Encoder Change” of 106 encoder counts per servo cycle resulting from a servo rate of 1KHz. A rate slower than this will overflow the signed 8-bit range unless an encoder divider value greater than 1 is used. Higher frequency servo rates are usable but we will start with the 1KHz rate. The “Current Velocity” reading of 108544 encoder counts per second ($106 * 1024$) is the absolute maximum speed for this motor/power supply combination. Tuning for this speed isn't a good idea since there is no load on the motor other than the gearhead. Arbitrarily I shall choose a speed of 75% of maximum (80000 encoder counts per second) as the top speed required for the tests. The “Peak Change Change” reading from the 100% pwm test is 5 giving a theoretical maximum acceleration of $5 * 1024 * 1024$ or 5242880 encoder counts per secondsecond. Again this is not a realistic value to tune towards so I shall arbitrarily choose an acceleration of 500000 encoder counts per secondsecond. According to the formula $t=v/a$ these settings will accelerate the motor to the final velocity in .16 seconds. We won't have an issue fitting a test distance of 20000 encoder counts into the statistics buffer with this servo rate/speed/acceleration.

Starting with the default symmetric control points (no fuzzy controls stored) the statistics generated by a trapezoidal move from -15000 to 15000 look promising (using the “Inv+St” is a quick and easy way to repeat a trapezoidal motion).

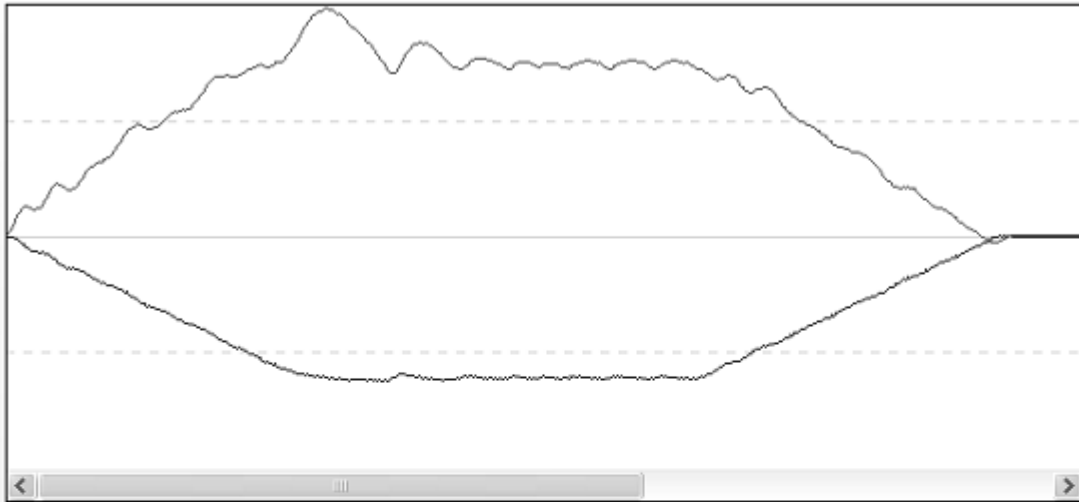


The saturation of the position error (red) indicates that the motor is not keeping up with the commanded position and needs more energy to be applied at the output. The blue line that indicates the change in position (speed) should be made as smooth as possible. Moving the input B control points more to the centre of the signed range will increase the probability for higher energy motor output. Also setting the output B control point closer to zero will increase motor energy for short distance motions.



This result is an improvement of the deceleration phase of motion but the position error is still partly saturated during the slow phase. Getting better performance from the system is best achieved by making adjustments to a single input or output at a time, repeating the motion and collecting a new set of statistics to compare with the previous set. Pressing the “Store” button writes the current set of statistics to memory so that it can be restored by checking the “Show Stored Statistics” checkbox. Checking and unchecking the control will alternately display the previous or current set of statistics and control points for visual comparison.

The following graph was achieved after about a dozen iterations of adjusting control points, repeating the motion and comparing the plots of statistics:



Show Stored Statistics

Statistic Fields

- Position Error (red)
- Speed (blue)
- Output (black)

Symmetric Cut Pts

| | | |
|---------------------------------|---|----------|
| <input type="text" value="20"/> | A | Position |
| <input type="text" value="60"/> | B | |
| <input type="text" value="14"/> | A | Speed |
| <input type="text" value="54"/> | B | |
| <input type="text" value="00"/> | A | Output |
| <input type="text" value="36"/> | B | |