

## How to Setup the Serial Servo Controller

1. Because the serial servo controller board will not be able to provide enough power to all of the servo motors, an external power supply is needed. A DC power supply GPS-3030D with maximum current of 3 A is connected to the "VDC +5V" connector, and of course with the output voltage of 5 volt. The switch "+5V/+VEXT" must select +VEXT to choose the external power source for the servo motors.
2. Setup the maximum and minimum angles of the servo motors from the HyperTerminal program (refer to the manual). The HyperTerminal must be configured its baud rate to 9600. The controller address is set to 0. The minimum and maximum of the servo motors is setup as shown in the following table. Note that at the position 150 (equivalent to pulse width of 1.5 ms), a servo motor will rotate to its center position. Each servo motor typically has range of motion approximately  $\pm 90$  degree.

Channel	Motor No.	Min	Max
1	1	69	232
2	2	79	217
3	3	67	236
4	4	67	222
5	5	65	230
6	6	109	191

3. After the setup is complete, we can start using the controller. Make sure the "RUN/SETUP" switch on the controller board is on RUN mode. We will only have to setup the controller for the first time.

## How to Use the Serial Servo Controller via the Program ET-RSS v1.0

ET-RSS v1.0 is the program provided with the serial servo controller. The program allows user to control the servo motors that connected on the controller by using graphic interface. To use the program:

1. Power the controller board (both from adaptor and the external 5-volt power supply). The green and red LEDs should be on.
2. Press the switch "SELECT", the red LED should be off, and the yellow LED should be on.
3. Open ET-RSS v1.0 program.

4. Choose COM port (we have COM3), ID address (our controller address is 0), and Delay Time to 1.
5. Move the slider bar around, the servo motors should move. If a slider is moved outside the min/max ranges we setup earlier, the motor will not move until the slider is moved back within the range. This is to prevent the servo motor from being damaged.

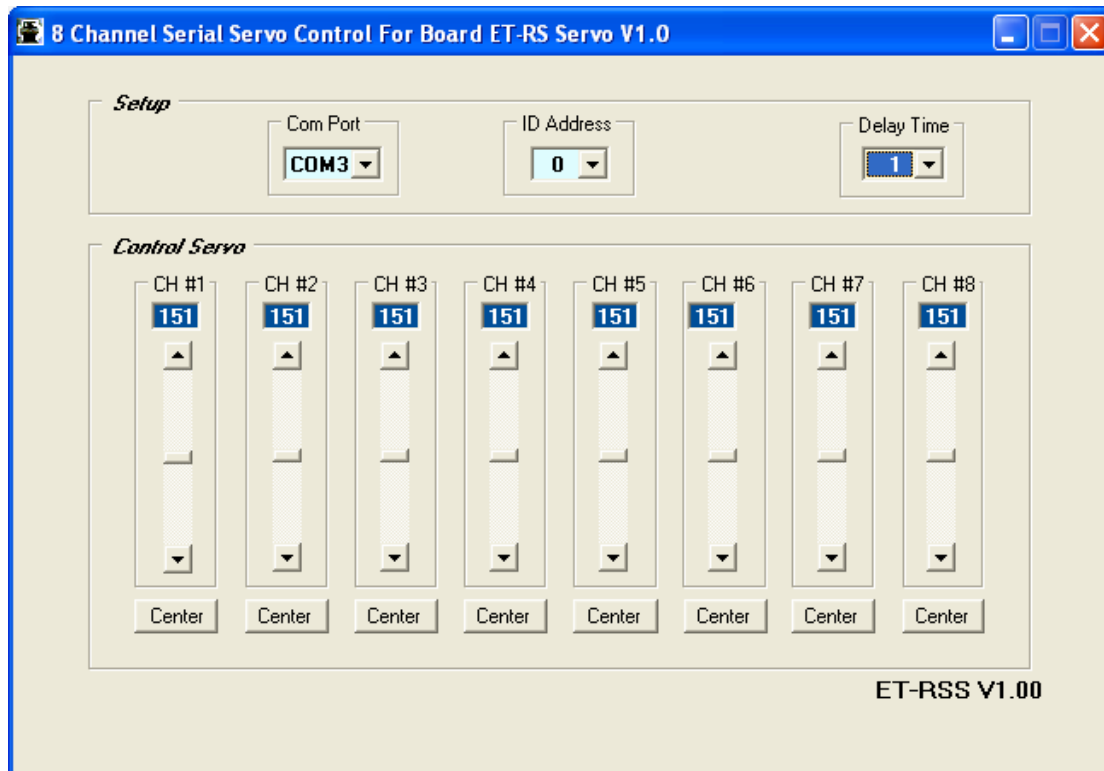


Figure 1 The graphic user interface of the program ET-RSS v1.0 from ETT company

### How to Program Step Commands to the Serial Servo Controller

1. Make sure that we are able to use the program ET-RSS v1.0 to control motors. (the red LED should be off, and the yellow LED is on.)
2. Select delay time. This is the delay time after the servo motor moves to a desired position.
3. Move a slider bar (only one motor per step) to the desired position.
4. Press "SAVE" switch, the red LED will blink once to confirm the save operation.
5. Repeat step 2-3 until all the steps are programmed.
6. If you made a mistake, you have to start over again by exit this mode and reenter.
7. Press "UP" switch to start the motions to programmed steps. All LEDs (green, red, yellow) will be on in this mode. To exit this mode press "SELECT" switch. After exit the robot can be controlled by using the program ET-RSS v1.0.

## How to Use the Serial Servo Controller by using Matlab Program

The serial servo controller can be controlled by any device via serial port (RS-232). In our case, we want to control the servo motors by using Matlab program. Therefore, Matlab must be configured to send commands out via serial port in the format specified by the serial servo controller. Here is the instruction to use the controller by using Matlab program.

1. Power the controller board (both from adaptor and the external 5-volt power supply). The controller will automatically be in the proper mode to control the robot via serial port. In this mode, the green and red LEDs should be on. (If the controller is in other modes, you may press the switch "DOWN" to get in this mode.)
  2. Start Matlab program.
  3. Initialize serial port by using command  
`>> s = serinit;`
  4. After the serial port is initialized. The robot can be moved around. For example, if we want to move the servo motor #3 to the position 160, use the command  
`>> sendcmd(s,3,160);`
  5. When finish using the robot, the serial port must be closed by using command  
`>> serclose(s);`
- See the example program in file example.m.

### File: serinit.m

```
function s = serinit
% This function initializes and returns serial port object

s = serial('COM3');
set(s,'BaudRate',9600);
fopen(s);
```

### File: sendcmd.m

```
function sendcmd(s,ch,pos)
% This function is to command the ETT serial servo controller via serial
% port. The serial port must be initialized by using function serinit
% first. The function take following arguments:
% S = serial port object (returned from function serinit)
% CH = channel of the servo motor to command
% POS = command position for the servo motor in range 50-250
```

```

% The ETT serial servo controller receives the 3-byte command format.
% byte1: Sync byte 0xFF = 255. If synchronized, it will send 0xFF back.
% byte2: Address(0-F) + Channel(1-8)
% byte3: Position (50-250)

% Clear up all the bytes received and not read
while s.BytesAvailable~=0
    out = fread(s,s.BytesAvailable);
end

% Send out sync byte until get the sync byte back
while s.BytesAvailable==0
    fwrite(s,255); % Send out byte 1 (sync byte): 0xFF
    pause(0.02); % Allow time for write operation (do not remove this line)
end

if (fread(s,1)==255) % Read one byte, check if it is 0xFF
    % Now the controller is synchronized, and ready for byte 2
    addr = 0; % The address of the serial servo controller
    byte2 = 8*addr + ch;
    byte3 = pos;
    fwrite(s,byte2); % Send out byte 2: address and channel
    fwrite(s,byte3); % Send out byte 3: position
end

```

**File: serclose.m**

```

function serclose(s)
% This function closes and deletes the given serial port object

fclose(s)
delete(s)
clear s

```