**Problems and questions about LabJack modules and LJM library**

**Mark Rivers, University of Chicago**

**November 23, 2022**

**Issue 1: T8 sockets**

**Summary:** T8 cannot sometimes reopen a socket for up to 30 seconds after closing one. This includes stream mode, which uses a separate socket.

**Status:** Fixed in firmware 1.0002.

**Issue 2: T8 cold junction temperature value**

**Summary:** T8 device temperature (cold junction) readout sometimes changes to an incorrect value, for example >400K. This seems to happen frequently when doing a stream input scan. A device reset is required to fix the problem.

**Status:** Soloman said on 11/22/2022 that they could not reproduce the issue and were planning on closing the issue. However, on 11/23/2022 I showed how I could easily reproduce the issue when doing an output stream.

**Issue 3: T8 stream output buffer status does not work properly.**

**Summary:** When doing a stream out operation, for example stream DAC0 and DAC1 with 100 Hz rate, the STREAM\_OUT0\_BUFFER\_STATUS always returns 0, rather than increasing as a non-looping scan proceeds. The actual waveform out as observed on an oscilloscope is correct. The same code works fine on the T4, T7, and T7-Pro where STREAM\_OUT0\_BUFFER\_STATUS changes correctly.

**Status:** Fixed in firmware 1.0002.

**Issue 4: T8 DAC stream output cannot start stream for some combinations of number of points and rate.**

**Summary:** If I attempt to stream out with certain combinations of number of points and rate. I get err=1302 LJME\_COULD\_NOT\_START\_STREAM. Note that it does **not** give err=2608 STREAM\_SCAN\_RATE\_INVALID.

For example the following combinations work and fail:

|  |  |  |  |
| --- | --- | --- | --- |
| **Number of samples** | **Requested rate** | **Actual rate** | **Status** |
| 2000 | 4000.000 | 4006.410156 | Success |
| 2001 | 4002.000 | XXX | Fail, err=1302 |
| 1000 | 2000.000 | 2003.205078 | Success |
| 1001 | 2002.000 | XXX | Fail, err=1302 |
| 2000 | 1000.000 | 996.492371 | Success |
| 2001 | 1000.500 | 996.492371 | Success |
| 2002 | 1001.000 | XXX | Fail, err=1302 |

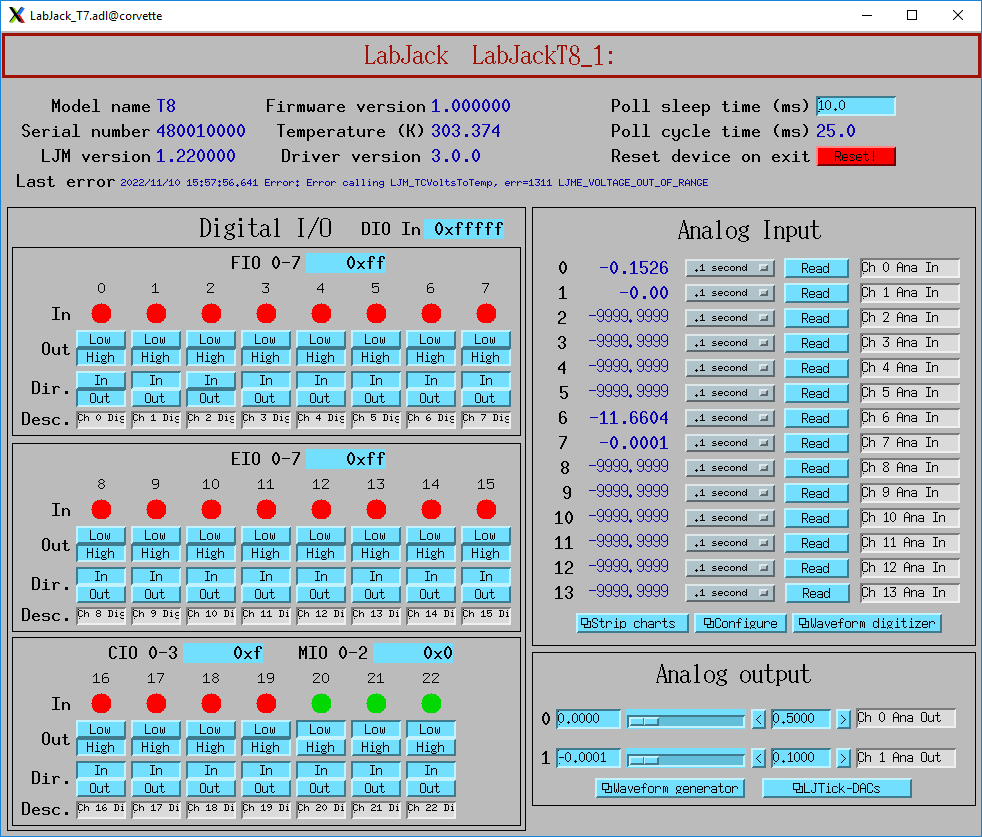
These combinations all work fine on the T4, T7, and T7-Pro. For those devices the actual rate is also much closer to the requested rate than it is on the T8

**Status:** Soloman said on 11/22/2022 that they are still working on this issue.

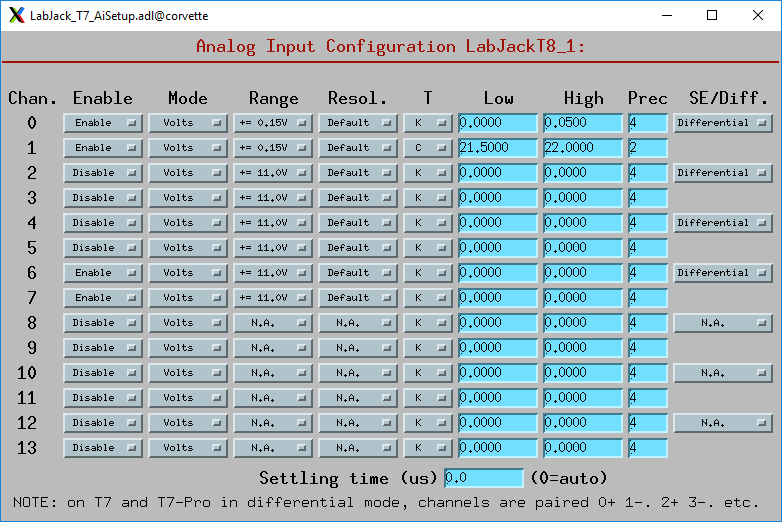
**Issue 5: T8 time to read analog inputs can vary between 1 and 8 ms. Seems to have no relationship to voltage range or resolution.**

**Summary:** The time to poll the analog input channels on the T8 implies a readout time of about 8 ms per channel after a reset. However, after doing a stream input or stream output operation the time can become much less. I have believe I have observed 1-4 ms per channel.

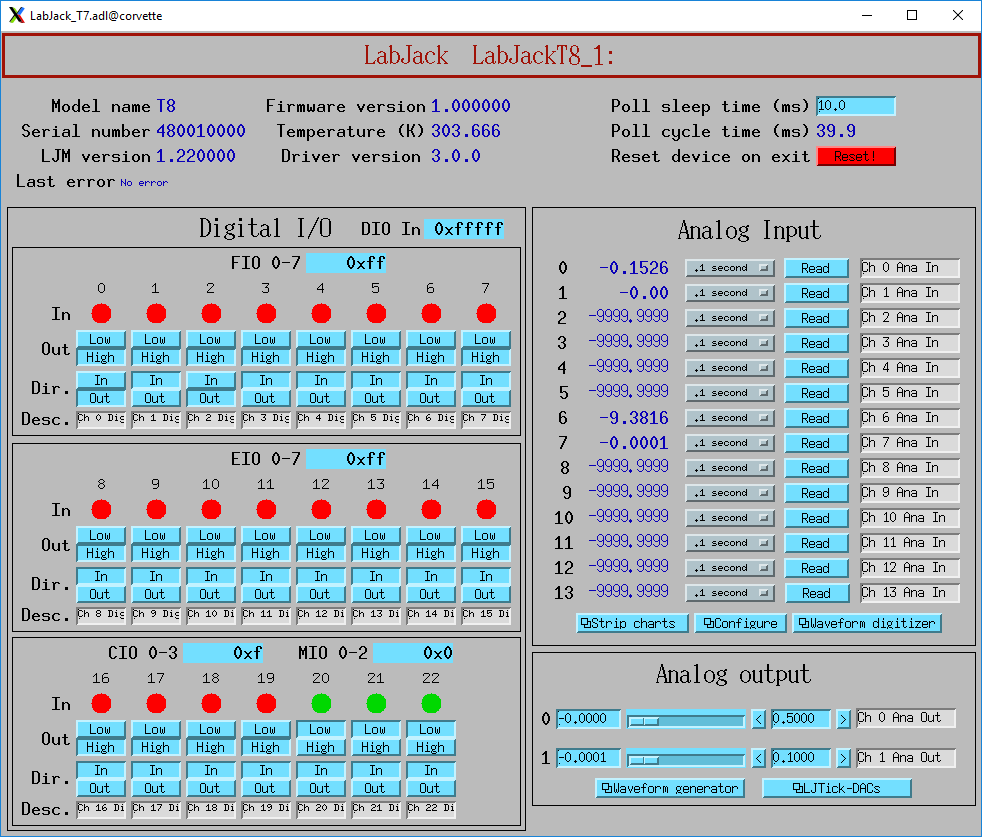
The following screen shot shows that the poll sleep is 10 ms and the measured poll cycle time is 25 ms. The time actually polling is thus about 15 ms .On each poll it is reading the DIO\_STATE and 4 analog inputs for the channels that are enabled. (channels 0, 1, 6, 7). If the DIO\_STATE read is fast then each AIN read is taking about 4 ms.



The following screen shows the configuration of the analog inputs. It shows which channels are enabled (i.e. being polled), and the mode, range, and resolution for each.



I then reset the T8 and restarted. The configuration shown above is identical. However, the poll cycle time is now 40 ms, or about 30 ms to actually read. This means the AIN reads are now taking about 8 ms, not 4 ms. Why?



**Status:** Soloman said the following on 11/16/2022

What you observed is expected behavior. The AINs are running at a specific frequency based on the AIN\_SAMPLE\_RATE\_HZ register. That register defaults to 100. The T8 will wait for a new sample before returning the results. If the period between samples is 10 ms then it can take 0-10 ms to get a result.

We do not currently have the AIN\_SAMPLE\_RATE\_HZ register documented. I will add some related information to the Analog Input section of the T-series datasheet.

I have now exposed this parameter which is actually called AIN\_SAMPLING\_RATE\_HZ in the EPICS driver. The behavior makes sense. Increasing the sampling rate increases noise, but results in more samples/s from the poller, up to about 2000/s. This allows more averaging to be done in the EPICS asynFloat64Average device support.

**Issue 6: Cannot set maximum number of samples that should be allowed with stream out.**

**Summary:** When setting STREAM\_OUT#BUFFER\_SIZE to N it works fine to write N/2-1 samples to STREAM\_OUT\_BUFFER\_F32. However, if N/2 samples are written there is no error but the outputs do not update. For example is STREAM\_OUT#BUFFER\_SIZE is set to 4096 then it works fine to write 2047 samples to STREAM\_OUT\_BUFFER\_F32. However, if 2048 samples are written then there is no error message, but nothing happens when LJM\_eStreamStart() is called. This happens on all models.

**Status:** Soloman said on 11/22/2022 that this is a limitation of the ring buffer design. It will not be changed, but they will document the behavior.

**Issue 7: T7-Pro and T8 cannot read the analog inputs when a stream output operation is in progress.**

**Summary:** On the T4 and T7 it is possible to read the analog input channels while a stream output operation is in progress. This is very convenient because one can connect the DAC outputs to the AIN inputs and monitor the output waveform. On the T7-Pro an attempt to read the analog inputs results in an error return from LJM\_eReadAddresses(). On the T8 there is no error, but the values read are all -9999. Is there a reason that the T7 can do this but the T7-Pro cannot? Can it be supported on the T8?

**Status:** LabJack support informed me on 11/14/2022 that the T7-PRO can read analog inputs, but only if the value of AIN\_ALL\_RESOLUTION\_INDEX is <=8. I changed the driver so that when starting a stream output operation on the T7-PRO AIN\_ALL\_RESOLUTION\_INDEX is set to 8 if it is currently 0 or >8. I was informed that the T8 cannot currently read the analog inputs while stream output is in progress. It might be added in the future, but it is not simple.

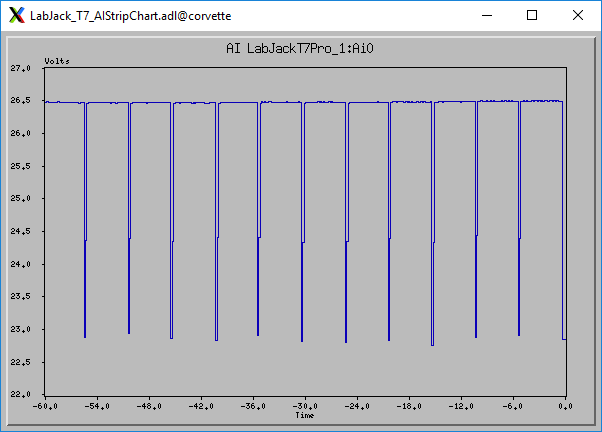
**Issue 8: There are very significant glitches in voltage readout immediately after reading TEMPERATURE\_DEVICE\_K.**

**Summary:** This is the issue I reported on November 4, and Soloman suggested increasing the settling time to 2000. That did not resolve the issue, but I have determined a solution described here.

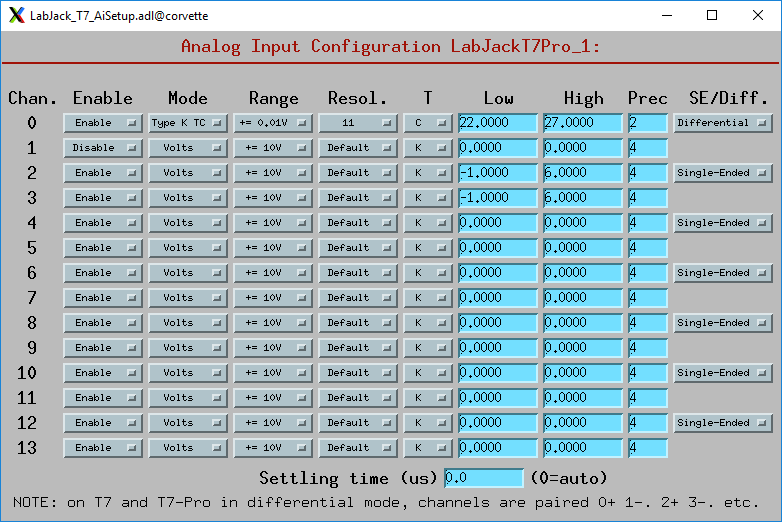
In this test on the T7-Pro I have the following connections to the analog inputs:

* AIN0: Type K thermocouple, differential mode, mode=Volts, conversion to degrees C in software. +- 0.1V range, resolution=11.
* AIN2: DAC0 output.
* AIN3: DAC1 output
* AIN4-13: No connection

This is what I see when reading TEMPERATURE\_DEVICE\_K in one thread every 5 seconds, while polling all AIN[0-13] in another thread at 10 Hz. Note that there are temperature glitches of about -3.5 C each time that TEMPERATURE\_DEVICE\_K is read. The glitch is not reduced by increasing AIN\_ALL\_SETTLING\_US to 2000, as Soloman suggested might work.

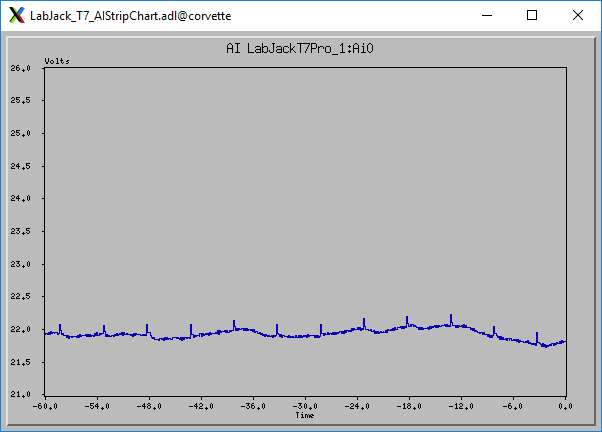


This was the configuration of the AIN channels:



I discovered that the primary reason for the glitch was the fact that the polling thread was reading the values of the AIN4-13 channels which have nothing connected to them. By changing these channels to Disable in the above screen they are not read by the polling thread.

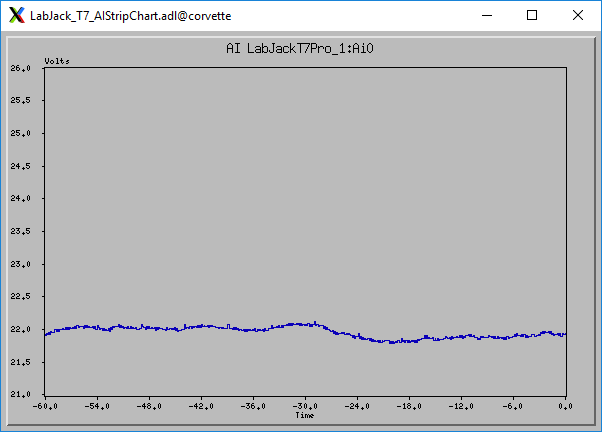
The input then looks like this.



There are 2 major changes:

* The temperature value read has changed from 26.5 to 22.0. This is a huge change.
* The glitches have changed from -3.5 C to +0.1C

I was then able to eliminate even these small glitches by changing the code that reads TEMPERATURE\_DEVICE\_K to also do a single dummy read of each of the enabled AIN channels and discard the results. The input then looks like this:



**Status:** Soloman said on 11/18/2022 that the problem was the I needed to add a 100 K ohm resistor between the negative thermocouple input and ground. That fixes the problem. The need for the resistor is documented here:

<https://labjack.com/pages/support?doc=/app-notes/sensor-types-app-note/thermocouples-with-the-t7-app-note/>

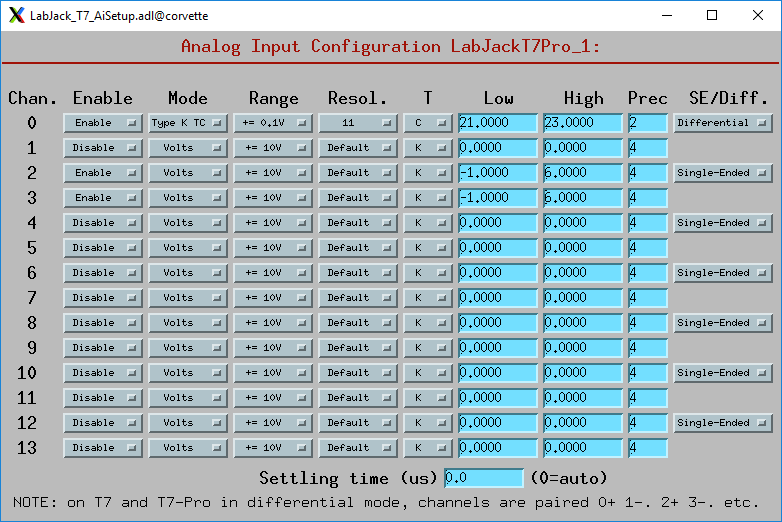
**Issue 9: There is significant cross-talk between analog inputs on the T7-Pro that cannot be reduced by increasing the settling time.**

**Summary:** In this test on the T7-Pro I have the following connections to the analog inputs:

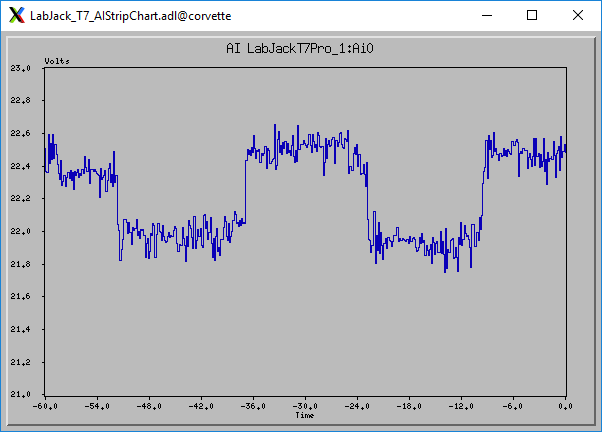
* AIN0: Type K thermocouple, differential mode, mode=Volts, conversion to degrees C in software. +- 0.1V range, resolution=11.
* AIN2: DAC0 output, single-ended mode, +- 10V range, resolution=Default.
* AIN3: DAC1 output, single-ended mode. , +- 10V range, resolution=Default.

In my polling thread I am only reading these 3 AIN channels.

This is the configuration screen.



This is a strip chart of the AIN0 temperature as I change both DACs between 0V and 5V. The AIN0 temperature about 22.6 when both DACs are set to 0. When both DACs are 5V the reported temperature drops by 0.6 C to 22.0. This cross-talk is independent of the value of AIN\_ALL\_SETTLING\_US, I have tried 0 (default), 5000, and 10000 and there is no difference.



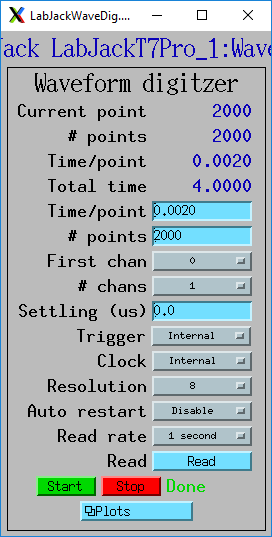
**Status:** Soloman said on 11/18/2022 that the problem was the I needed to add a 100 K ohm resistor between the negative thermocouple input and ground. That fixes the problem. The need for the resistor is documented here:

<https://labjack.com/pages/support?doc=/app-notes/sensor-types-app-note/thermocouples-with-the-t7-app-note/>

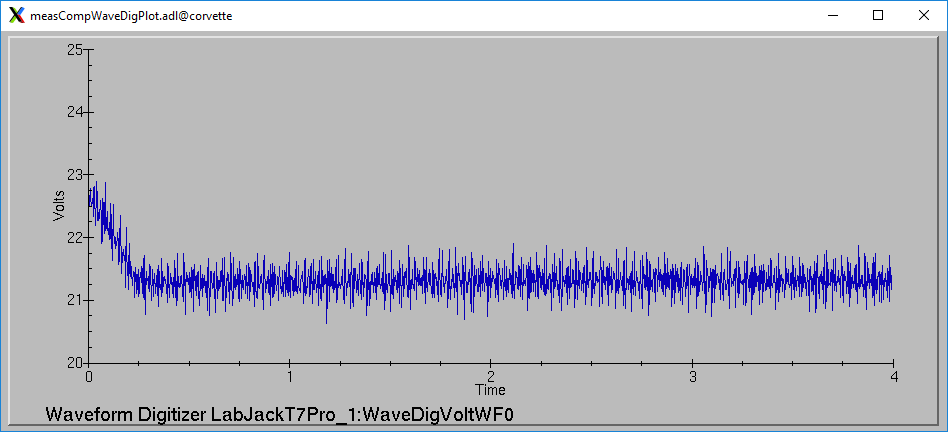
**Issue 10: Initial scan values have large errors on T7-Pro when running high gain.**

**Summary:** I have AIN0 connected to a type K thermocouple, differential mode with Range=+-0.01V. AIN2 is connected to DAC0 and AIN3 is connected to DAC1. Both DACs are at 0 V.

I do an input stream with just AIN0, rate=500 Hz (2 ms/point), with STREAM\_RESOLUTION\_INDEX=8. This shows the stream input configuration:

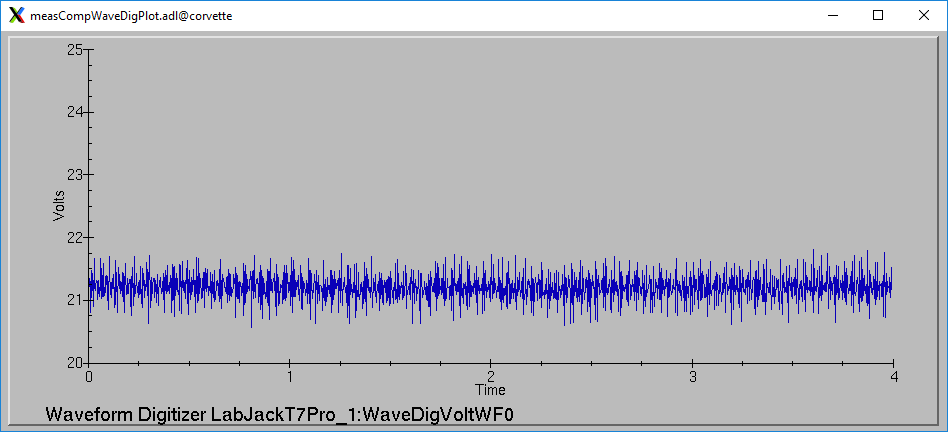


This shows the stream in data. The vertical axis is actually temperature, not Volts. Note that the temperature starts at 22.5 and drops to 21.5 in the first 0.25 seconds. This is reproducible, it happens on every scan.



Increasing STREAM\_SETTLING\_US to 1000 does not change the result. This is expected since I am only scanning 1 input, STREAM\_SETTLING\_US should do nothing.

I then change my polling so that when stream is inactive it is only polling AIN0, and not AIN2 and AIN3. The initial glitch is then gone. I therefore believe this problem is related to the cross-talk issue (#9) above.



**Status:** Soloman said on 11/18/2022 that the problem was the I needed to add a 100 K ohm resistor between the negative thermocouple input and ground. That fixes the problem. The need for the resistor is documented here:

<https://labjack.com/pages/support?doc=/app-notes/sensor-types-app-note/thermocouples-with-the-t7-app-note/>

**Issue 11: Limitations of the current LJM\_eStreamRead API.**

I think there are two important limitations/issues with the current LJM\_eStreamRead API. I also think there is a straightforward and simple fix.

1) It is very important to be able to detect when the user specifies incompatible values for rate and resolution. The documentation in section 3.2 of the T-Series Datasheet says:

Sample rate is too high: When the sample rate is too high, it causes a STREAM\_SCAN\_OVERLAP error and stream is terminated.

Scans are triggered by hardware interrupts. If a scan begins and the previous scan has not finished, the device stops streaming and returns a STREAM\_SCAN\_OVERLAP error (errorcode 2942), which LJM returns immediately upon the next call to LJM\_eStreamRead.

So in order to detect that a STREAM\_SCAN\_OVERLAP error has occurred the user needs to call LJM\_eStreamRead. The problem is that this does not work if one is using callbacks with stream input. If using callbacks the call to LJM\_eStreamRead() is done in the callback function. However, if there is a STREAM\_SCAN\_OVERLAP error LJM never calls the callback function and hence the user has no way to determine that this error has occurred.

2) It is important to be able to use external clocking for collecting each sample, and this must work for any value for the number of points N in a scan. When using external clocking it is possible that the user’s clock source will only output exactly N clock signals. For example, I use motion controllers where I can tell it to move from point A to point B and put out a clock pulse every 10 microns along the path. I know how many pulses it will output and I configure the T7 stream input scan accordingly. The value of N could even be a prime number, for example 853. In that case the only choice for scansPerRead that will work is 1 or 853. If I use 100, for example, then I will only be able to read values 0 to 799 because the clock signals for channels 853-899 will never arrive. Using 1 for scansPerRead is very inefficient, and using 853 means I cannot visualize any intermediate results.

It seems to me that the solution to both of these problems is to implement something that it appears that has been already considered for LJM. In LabJackM.h there are these lines, all commented out.

/\*\*

\* Name: LJM\_STREAM\_SCANS\_RETURN\_AVAILABLE

\* Desc: A mode that will cause LJM\_eStreamRead to never sleep, and always

\* consume the number of scans that the LJM handle has received, up to

\* a maximum of ScansPerRead. Fills the excess scan places in aData

\* not read, if any, with LJM\_SCAN\_NOT\_READ.

\* Note: ScansPerRead is a parameter of LJM\_eStreamStart.

\* TODO: LJM\_STREAM\_SCANS\_RETURN\_AVAILABLE is not currently implemented.

\*\*/

// LJM\_STREAM\_SCANS\_RETURN\_AVAILABLE = 3

Implementing LJM\_STREAM\_SCAN\_RETURN\_AVAILABLE is exactly what is needed. In my case it could simplify my driver a lot, because I could call LJM\_eStreamRead within my poller, and not have to implement a callback function at all. I cannot currently do that because LJM\_eStreamRead currently blocks, but this new mode would not block. It fixes the scansPerRead issue because I can that that to 100, but at the end of my 853 point scan it will just return the final 53 points. It fixes the problem of detecting the STREAM\_SCAN\_OVERLAP error because I can call LJM\_eStreamRead without using callbacks.

**Status:** Soloman replied on 11/11/2022.

I would agree that we should add a feature to consume remaining scans in the cases that you need to consume less than scansPerRead scans. However, note that there is currently a mode available that makes eStreamRead non-blocking. See the LJM\_STREAM\_SCANS\_RETURN\_ALL\_OR\_NONE mode described here:

https://labjack.com/pages/support?doc=/software-driver/ljm-users-guide/ljm\_stream\_scans\_return/

I changed the driver to use LJM\_STREAM\_SCANS\_RETURN\_ALL\_OR\_NONE. This has 2 benefits:

* The code is simplified since it just calls LJM\_eReadStream() from the poller thread, does not need to use a stream callback function.
* It now receives an error if STREAM\_SCAN\_OVERLAP occurs because the user specified incompatible STREAM\_RESOLUTION\_INDEX and scan rate.

This solves item 1) above.

Item 2) is not resolved, it will fail if the device only receives N triggers and N is not a multiple of scansPerRead.