ToolkitRC ST8 servo tester

Table of Contents

Peter Scott © 2020

Last edit 24 July 2020

I bought this just-released device from AliExpress for 46 USD. It took 17 calendar (not ‘working’) days to arrive from China. It is well made and fairly robust but I think it would be sensible to keep it in its very solid box if taking it to the field.

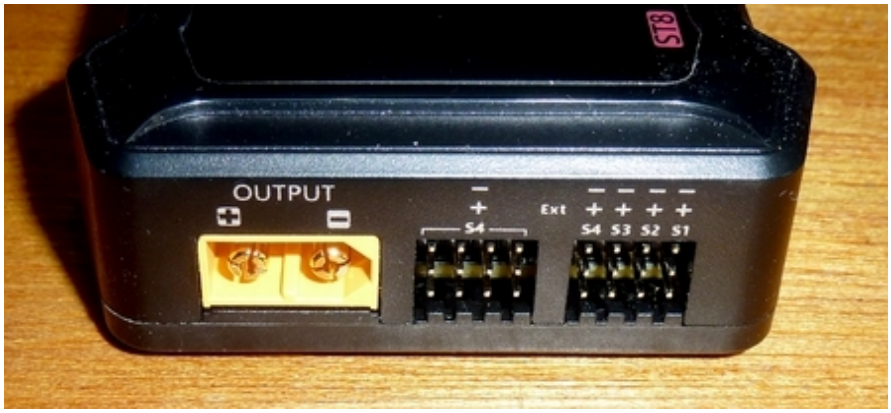
The manual in the box is useless. It is important to download the full manual from the company’s website at https://www.toolkitrc.com/st8. The full one isn’t wonderful but is better. I used to tell my adult students that a manual (computer in that case) is like sex. When it’s good it’s wonderful, but when it’s bad it’s better than nothing. Click **Downloads** to find the latest manual (V1.0 at the time of writing). This is part one of the review. It covers the basic and most important functions. In part two I will cover the more advanced functions.

What it can do

- Test all servos up to 8.4 V and 2 A as standard, and others at higher voltages and currents using a special lead.
- Measure the current drawn by servos under different conditions.
- Find the safe movement for a servo.
- Match servos for critical situations.
- Set a central point on a servo exactly.
- Find out the signal lengths required for a particular maximum and minimum deflection.
- Measure the time a servo takes to reach maximum deflection.
- Test servos at the field, using a standard XT60 lipo flight battery.
- Run a long test on a possibly faulty servo.
- Take receiver signals from a standard pulse width modulation (PWM) channel or from S.BUS or PPM.

The tester can handle up to eight servos. There are four separately controllable channels **S1** to **S4** and four more that are just paralleled with **S4**. It can also take external PWM, PPM and S.BUS signals from receivers and other devices such as an arduino.

There are three controls. The first, called **P1**, is a silver rotating knob used to operate the servos that are plugged in. The second is labelled **OK**. This has a central button used to select or open something and a part you turn to move between options or to change a value. The third is labelled **EXIT** which speaks for itself.



Left end

Right end

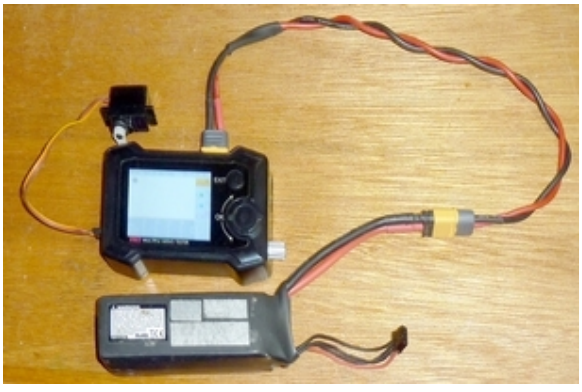
More extreme servos

A high torque, digital or coreless servo might take more than 2 A. If so, you must power it from the XT60 OUTPUT port and make up a special lead. However no picture or specification is given for this lead, so I had to guess what was needed. Higher voltage servos can be tested, up to 28 V.

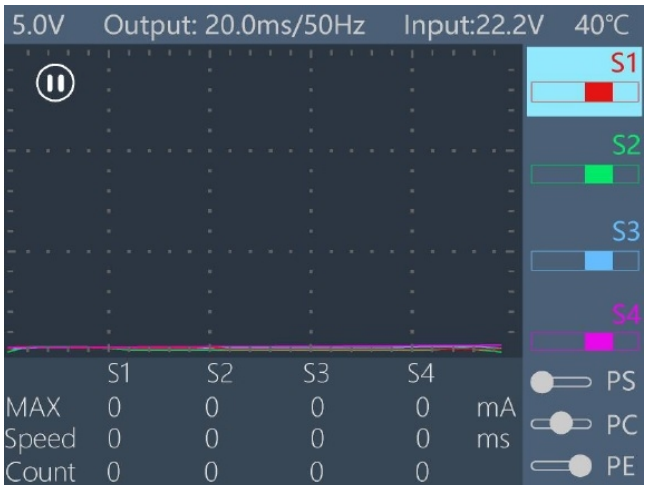
Time to play

The first thing I did was to connect the ST8 to my computer using a micro USB lead. My computer recognised the ST8 but I wasn't ready to update the software so left that for the time being. There is always the danger of 'bricking' the device if you don't know exactly what you are doing. As it is so new there is unlikely to be an update.

I connected a 3S lipo to the **INPUT** socket. The screen lit up and the tester beeped. You need to find or make an XT60 extension lead for the battery, as shown below, or it is awkward to pick up the tester.

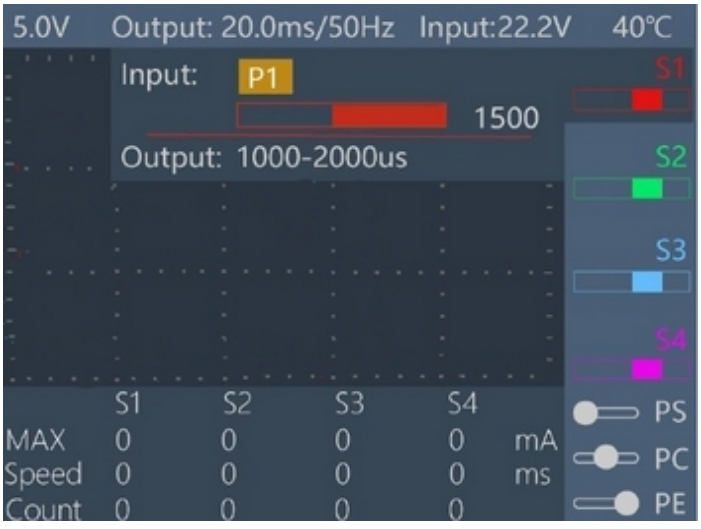


Pressing **EXIT** moved to an oscilloscope type screen. The internal noise signal displayed at the bottom. Down the right hand side of the screen are the four servo channels S1 to S4. Each is colour coded. Each connects to one of the JR-style channel sockets on the side.



Then I plugged an old servo into S1. Turning the **P1** knob on the side made the servo move, and the PWM signals being sent to it displayed as a red, vertical bar chart rapidly moving across the screen. The height showed the current drawn. Slow movement produced spaced out bars and rapid ones made them closer packed and taller. The current for S1 was shown at the bottom of the screen, as MAX mA.

I pressed **OK** and got a screen similar to this. This showed that the input signal being sent to the servo was coming from P1. It also showed the length of the PWM pulse currently being sent and the maximum and minimum values. Note that microseconds are shown on the screen in the simpler to display **us** unit format rather than the more correct **µs**.



Along the top of my screen it said:
5.0V Out: 20.0ms/50Hz Input:12.1V 36ºC
So it defaults to 5 V outputs and the standard PWM signal cycle time.

By burning out a servo, I had discovered a while back that cheap testers output the same voltage as you power them with. Not this one. The output voltage can be changed as you will see later.

As it appeared safe, I then connected four different 9 g micro servos into channels S1 to S4.
Each of these channels has a different colour.
When the pulses are displayed on the screen they have the same different colours.
The screen becomes a simple oscilloscope. Not a very useful one, as you see later.

Turning **P1**, made the servos move.
The red bar showed the PWM pulse length in µs.
At the bottom, the display showed the current draw, which I found surprisingly high for 9 g servos at up to 1.6 A. The faster I turned the knob and the faster the servos moved, the higher the current. Gentle movements such as you use in normal flying showed lower readings.

Here are the data from the four analogue and digital servos. The last value stays on the screen for a couple of seconds after you stop moving P1.

			Fast mA	Normal mA
S1	Hobby King	HK15178 (analogue)	950	200
S2	Tower Pro	MG90S (digital)	1000	600
S3	Corona	CS-929MG (digital)	1600	600
S4	Tower Pro	SG90 (analogue)	500	150

It is important to know what maximum current the servo draws under normal use at high speed, and stalled, perhaps caused by a stuck control surface. You can then decide if you need to use a power box to avoid the currents overloading the battery eliminator circuit (BEC) or the receiver.

Changing the signal

I then wondered what the large knob labelled **OK** was for. I decided it was now wise to remove all but one disposable servo.
I pressed **OK**.

I turned **OK** and found that I scrolled around the **Input** and the **Output** PWM signal timings.

There were two boxes under **Output**, one for the low pulse and the other for the high. This allows us to set the servo range of movement.

I scrolled to the 1000 µs box and pressed **OK**.
By turning **OK** I changed it to 1300 µs.
I pressed **EXIT** and scrolled to high and changed it to1700 µs. As you would expect, the servo movement was a lot less when I turned the P1 knob.

With P1 fully turned clockwise I then increased the maximum pulse to 2200 µs. The servo of course moved further but didn't buzz. This would be a good way to check the maximum safe range of servo travel.

Increasing the signal to 2400 µs gave nearly 90º deflection but the servo started buzzing so I stopped there and went back to 1000 to 2000.

Having got so far using the classic suck it and see principle, I then needed to RTFM. In other words Read The Friendly Manual. At least I think that's what the F means. I continued to pl… er investigate.

Setting up the servo output channels

You can control the servo(s) under test using P1. You can also put signals into S5 on the right from a receiver, or other sources such as an Arduino. These can be PWM, PPM or S.BUS. There are built-in (internal) signal sources for testing as well. You can select which source goes to which output channel. Each channel may be set up totally differently.

The first thing to do is select which servo channel, S1 to S4, to set up.
Let's start with S1.

Start from scratch by restarting the tester.
Press **EXIT**.
S1 should be selected. If not turn **OK** until it is.
Press **OK** to select the **Input/Output** panel.
Press **OK**.
P1 is already selected.
Press **OK** again and the characters P1 are highlighted for edit.
Turn **OK** and you scroll round to:

- **Key** to use values from buttons PS/PC/PE (part 2).
- **Internal.** You can scroll to options for Linear and Stage used for soak testing (next).
- **S5** which allows you select PWM/ PPM and channel/ SBUS and channel for the S5 port (part 2).

Press **EXIT** to accept the value and leave the setup.

S1 has four options available: P1, Key, Internal, S5.
S2, S3 and S4 have an additional option - to be the same as S1.
The four channels to the left of S4 are parallels to, and set the same as, S4.

System setup

Hold down the **OK** button until you enter the Setup screen. There are nine things to change of which probably five are of interest:

- **VoltageOutput**: This defaults to OFF but can be set to a voltage higher than 5 V and must be set if using the XT60 main port for high current servos.
- **CycleCount**: This is for soak testing and is 5000 by default. You can change it.
- **Lowest Input**: This determines what voltage the supply battery can go down to before the tester switches off. Set it according to the safe minimum for the battery you are using, for example 11.3 V for a 3S lipo.
- **Safe temperature**: This is used when the main port is used. It switches off the tester when the temperature gets too high. It defaults to 70°C but can be changed.
- **CycleCountClear**: This sets the CycleCount back to zero.

Soak testing

This is the nerdish name for running a device or component continuously, and possibly under stress, for an extended time to see if it works properly or fails. It is particularly useful for checking old, suspect or crashed servos.

Set the **Input** to **Internal**.
It is probably in **Linear** mode.
The servo moves continuously and the Count at the bottom of the screen goes up by one for each cycle.
Press **OK** and scroll to **Stage**.
The servo now jumps from one extreme to the other, again being counted.
To leave an option choice press **EXIT**.
You could leave it running for hours but that would be more than a lifetime's flying so hardly a worthwhile test. You can set the number that the testing stops at. This was described under **System setup**.
To set the count back to zero for a new test hold down **OK** and select **CycleCountClear.**

Move back to **Linear**.
Turn **OK**.
The number next to **STEP** is highlighted.
You can now set the length of time of each step.
Turn **OK** in steps up to 10 and watch how the movement speeds up.
Press **OK** and turn **OK** to get to **SPEED.**
Again press **OK** and turn **OK** to change the value.
As you move up to 10 you see the speed go down.
Press **EXIT** to quit.
Clearly if you want to hammer a servo under test setting STEP high will do it.
Don't ask me what the numbers mean. I just know what happens. Maybe I'll cover it in part two.

Tests on a range of servos

I then tested several servos, large and small, at high speed, for current draw in mA. Compare these data with those for the 9 g servos listed above.

Turnigy TGY-778MG	700	(slim wing servo)
AeroStar ASI-621MG (coreless)	2050	
Tower Pro MG958	3500	
Turnigy 555MG	1050	
Futaba S3003	950	(ancient analogue servo)
Corona DS-238MG	1300	
Turnigy TGY-4409MD	2200	

I was cautious when testing the large servos in case I blew up the tester. It seems that you can safely do very brief tests at currents higher than the specified 2000 mA maximum on the S1 to S4 ports. The tester didn't get warm. In any case its temperature is displayed on the screen and it switches off if it rises too high. I think it wise only to connect one large servo at a time. Doing it this way is at your own risk of course. I certainly wouldn't do a soak test. I'd use the Output power XT60 port which will be described in part two.

Current draw is a reasonable guide when matching servos, though timing might be more important for some 3D flyers.

Using the screen

In the manual you see pictures of the screen with enlarged signal traces. I have failed completely to find out how to do that. The manual doesn't mention it and I have given up trying to do it by pla… experimenting.

PS/PC/PE positions

When you select KEY as the INPUT you can then scroll using OK to the the three values stored in PS, PC and PE. These can be used to measure the time a servo takes to move from one position to the next. Normally they would be left as 1000, 1500 and 2000 µs as these values give full 60º deflections.

Set **KEY** as the Input for S1 and set S2 to S4 to use the settings for S1.
Press **EXIT** to get the main screen.
Turn **OK** to move down to PS.
Press **OK**.
The servo jumps to a the extreme low position (1000 us).
Turn **OK** to move down to PC.
Press **OK**.
The servo jumps to the central position.
The time it took in milliseconds (ms) is displayed under Speed at the bottom.
Move to PE and test again.
You can jump between the **KEY** positions and see the times displayed.

I did this with the original four servos and found, unsurprisingly, that they were different. What was a surprise was that the times varied for each servo without any apparent pattern. The servo with the least variation was the cheapest one. The two digital (D) ones were worse than the analogue (A) ones. Variations were (%) 32, 95, 79, 68 (A, D, D, A). Perhaps it is to do with the order in which the PWM pulses were sent to the servos.

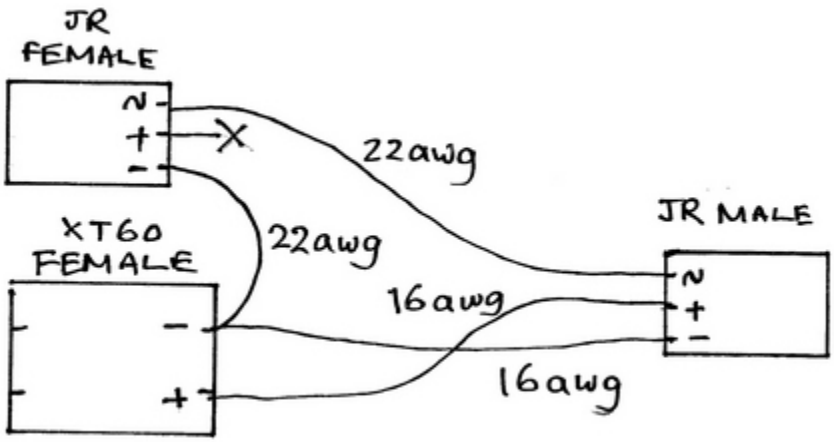
Then I tested the fastest servo I had, a coreless Aerostar ASI-621MG. I hoped it wouldn't blow up the tester. The speed averaged 0.16 s for a 60º swing, which was very close to the specified speed of 0.152 s at 4.8 V. The variation was way better at 8%. Current was about 2.6 A.

My final test was to see how much the same model of servo varied. I used four brand new Tower Pro SG90 9 g ones. One didn't work at all, which was worth knowing. It went in the bin. The others showed quite a range of variation, perhaps not surprising as they are cheap. But it does show the wisdom of matching up pairs of similar servos.

High power servos

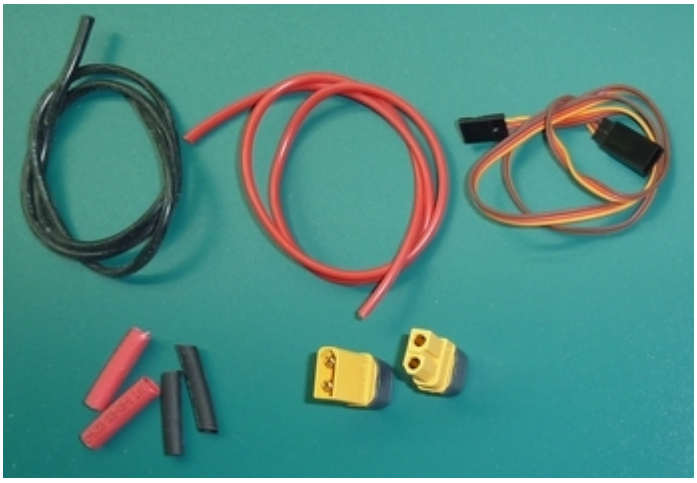
For high power servos try a slow deflection first. If it is clear that more than 2000 mA are needed then make up a lead.

Wiring diagram for the lead

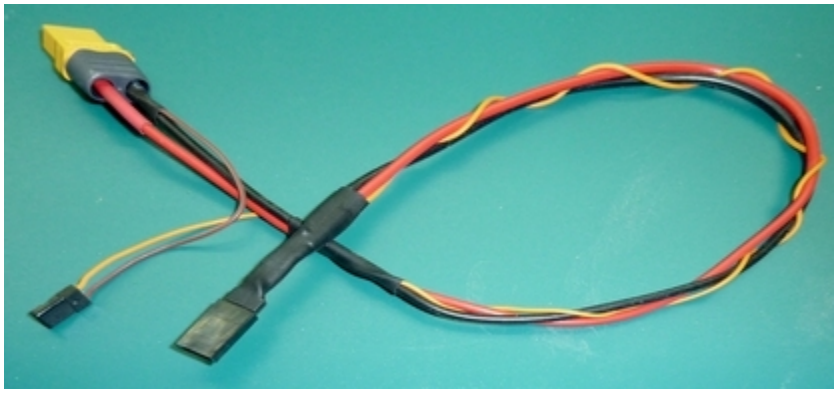


The bits you need

A 500 mm 22 awg servo extension, 500 mm each of red and black 16 awg silicone covered wire, an XT60 female connector and various bits of heat shrink. Which is the redundant part in the picture below? Note the use of male and female correctly refers to the metal parts, not the outside case as some non-electronically-knowledgeable people use to confuse us. Or perhaps so we buy the wrong thing and have to place another order?

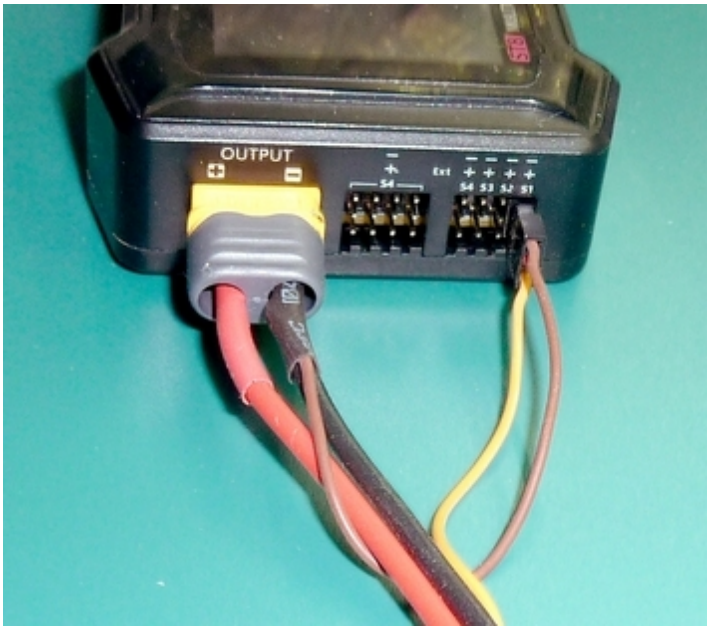


Complete



Note that you need to mark which side of the JR plug is the signal pin. I had to alter the above lead to show that. I checked the cable for continuity and shorts with a multimeter and then it was time to try it out.

Plugged in



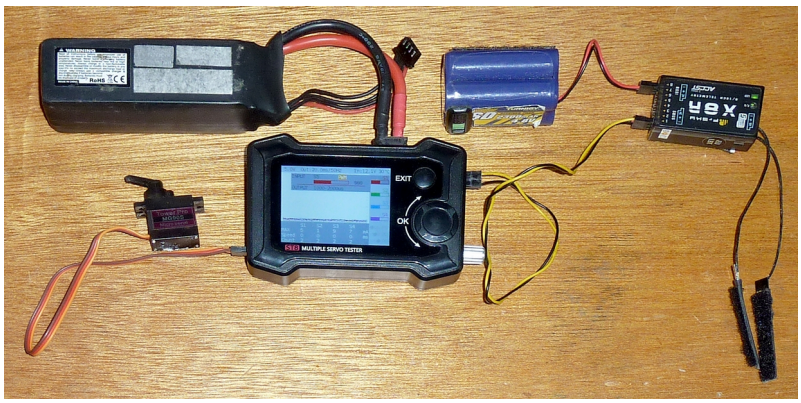
I decided to try this out with the Aerostar ASI-621MG.
The first step was to turn on the **OUTPUT** socket.
I went into Setup and switched the socket on at 5 V.
I plugged the servo in using the lead I had made.
It worked.
I used the speed test as above and got the same results for timing.
I did not get any useful data for current though, as it showed about 450.

This arrangement is useful for when the servo shows a current significantly more than 2000 mA on the normal test. It allows you to test servo speeds safely. It shows you that you need a bigger BEC or a power box. But it does not tell you the current drawn.

Taking signals from a receiver

I decided to send a standard PWM throttle signal from a FrSky X8R receiver throttle channel 1. I chose this channel because the transmitter stick isn't spring loaded so I can set it to a value whilst I study it. The receiver was connected to a battery so I used a female to female servo lead with the red wire removed (yellow and black in the picture).

This how it looked:



It worked, once I set S5 to PWM rather that the default S.BUS. As I moved the throttle stick the servo moved and the vertical signal bars scudded across the screen. However apart from finding the exact values of throttle maximum and minimum signals I don't think I learned anything new.

Using the screen

In the manual there are pictures of servo signals expanded on the screen. I was hoping that I could find a way to do it, but failed. Perhaps it's a software version problem? None of the youtube videos covered it either, but then they usually aren't a great deal of help anyway.

A version (not aversion)

Any techie will tell you that version 1 of any software or hardware is never properly finished. Way, way back, the BBC very cleverly named the operating system for its first model B computer 'version 0.9'. When the bugs had been ironed out they released version 1.0. The OS was burned into a erasable read only memory chip (EEPROM). Back then computer enthusiasts knew how to rewrite the chip when the new version came out. Apart from those from Apple, which had already started to lock up its products, computers were open for people to change. So I imagine, and hope, that when the updated ST8 operating system version 1.1 appears my criticisms, limited though they are, will be sorted out.

Summary

This is a useful and reasonably priced piece of kit with a few rough edges in its software. The manual could be a lot better and you will have to make up at least one special lead. The key question is whether it is too complex to be of use to someone who just wants to do simple tests on ordinary servos. My answer is 'no', it's fine.