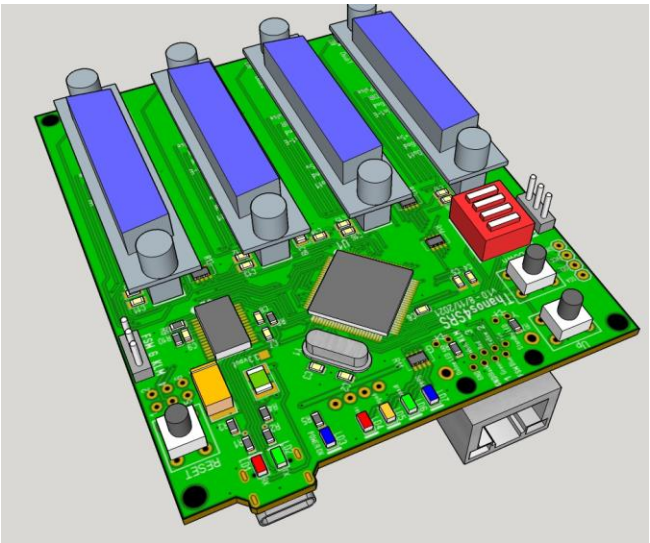
Thanos4U documentation

V1.04



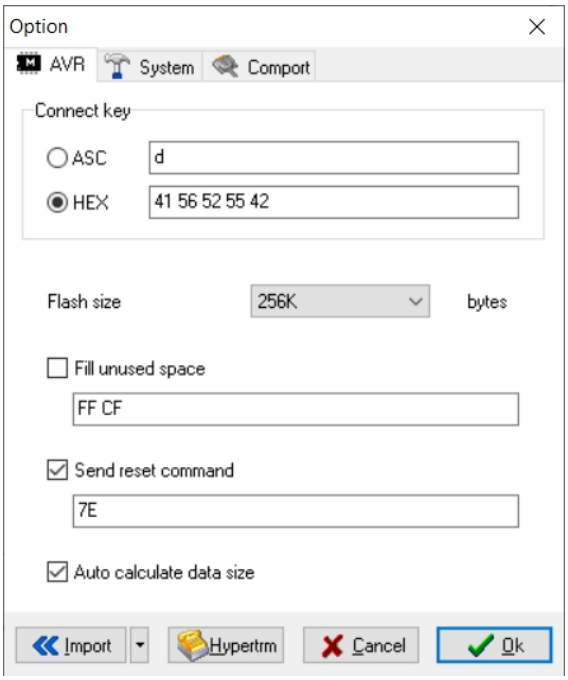
For firmware: Thanos4U_2560_4dof_v1_01_fix1

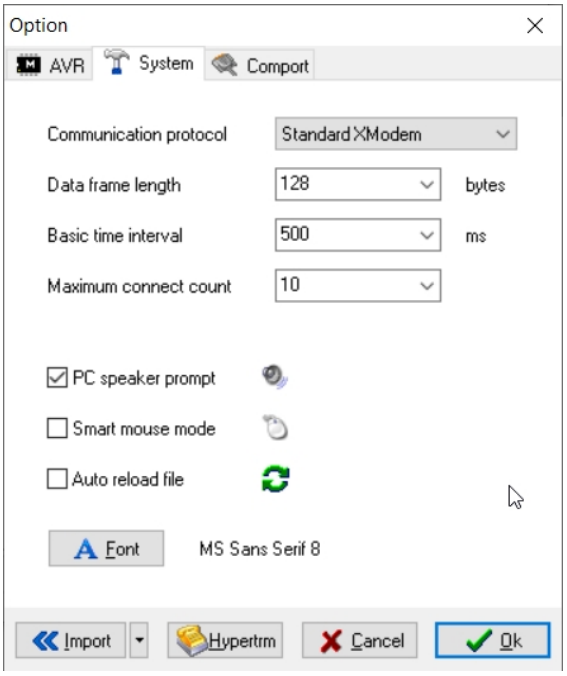
Contents:

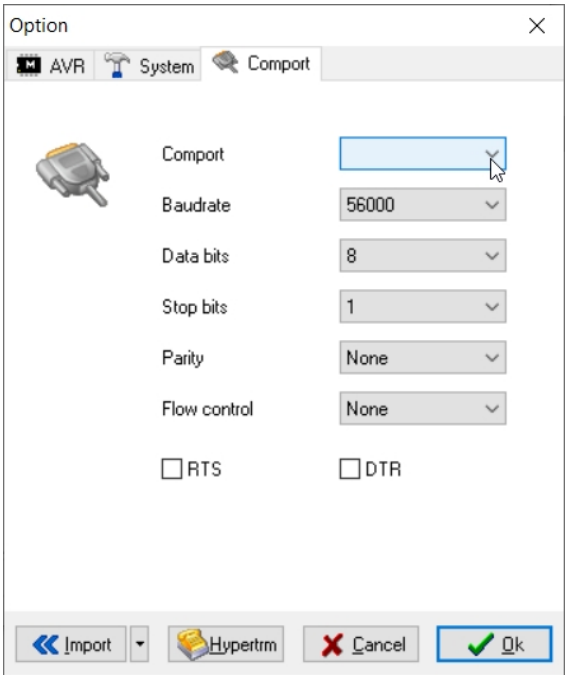
https://www.thanos-motion.com/

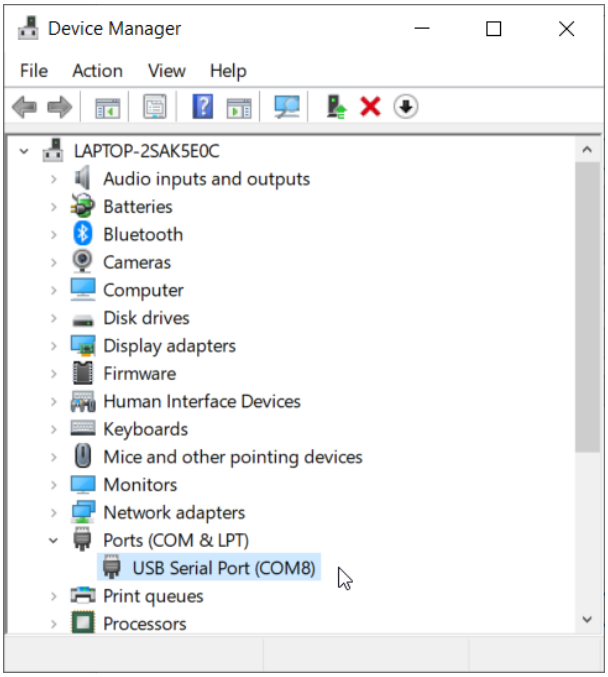
Follow us on Discord: https://discord.gg/bx4PxYR

Manually updating the firmware, AVRUBD settings:









Upon connecting the Thanos4U controller a COM port should appear in the Windows Device Manager COM ports section. If not, you may need to install FTDI drivers from here: https://ftdichip.com/drivers/vcp-drivers/

Data Packet format:

The data packet string is 20 bytes long and includes additional spare motion data slots for up to 8 actuators

The ID is byte values 0xFF + 0xFF
Each ACT is 16bit wide. (FF FF)
LF+CR is required in the end (0x0A + 0x0D)

ID ACT1 ACT2 ACT3 ACT4 ACT5 ACT6 ACT7 ACT8 LF/CR

Only ACT1-ACT4 respond to motion data, you can send 0 bytes to ACT5-ACT8, to keep the 20 bytes string compatibility. Baud speed is 250000bps, same as all AMC controllers.

I add the two bytes to form a 16-bit value (for 0 to 65535 range, with 32512 mid position) like this:
act1word = act1high
Shift act1word , Left , 8bits
act1word = act1word + act1low

where
act1word is word type (65535)
act1high is byte type
act1low is byte type

---Example of data to send for 4 actuator (Must include 0 values for not used axis):

1 0xFF ID
2 0xFF ID
3 0x7F ACT1 MSB
4 0x0F ACT1 LSB
5 0x7F ACT2 MSB
6 0x0F ACT2 LSB
7 0x7F ACT3 MSB
8 0x0F ACT3 LSB
9 0x7F ACT4 MSB
10 0x0F ACT4 LSB
11 0x00 ACT5 MSB
12 0x00 ACT5 LSB
13 0x00 ACT6 MSB
14 0x00 ACT6 LSB
15 0x00 ACT7 MSB
16 0x00 ACT7 LSB
17 0x00 ACT8 MSB
18 0x00 ACT8 LSB
19 0x0A LF
20 0x0D CR

*See example code in the end of this document

Upon power up (Plug USB cable) or reset via button it will display the following 3 lines:

Boot OK
Thanos Motion Electronics LLC 2021
Thanos4U(M) v1.01 fix0

The (M) indicates Main device for use with vertical actuators 1-4 , and (S) Secondary device for use with horizontal actuators 5-8 on a multi axis rig. Example of Secondary device indication:

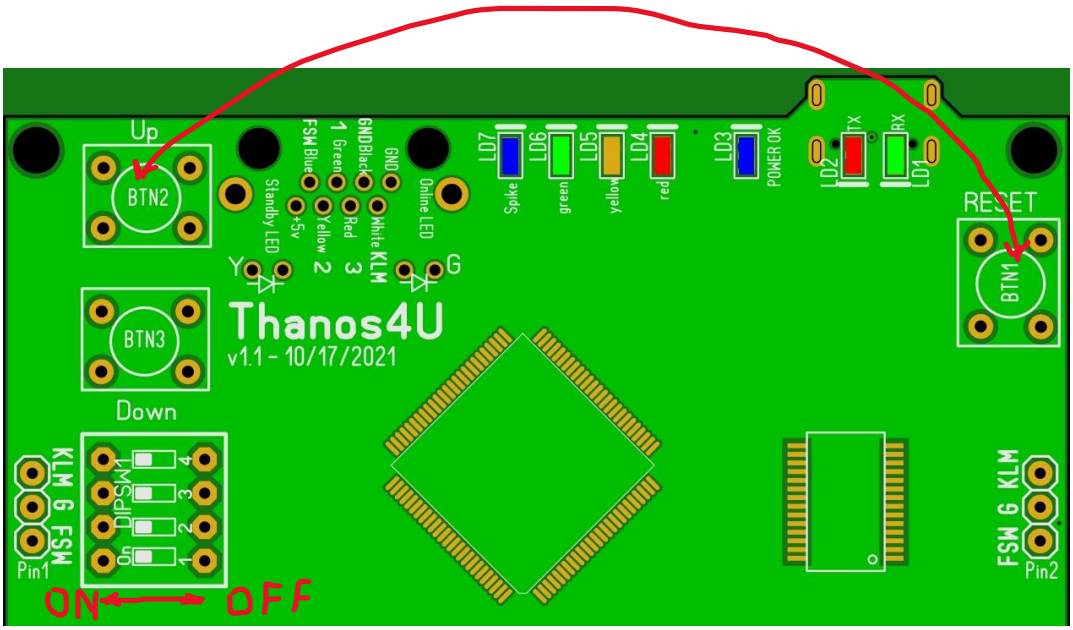
Thanos4U(S) v1.01 fix0

By using the CMD44, you can quickly see information for the actuators formatted as:

>CMD44
Act1: Stroke 100mm, Leadscrew 5mm/rev, Inline, Belt NO
Act2: Stroke 100mm, Leadscrew 5mm/rev, Inline, Belt NO
Act3: Stroke 100mm, Leadscrew 5mm/rev, Inline, Belt NO
Act4: Stroke 100mm, Leadscrew 5mm/rev, Inline, Belt NO

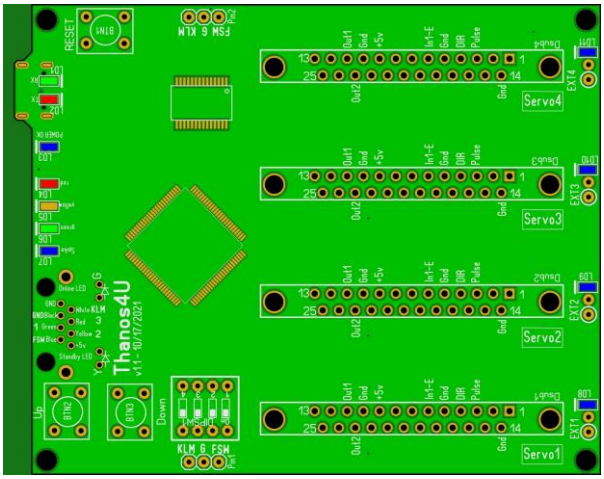
Restoring Factory Defaults:

To restore default values Hold pressed the “UP” button and press release the “RESET” button. The “POWER OK” LED should blink 10 times indicating the parameters were restored before it performs the normal LED color cycle on power up.



DIP SWITCHES positions meaning:

DIP #1: OFF = Main Device (Actuators 1-4)	ON = Secondary Device (Actuators 5-8)
DIP #2: OFF = LED lights active ,	ON = LED lights disabled
DIP #3: OFF = normal operation	ON = activates automatic servo motion test (DIP3 + DIP4)
DIP #4: OFF = normal operation	ON = activates automatic servo motion test (DIP3 + DIP4)



Thanos4U controller DB25 order markings and LED colors

Software Setup

SRS – Sim Racing Studio:

http://simracingstudio.rurl.me/thanos

SRS automatically detects the Thanos4U controller and its ready to use almost immediately. SRS will automatically set parameters it needs, including Spike filter or centering of actuators when using Secondary Thanos4U unit.

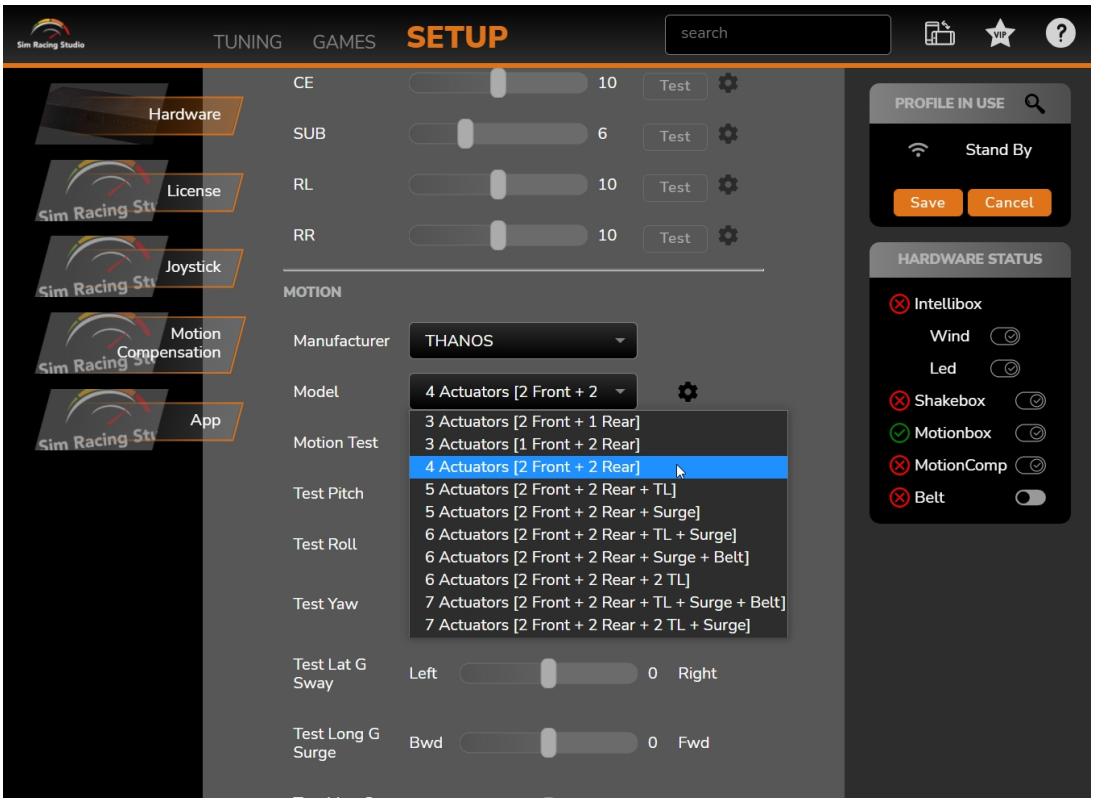
Quick start guide for AMC-AASD15A controller setup:
https://bit.ly/3lX6ON5

More links to information setting up for SRS:

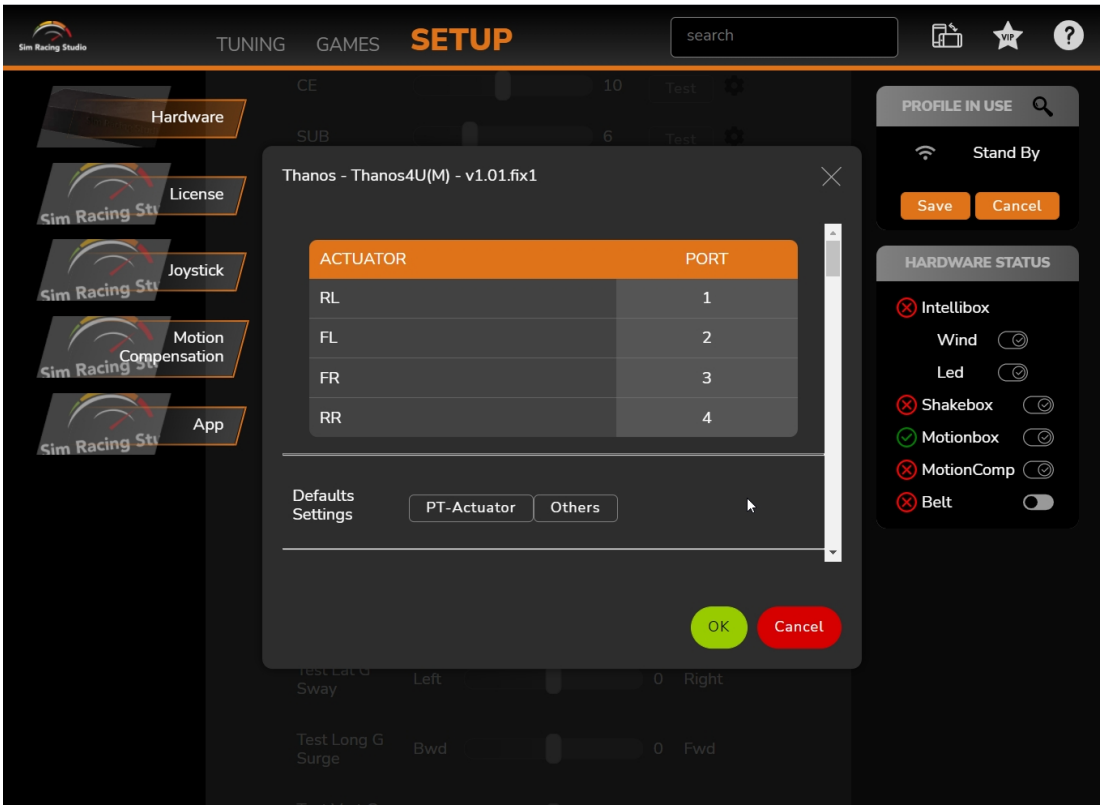
https://www.simracingstudio.com/forum/motion-profiles-actuator/actuator-setup-links

Don't forget to have a look at the Tuning guide:
https://www.simracingstudio.com/forum/motion-profiles/srs-2-0-motion-tuning-guide

SRS software basic configuration:

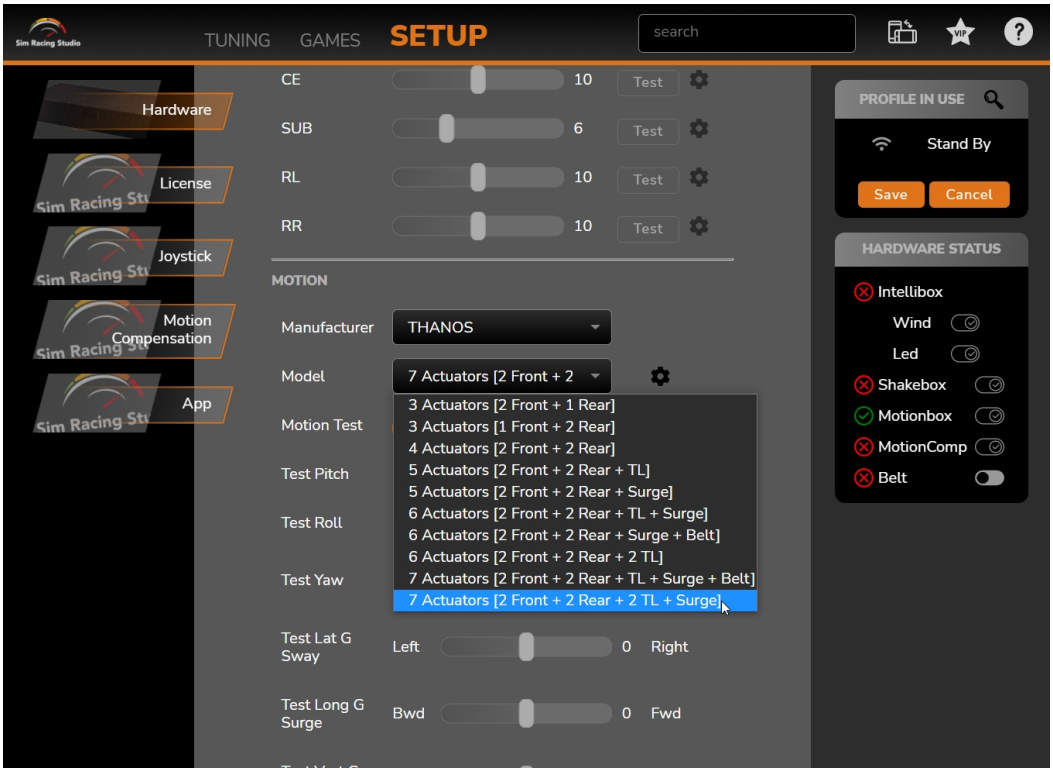


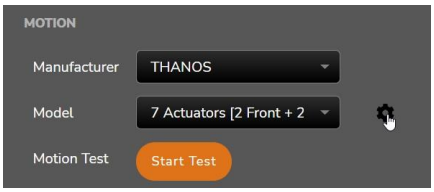
Rig type selection for using one Thanos4U unit

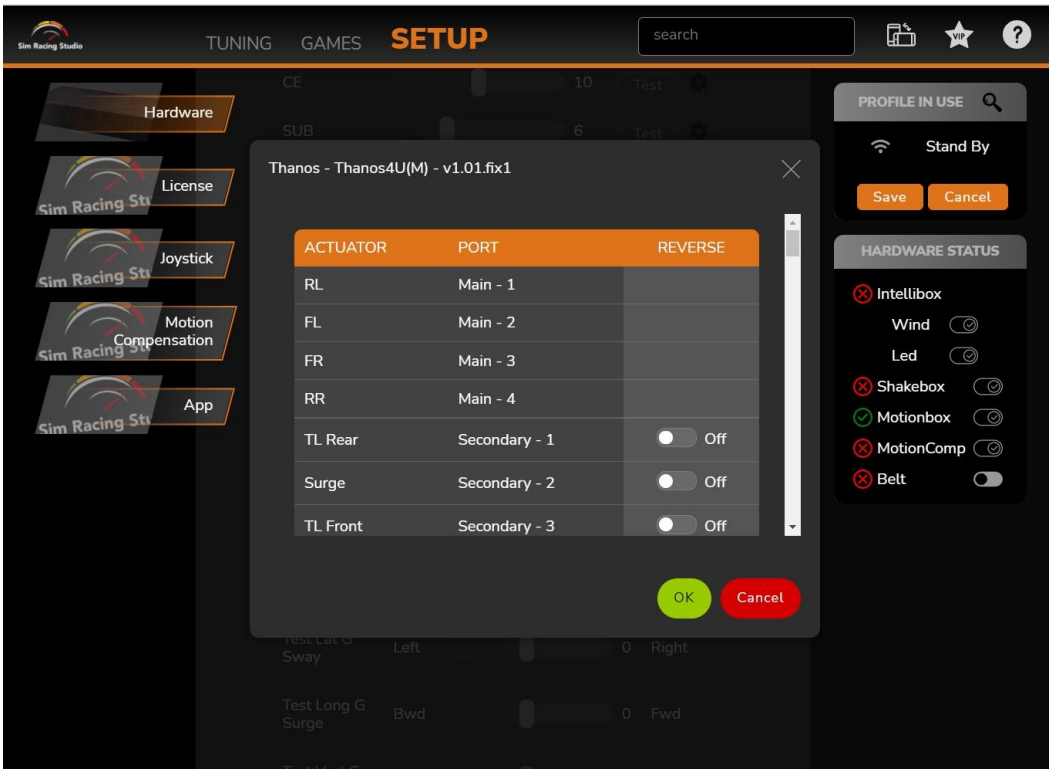


Actuator placement order for 4 actuator rig

Configuration using two Thanos4U controller for up to 7 actuators rig…

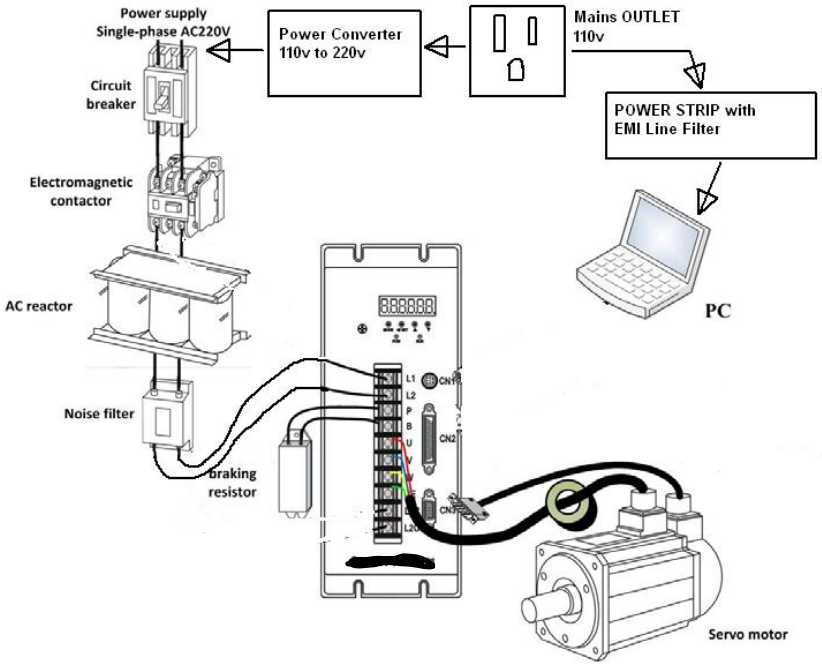




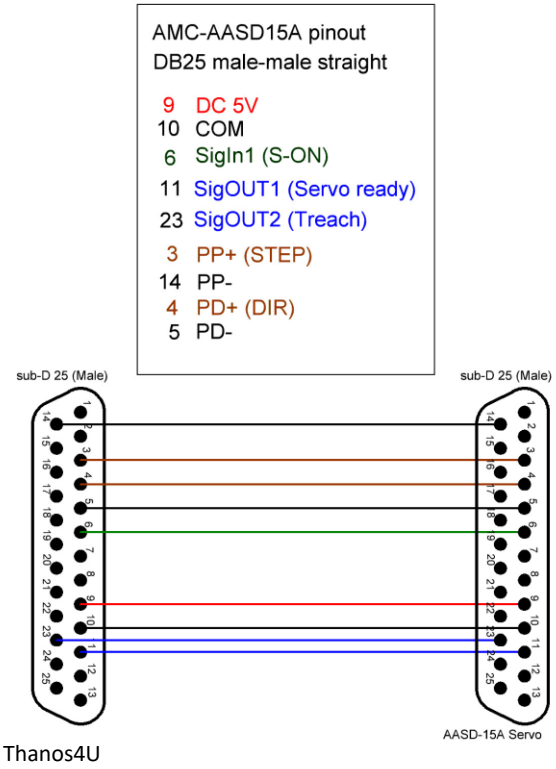


Testing manually motion using the sliders:

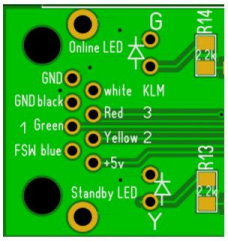
Consider adding a line filter to prevent some of the EMI noise to propagate to other connected devices on same power line like the PC or peripherals. Usually all system must be star connected to a single outlet to avoid ground loops:



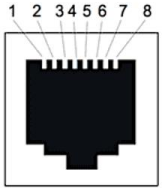
If you want to see which wires are used in the DB25 cables:



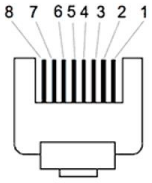
E-stop button box – RJ45 – Thanos4U



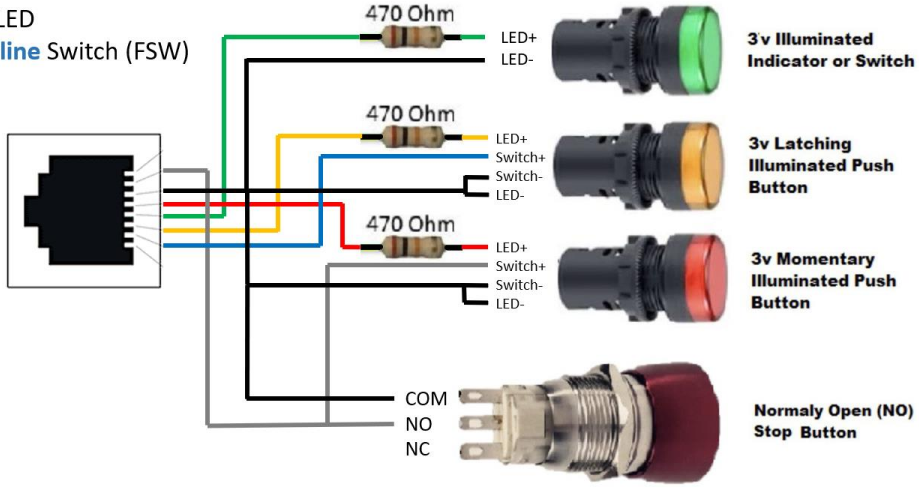
1. Gnd
2. **E-stop** Switch (KLM)
3. **Gnd**
4. RED LED
5. GREEN LED
6. YELLOW LED
7. **Force Offline** Switch (FSW)
8. +5v



Rj-45 Jack (Female)



Rj-45 Plug (Male)



AASD-15A Servo drives configuration:

The AASD-15A drives need some parameters before they are ready to be used.
Most of the parameters are same as SFX100 DIY, but some additional one are required.

AASD-15A Servo Settings:
Push MOD until you see Pn000. This enters the parameter mode.
Change and check these settings on all motors:

Pn8 = 300 - Max peak torque % CW
Pn9 = -300 - Max peak torque % CCW
Pn51 = 3000 - Max servomotor speed in RPM
Pn98 = 20 - Pulse Multiplier (electronics gear)
Pn109 = 1 - smoothing, (1=fixed smoothing, 2=s-Shaped smoothing)
Pn110 = 30 - Smoothing Filter Time
Pn113 = 20 - Feedforward %
Pn114 = 10 - Feedforward Filter Time (ms)
Pn115 = 100 - Gain %
Pn24 = 100 - Torque level for home calibration
Pn52 = 1
Pn60 = 2
Pn61 = 6

---List of commands---

Commands can be sent while the controller is offline and no motion data is sent. You can send CMD## or spv##$$$ in plain characters and numbers:

---CMD## , spv##$$$, where ##=parameter number and $$$= value 0-255

CMD01 Motornumber	: spv011-spv014	(4=default)
CMD02 DisableParktype	: spv021-spv025	(1=disabled)*
CMD03 Pulse_freq	: spv0300-spv0350	(00=default 200khz) – Don't alter this
CMD04 Park_Position	: spv04001-spv04254	(001=default 0%, 1-254 is percentage of the stroke)
CMD05 Park Move Speed	: spv05001-spv05254	(002=default) -Parkmovespeed
CMD06 Park move timeout	: spv0601-spv0690	(01=default) -Parkmovetimeout
CMD07 Standby Position	: spv07001-spv07254	(127=default 50%) -6dofstartpos
CMD08 Standby Speed	: spv08001-spv08254	(005=default) -Startmovestep
CMD09 Standby Timeout	: spv0901-spv0990	(05=default) -Startmovetimeout
CMD10 Platform_check	: spv101-spv102	(1=default disabled, 2=enabled)
CMD11 Rotation_offset	: spv11000-spv11254	(000=default) – initial backtrack after home calibration
CMD12 Buttonspeed	: spv1201-spv1299	(15=default) – how fast the actuators move with buttons manually
CMD13 Estop_mode	: spv131-spv132	(2=default kill power, 1=Servos Hold position powered)*
CMD14 Klm_mode	: spv141-spv142	(1=default)*
CMD15 Spike filter Range	: spv15001-spv15254	(127=default) -Spike_f_range
CMD16 Spike filter Level	: spv16001-spv16254	(127=default) -Spike_f_mult1
CMD17 Spike 1=On, 2=Off	: spv171-spv172	(2=default disabled) -Spike_filter_En
CMD18 TL Spike Level	: spv18001-spv18254	(127-default) -Spike_f_mult1_tl
CMD19 TL Spike On-Off	: spv191-spv192	(2=default disabled) -Spike_filter_en_tl
CMD20 Filter_factor	: spv201-spv205	(0=default no pulse filter)*
CMD21 Force_offline_mode	: spv211-spv212	(2=default move to Park position, 1=move to standby position)
CMD22 Spike Esc speed	: spv22001-spv22254	(024=default) -Spike_move_speed
CMD23 Spike Fine speed	: spv23001-spv23254	(011=default) -Spike_finemove_speed
CMD24 Stroke	: spv24001-spv24050	(010=default 100mm)*
CMD25 Stroke2	: spv25001-spv25050	(010=default 100mm)*
CMD26 Stroke3	: spv26001-spv26050	(010=default 100mm)*
CMD27 Stroke4	: spv27001-spv27050	(010=default 100mm)*
CMD28 Lead_screw	: spv281-spv284	(2= default 5mm/rev , 1=4mm/rev, 3=10mm/rev, 4=25mm/rev)
CMD29 Lead_screw2	: spv291-spv294	(2= default 5mm/rev , 1=4mm/rev, 3=10mm/rev, 4=25mm/rev)
CMD30 Lead_screw3	: spv301-spv304	(2= default 5mm/rev , 1=4mm/rev, 3=10mm/rev, 4=25mm/rev)
CMD31 Lead_screw4	: spv311-spv314	(2= default 5mm/rev , 1=4mm/rev, 3=10mm/rev, 4=25mm/rev)
CMD32 Inline	: spv321-spv322	(1=default inline, 2=foldback)
CMD33 Inline2	: spv331-spv332	(1=default inline, 2=foldback)
CMD34 Inline3	: spv341-spv342	(1=default inline, 2=foldback)
CMD35 Inline4	: spv351-spv352	(1=default inline, 2=foldback)
CMD36 Ratio	: spv361-spv363	(1=default 1:1, 2=1:1.5, 3=1:2)
CMD37 Ratio2	: spv371-spv373	(1=default 1:1, 2=1:1.5, 3=1:2)
CMD38 Ratio3	: spv381-spv383	(1=default 1:1, 2=1:1.5, 3=1:2)
CMD39 Ratio4	: spv391-spv393	(1=default 1:1, 2=1:1.5, 3=1:2)

CMD44 Displays actuator info
CMD55 Prints short, delimited parameters values list
CMD56 SRS Prints short, delimited parameters values list
spv45 Saves all parameters at once*

spv46 Saves all SRS parameters at once
RQM Displays model, revision, and number of motors
Park Parks the actuators if in standby mode

Some explanation of the parameters with Asterisk *:

CMD24 Stroke : spv24001-spv24250 (010=default 100mm)*
The stroke has to be limited to the max available pulses that can be up to 65535 so it will less than the 16bit position value we receive from SRS for each actuator. The final stroke is scaled to match the 0-65535 position packet. Here is how the max stroke value is calculated for various leadscrew and belt ratio options:

```
'150mm Stroke
  'Leadscrew 4mm / Rev , 37rev , 1100 pulses/rev
  'Leadscrew 5mm / Rev , 30rev , 1000 pulses/rev
  'Leadscrew 10mm / Rev , 15rev , 500 pulses/rev
  'Leadscrew 25mm / Rev , 6rev , 188 pulses/rev

'One_rev * Stroke = Total Pulses    (Ratio 1:1)
  '1100 * 58 = 63800          (Max stroke 580mm for 4mm/rev leadscrew)
  '1000 * 64 = 64000          (Max stroke 640mm for 5mm/rev leadscrew)
  '500 * 129 = 64500          (Max stroke 1290mm for 10mm/rev leadscrew)
  '188 * 250 = 47000          (Max stroke 2500mm for 25mm/rev leadscrew)

'one_rev * stroke * ratio = total pulses  (Ratio 1:1/2)
  '1100 * 39 * 1.5 = 64350    (Max stroke 390mm for 4mm/rev leadscrew)
  '1000 * 43 * 1.5 = 64500    (Max stroke 430mm for 5mm/rev leadscrew)
  '500 * 86 * 1.5 = 64500     (Max stroke 860mm for 10mm/rev leadscrew)
  '188 * 166 * 1.5 = 46812    (Max stroke 1660mm for 25mm/rev leadscrew)

 'one_rev * stroke * ratio = total pulses (Ratio 1:2)
  '1100 * 29 * 2 = 63800      (Max stroke 290mm for 4mm/rev leadscrew)
  '1000 * 32 * 2 = 64000      (Max stroke 320mm for 5mm/rev leadscrew)
  '500 * 64 * 2 = 64500       (Max stroke 640mm for 10mm/rev leadscrew)
  '188 * 125 * 2 = 47000      (Max stroke 1250mm for 25mm/rev leadscrew)
```

CMD02 DisableParktype : spv021-spv025 (1=disabled)* Parks horizontal actuators to 50%
Disableparktype = 5 - four horizontal actuators (Act1, Act2, Act3, Act4)
Disableparktype = 4 - three horizontal actuators (Act2, Act3, Act4)
Disableparktype = 3 - two horizontal actuators (Act3, Act4)
Disableparktype = 2 - one horizontal actuator (Act4)
Disableparktype = 1 - no horizontal actuators, disabled

CMD14 Klm_mode : spv141-spv142 (1=default)*
1= will treat the e-stop as monetary button for 1 second to switch between Park and Standby positions
2= will immediately activate the e-stop function

CMD13 Estop_mode : spv131-spv132 (2=default kill power, 1=Servos Hold position powered)*
1= Will stop the motors and hold their position powered
2=Will cut the power of the motors, so if the leadscrew allows the actuator will start dropping if loaded.

CMD20 Filter_factor : spv201-spv205 (0=default no pulse filter)*
Servo motor pulses smoothing, to reduce vibrations and noise
1= Hard Filter, 2= Semi-Hard Filter, 3= Semi-Soft Filter, 4= Soft-Filter, 5= very soft-molasses

CMD55 Prints short, delimited parameters values list:

> data:v1.01:T4UM:4:1:2:5:127:5:5:1:0:1:127:127:2:1:2:24:11:127:2:0:2:2:0:10:10:10:10:2:2:2:2:1:1:1:1:1:1:1:1:

Here is what each parameter corresponds on the above reading:

"data:" Firmwareversion ":T4U"
(If Main_device) "M" or (If Secondary_device) "S"
":" Motornumber ":" Park_position ":" Parkmovespeed
":" Parkmovetimeout ":" 6dofstartpos ":" Startmovestep ":" Startmovetimeout
":" Disableparktype ":" “0” ":" Klm_mode ":" Spike_f_range ":" Spike_f_mult1
":" Spike_filter_en ":" Platform_check ":" Estop_mode ;
":" Spike_move_speed ":" Spike_finemove_speed ":" Spike_f_mult1_tl ":" Spike_filter_en_tl
":" Pulse_freq ":" ; Buttonspeed ":" Force_offline_mode ":" Filter_factor
":" Stroke ":" Stroke2 ":" Stroke3 ":" Stroke4 ":" Lead_screw ":" Lead_screw2
":" Lead_screw3 ":" Lead_screw4 ":" Inline ":" Inline2 ":" Inline3
":" Inline4 ":" Ratio ":" Ratio2 ":" Ratio3 ":" Ratio4 ":"

spv45 Saves all parameters at once*

When you save the parameters, only the ones that are different from the existing values in the epprom will be written. This way no unnecessary writes will be performed when you want to permanently same the parameters that are adjusted using the spv commands. Here is an example result:
> Saving... 4, 7, 13, 33, :[done] All changed parameters saved successfully:

CMD56 SRS Prints short, delimited parameters values list

>dataSRS:v1.01:fix0:T4UM:4:0:127:127:2:10:127:2:1:1:

Here is what each parameter corresponds on the above reading:

"dataSRS:" Firmwareversion ":" Betaversion ; ":T4U"
(If Main_device) "M" or (If Secondary_device) "S"
":" Motornumber ":" Filter_factor
":" Spike_f_range ":" Spike_f_mult1 ":" Spike_filter_en ":" Stroke ":" Spike_f_mult1_tl
":" Spike_filter_en_tl ":" Disableparktype ":" Platform_check ":"

spv46 Saves all SRS parameters at once

> 46.Saving... 15, 16, 17, 20:SRS_qsav: ...All parameters saved successfully:

Spike Filter calculation

To calculate the spike filter level value you need to combine the "Spike_range" and "Spike_factor" values as:

Spike_filter_level = Spike_range * Spike_factor
(given these are byte values with 0-254 values, they can combine up to 64516 value)

To convert the Spike filter level to mm distance you need to also use the "Stroke_cm" value:

Spike_mm_calc = Stroke_cm * 10
Spike_mm_calc = Spike_mm_calc / 65535
Spike_mm = Spike_filter_level * Spike_mm_calc

To reverse get the calculation for the spike level to send the spike level values you need, you can prepare the "Spike_range" and "Spike_factor" as:

To display on your code:
- Spike_filter_range = Spike_range * 255
- Spike_filter_level = Spike_range * Spike_factor

To encode to transmit to the AMC:
- Spike_range = Spike_filter_range / 255 (example: 23970/255 = 94)
- Spike_factor = Spike_filter_level / Spike_range (example: 23218/94 = 247)

To check the state of the spike filter:
Spike_filter_en = 1 (Spike filter is active)
Spike_filter_en = 2 (Spike filter is disabled)

If you want to access more information like the TL spike filter use the "CMD56" which returns:
>Thanos4U(M):v1.00:fix1:AASD::4:0:127:127:2:10:1:127:2

With the "CMD56" you can also read the TL spike filter level and if its activated or not.
If the TL spike filter is active it be used instead of normal spike filter for Front/Back TL axis and for Surge axis

I uses shared Spike_range value as with the normal spike filter. So the calculation apply as:
To display on your code:
- Spike_filter_range = Spike_range * 255
- Spike_filter_level_TL = Spike_range * Spike_factor_TL

To encode to transmit to the AMC:
- Spike_range = Spike_filter_range / 255 (example: 23970/255 = 94)
- Spike_factor_TL = Spike_filter_level_TL / Spike_range (example: 23218/94 = 247)

8000 / 255 = 031
4000 / 31 = 129
1800 / 31 = 058

Spv35031
Spv33129
Spv39058

35.Spike Range:31 (on display 7905)
33.Spike Level:129 (on display 3999)
39. TL Spike Level:58 (on display 1798)

To alter the parameters for Filter factor and spike filter you need to use the spv#### commands to transmit the correct values to correct registers.
The spv consists of an address and the value, for example the spv15127 is the address 15 and value 127. Use leading zeros for small value like spv15002.

Command	display value	Command to save value
CMD15	"Spike Range:"	spv15001-spv15254"
CMD16	"Spike Level:"	spv16001-spv16254"
CMD17	"Spike On-Off:"	spv171-spv172"
CMD18	"TL Spike Level:"	spv18001-spv18254"
CMD19	"TL Spike On-Off:"	spv191-spv192"
CMD20	"Filter Factor 0-5:"	spv200-spv205"

Other commands:
CMD56 "SRS Prints short delimited command list"
spv46 "Saves all SRS parameters at once"

Remember not to use extra characters when you send CMD or spv commands (No line ending characters)

All parameters transmitted with spv are temporarily saved in memory of the controller and will be lost on reset.
If you wish to save the parameters in the EEPROM memory permanently, use the command "spv46". It will return:
> 46.Saving... 15, 16, 17, 20:SRS_qsav: ...All parameters saved successfully:

~

AVRUDB Firmware update info (via xmodem)

Communication Protocol: Standard XModem
Data Frame length: 128 bytes (not including control bytes and checksum bytes)
interval between connections: 500ms
maximum send connect command count: 50 times

Baud 56000, data bits 8, stop bits 1, parity none, flow control none.
ConnectKEY[] (in hex) 41 56 52 55 42

-----Simplified example code for sending axis data (for arduino):

```
int outputValue0 = 0;        // value output
int outputValue1 = 0;        // value output
int outputValue2 = 0;        // value output
int outputValue3 = 0;        // value output
int outputValue4 = 0;        // value output
int outputValue5 = 0;        // value output
int outputValue6 = 0;        // value output
int outputValue7 = 0;        // value output
byte buf0[2];
byte buf1[2];
byte buf2[2];
byte buf3[2];
byte buf4[2];
byte buf5[2];
byte buf6[2];
byte buf7[2];
byte ID[2];
byte endstring[2];
void setup() {
  Serial.begin(250000);
}
void loop() {
  // ID ACT1 ACT2 ACT3 ACT4 ACT5 ACT6 ACT7 ACT8 LF/CR
  // - The ID is byte values 0xFF + 0xFF
  // - Each Axis is 16bit wide.
  // - LF+CR is required in the end (0x0A + 0x0D)
  // change the analog out value:
  ID[0] = 255;
  ID[1] = 255;
  buf0[1] = outputValue0 & 255;
  buf0[0] = (outputValue0 >> 8) & 255;
  buf1[1] = outputValue1 & 255;
  buf1[0] = (outputValue1 >> 8) & 255;
  buf2[1] = outputValue2 & 255;
  buf2[0] = (outputValue2 >> 8) & 255;
  buf3[1] = outputValue3 & 255;
  buf3[0] = (outputValue3 >> 8) & 255;
  buf4[1] = outputValue4 & 255;
  buf4[0] = (outputValue4 >> 8) & 255;
  buf5[1] = outputValue5 & 255;
  buf5[0] = (outputValue5 >> 8) & 255;
  buf6[1] = outputValue6 & 255;
  buf6[0] = (outputValue6 >> 8) & 255;
  buf7[1] = outputValue7 & 255;
  buf7[0] = (outputValue7 >> 8) & 255;
  endstring[0] = 10; //LF
  endstring[1] = 13; //CR
  Serial.write(ID, sizeof(ID));
  Serial.write(buf0, sizeof(buf0));
  Serial.write(buf1, sizeof(buf1));
  Serial.write(buf2, sizeof(buf2));
  Serial.write(buf3, sizeof(buf3));
  Serial.write(buf4, sizeof(buf4));
  Serial.write(buf5, sizeof(buf5));
  Serial.write(buf6, sizeof(buf5));
  Serial.write(buf7, sizeof(buf5));
  Serial.write(endstring, sizeof(endstring));
  delay(2);   // wait 2 milliseconds before the next loop
}
```