Servo 8 Torque Board Doc V 1.4

RS-232 hobby servo controller with torque feedback
No servo modifications required

Features:

- Eight independent 8-bit servo control outputs allow 254 positions for each servo.
- Standard 90 degree or extended 180 degree rotation range.
- Servos can be turned on or off.
- Eight independent 16-bit servo torque input values using standard hobby servos.
- Eight independent servo home positions.
- Daisy chain connector allows independent control of up to 8 boards from a single RS-232 serial line.
- ServoGUI included. Free source code.
- CPU power range is 5.5 to 9 VDC.
- Servo power range is 5 to 9 VDC (depending on your servos)
- CPU and servos can share power, or they can use two separate power inputs for increased isolation.

Method of operation

The torque board measures torque by analyzing the servo. Here's how it works -- when you send a command to a servo to move to a certain position, the servo needs to figure out how far it should move. It does this by taking the difference between its current position and commanded position. This difference is called the error.

The servo's job in life is to keep the error as close to zero as possible. It does this by applying power to its internal motor in such a way as to move the output arm in the direction that minimizes the error.

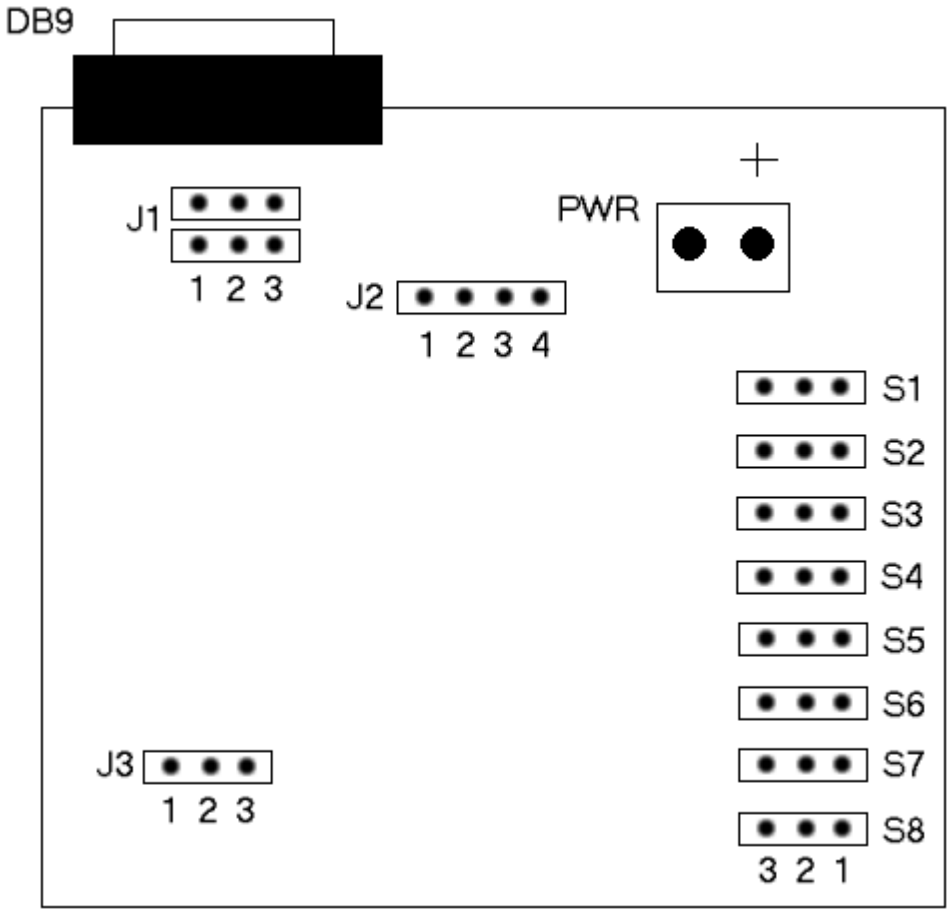
Consider the case where the servo is holding a constant position, and an external torque is applied to the output arm. If you assume an ideal servo with a frictionless gear train, the torque will cause the output arm to move slightly, and the absolute value of the error will increase to a nonzero value. In attempting to drive the error to zero, the servo will apply power to the motor to oppose the torque. The CPU on the board analyzes this condition.

In a real servo, the gear train will have a certain amount of friction. The friction needs to be taken into consideration, since the torque being measured is actually at the motor shaft rather than the output arm.

The presence of friction can sometimes be used to advantage, since it can cause hysteresis when the servo movement changes direction. The servo board actually reports the absolute value of torque, not the torque itself, which means it is possible to infer the torque sign by comparing the torque readings when approaching a position from two different directions.

Servos generally need to be calibrated if you want to be able to quantitatively relate the actual torque to the numerical torque values reported by the board. For a typical calibration, you would apply a known torque to the servo, as measured in units of Newton-meters or ounce-inches, then read the value reported by the servo board. Calibration curves will generally vary, depending on the design and size of each servo.

Servo 8 Torque Board Configuration



Connectors and Jumpers:

DB-9: 9-pin female RS-232 connector.

PWR: Servo power, +5 to 9 VDC. Also supplies CPU power if shared power option is used.

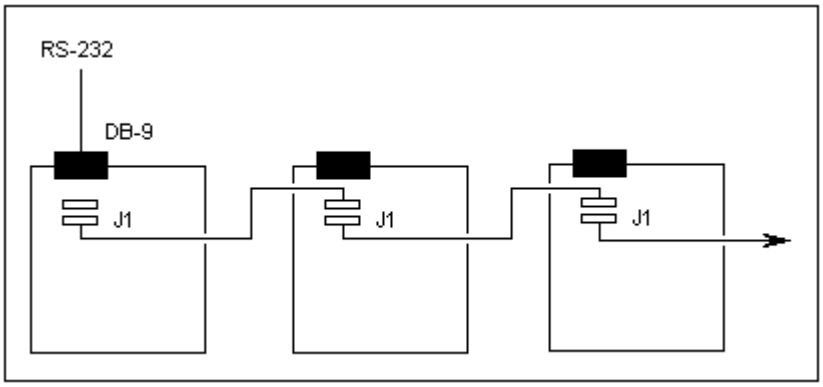
J1: RS-232 daisy chain connector. Default is open.

J2: Configures CPU power.

J3: Allows changes to module address and baud rate. Default is open.

S1-S8: Servo connectors. Pin 1 is signal, pin 2 is positive power, pin 3 is ground.

Jumper	Position	Purpose
J2	1 to 2	CPU shares power with servos (default)
	Open	Pin1 Pin 2 is +5 to 9 VDC CPU power, pin 3 is ground
J3	Open	Write-protect module address and baud rate (default)
	1 to 2	Restore factory settings for module address & baud rate
	2 to 3	Allow changes to module address and baud rate



DAISY CHAIN CONFIGURATION

A daisy chain allows you to use a single serial line to control up to 8 modules. You can chain modules together by connecting three in-row pins on jumper J1 to the corresponding pins on adjacent modules. J1 is internally connected to the DB-9 as follows:

J1 Pin	DB-9 Pin	Function
1	5	Ground
3	3	Transmit data
2	2	Receive data

Note: The daisy chain connector (J1) connects directly to the DB-9 and is not TTL level.

COMMAND FORMAT

Each command is a sequence of bytes sent to the servo. The length of the sequence is four or five bytes, depending on the command.

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Header	Module Address	Servo Address	Command	Data (optional)

All commands:

Header: “>” (ASCII 62)

Module Address: “1” .. “8” (ASCII 49 .. 56)

Servo Address: “1” .. “8” (ASCII 49 .. 56)

Send multiple position commands:

Command: “m” (ASCII 109)

Data: 8 separate bytes representing servo position 1 - 8 (binary)

Turn on servo and set position:

Command: “a” (ASCII 97)

Data: 1 .. 255 (binary) position. Turns on pulse train to servo.

Turn off servo:

Command: “a” (ASCII 97)

Data: 0 (binary). Turns off pulse train to servo.

Set current position as home:

Command: "c" (ASCII 99)

Data: (not used)

Go to home position:

Command: "h" (ASCII 104)

Data: (not used)

Set width resolution

Command: "w" (ASCII 119)

Data: 0 (ASCII 0) = standard
1 (ASCII 1) = extended

Get position

Command: "g" (ASCII 103)

Data: (not used)

Get torque

Command: "t" (ASCII 116)

Data: (not used)

Set module address

Command: "s" (ASCII 115)

Data: "1" .. "8" (ASCII 49..56) new address

Remarks: Jumper J3 pins 2 and 3 must be connected for this command.

Set baud rate

Command: "b" (ASCII 98)

Data: "0" (ASCII 48) = 19200 baud
"1" (ASCII 49) = 9600 baud
"2" (ASCII 50) = 4800 baud
"3" (ASCII 51) = 2400 baud

Remarks: Jumper J3 pins 2 and 3 must be connected for this command.

Example command:

Byte #	1	2	3	4	5
ASCII	“>”	“1”	“2”	“a”	--
Binary	62	49	50	97	255

This command goes to module #1, turns on servo #2 and sets the servo position to 255, which is one endpoint of the servo travel range.

RESPONSES TO COMMANDS

With two exceptions, the servo board responds to all commands with a carriage return character (Dec 13). The exceptions are as follows:

(1) The response to the “get position” command is a 1-byte binary position, followed by carriage return.

(2) The response to the “get torque” command is a 2-byte binary torque value, followed by carriage return. The first and second data bytes form the lower and upper byte of a 16-bit unsigned integer torque value. The precise relationship between measured torque and actual torque depends on many things, such as the type of servo.

Note: In processors with buffered serial ports such as the BasicX, you should periodically clear the serial input buffer to prevent the accumulation of the servo board’s (Dec 13) responses from overflowing the serial port buffer.

CHANGING BOARD ADDRESS AND BAUD RATE

Procedure for changing the board address or baud rate:

1. Turn on power.
2. On jumper J3, connect pins 2 and 3.
3. Send the “set module address” or “set baud rate” software command.
4. Turn off power.
5. Move jumper J3 to the center-off position (no pins connected).
6. Turn on power.

The board is factory-set to board address 1 and 19200 baud rate. If you want to restore the factory settings, use the following procedure:

1. On jumper J3, connect pins 1 and 2. Power can be on or off.
2. Turn on power if it’s not already on.
3. Turn off power.
4. Move jumper J3 to the center-off position (no pins connected).

SERVO CONTROL SIGNAL BASICS

Each servo is controlled by sending a train of pulses over the signal wire, which is pin 1 of the servo connectors. The pulse rate varies somewhat, but is a minimum of 50 Hz.

The servo position is controlled by the pulse width. In standard mode, the pulse width varies between 1 ms and 2 ms, with the center at 1.5 ms. In extended mode, the pulse width varies between 0.48 ms and 2.52 ms, with a center again at 1.5 ms.

When you send a command to turn off the servo, the pulse train is halted, although power continues to be applied to the internal servo electronics. Absence of a pulse train usually means that the servo motor is left unpowered, and the servo will not try to maintain any position – in other words, the servo is free to move.

Servo 8 Torque Board FAQ

Frequently Asked Questions

1. What size power supply do I need?
You will need 1 Amp of current for each connected servo.

2. How do I verify that the board is working without a computer attached?
Turn off power to the board. Plug a servo into connector S1, turn the servo arm away from 90 degrees and then apply power. The servo should center to 90 degrees, which indicates the board is controlling the servo.

3. What are the default settings?
The factory default settings are board address 1, servo home position set to 127 (center position) and the baud rate is 19200,N,8,1.

4. How do I reset the board to its default settings?
On jumper J3, connect pins 1 and 2. Turn power on if it's not already on. Turn off power. Move jumper J3 to the center-off position (no pins connected).

5. How do I change the board baud rate?
Turn on power. On jumper J3, connect pins 2 and 3. Send the "set baud rate" software command (you can use the ServoGUI for this). Turn off power. Move jumper J3 to the center-off position (no pins connected). Turn on power. The board is now set to your selected baud rate.

6. How do I change the board address?
Turn on power. On jumper J3, connect pins 2 and 3. Send the "set module address" software command (you can use the ServoGUI for this). Turn off power. Move jumper J3 to the center-off position (no pins connected). Turn on power. The board is now set to your selected baud rate.

7. Why does the LED image change to red when using the ServoGUI?
When using shared power it is possible that the servos can cause a voltage drop that will power down the board, which needs a solid 5.2-VDC level. This is why we suggest 6 VDC @ 1Amp minimum when using the shared power jumper.

8. How do I use separate power?
Remove the jumper on J2 pins 1 and 2, which makes it possible to put up to 9 VDC on J2 pin 2 and Gnd on Pin 3, The servo PWR connector should never exceed 9 VDC. In this configuration it's possible to use a 9 VDC battery on J2 and four D-cell type batteries on the servo PWR connector.

9. How do I disable a servo?
When you send the "turn off servo" command (which is equivalent to sending a servo position command of zero) this turns the pulse train off for the selected servo. This minimizes drain on the batteries as well.

BasicX Example Programs

BasicX-24 Example Program 1:

```
‘ Connections: BX-24 DB-9 to Servo8T DB-9 using null modem cable
‘ ************************************************************
‘ This Program moves Servo 1 through each of it’s 255 positions.
‘ ************************************************************
Sub Main()

Dim Servo_Position As Byte
Dim I As Byte

Do

   For I = 1 to 255

      Servo_Position = I
      Debug.Print “>” & “1” & “1” & “a” & Chr(Servo_Position)
      Call Sleep(25)
   Next

Loop

End Sub
```

BasicX-24 Example Program 2:

```
‘ Connections: BX-24 DB-9 to Servo8T DB-9 using null modem cable
‘ ************************************************************
‘ This Program uses the “m” command to simultaneously move Servos
‘ 1-8 through each of their 255 positions.
‘ ************************************************************
Sub Main()

Dim I As Byte

Do

   For I = 0 to 255

      Debug.Print ">" & "1" & "1" & "m" & Chr(I) & Chr(I) & _
      Chr(I) & Chr(I) & Chr(I) & Chr(I) & Chr(I) & Chr(I)
      Call Sleep(25)
   Next

Loop

End Sub
```

BasicX-24 Example Program 3:

```
' Connections: BX-24 Com1 to PC serial port. BX-24 Pin12 to
' Servo8T J1(Pin3), BX-24 Pin11 to Servo8T J1(Pin2), BX-24
' Pin4 (Gnd) to Servo8T J1(Pin3)
'
' This program updates Servo1 with the position 127 (center)
' and uses debug.print to display it's torque value on the PC.
Dim Com3_In(1 to 30) As Byte
Dim Com3_Out(1 to 30) As Byte
Dim Torque_Display As Integer

' ************************************************************
Sub Main()

Call OpenQueue(Com3_In, 30)
Call OpenQueue(Com3_Out, 30)
Call DefineCom3(11, 12, bx1000_1000)
Call OpenCom(3, 19200, Com3_In, Com3_Out)

Do

   ' Move servo1 to center postion
   Call PutQueueStr(Com3_Out, ">" & "1" & "1" & "a" & Chr(127))
   Call Sleep(15)
   ' Remove the servo boards (Dec 13) Responce form the serial buffer
   Call ClearQueue(Com3_In)
   ' Get the torque value for Servo 1
   Torque_Display = Read_Torque(1)
   ' Print the value out com1 @ 19,200
   Debug.Print Cstr(Torque_Display)

Loop

End Sub

' ************************************************************************
Function Read_Torque(ByVal Servo As Byte) As Integer

Dim Torque_Bytes As Integer
Dim Data As Byte

' Make sure the serial input queue is clean
Call Sleep(15)
Call ClearQueue(Com3_In)
' Request the torque reading
Call PutQueueStr(Com3_Out, ">" & "1" & Cstr(Servo) & "t")
' Wait for Servo 8T to respond and put first 2 bytes in
' var "Torque_Bytes".
Call GetQueue(Com3_In, Torque_Bytes, 2)
Call GetQueue(Com3_In, Data, 1)

' Check the 3rd byte and make sure it was Dec 13
If Data = 13 Then
   ' Read was good return the data
   Read_Torque = Torque_Bytes
Else
   ' Read was bad return 0
   Read_Torque = 0
End If

End Function
```

NetMedia, Inc.

10940 N. Stallard Place
Tucson, Arizona 85737

Tel: 520-544-4567
Fax: 520-544-0800

E-mail: Sales@Netmedia.com