BLOG: RoboRIO Performance Insight and Optional Phoenix 5.14.1 Update

Hello FRC Community!

There's been several questions about the performance limitations of the roboRIO this season, particularly in regard to CAN-bus. In response to this, we'd like to provide some insight to better educate teams on the limitations of the control system. Additionally we made some tuning adjustments in Phoenix to help alleviate the symptoms caused by these limitations.

Please read the sections below for more information.

Omar Zrien Co-owner CTR-Electronics

roboRIO Limitations ("net comm")

The roboRIO has a limitation where there is delay with every call that goes to the NI Network Communication process (often called "FRC NetComm"). This includes:

- sending/receiving CAN
- getting team/match info (see note below)
- getting joystick data (see note below)
- Generally anything involving the Driver Station software. (see note below)
- Generally anything not from the FPGA.

Note

Most (but not all) of these are buffered in WPI C++/Java so they actually get called once every ~20 ms, regardless of how often you call getters. However the LabVIEW VIs appear to not be buffered and may experience this call delay.

These calls average approximately 0.3 milliseconds. Many of you may think that does not sound like much, but consider the number of get calls you execute on your peripheral devices per loop. Ten "get" calls on ten unique devices will yield 100 calls per loop, which would be 30ms (although Phoenix has optimizations to reduce this explained below).

The worst-case call time can also be much longer, several milliseconds in fact, depending on the task management of the operating system. We've found these worst-case events to occur intermittently and vary depending on:

- CPU load
- Ethernet traffic
- Threading strategies

These intermittent call-delays can occasionally trip the WPILIB Driver Station warnings:

- Watchdog not fed within 0.020000s (see warning below).

Loop time of 0.02s overrun.

Warning

This is not the same as the "Watchdog" issue that was addressed in roboRIO v14. That refers to the FPGA Watchdog, which is a component of the roboRIO, not WPILIB.

Which means you may be seeing these warnings despite having reliable control of the roboRIO during teleoperated operation. These intermittent events may also impact your robot negatively if you are using a software strategy that requires deterministic timing.

Phoenix 2017 - 2018 (last season)

In our library, there are typically three kinds of calls: getters, setters, and config routines.

Knowing that the average call time of the FRC/NI layer is 0.3 ms, we can predict the call time for the following scenarios:

~0.3 ms per call for any get* routines.

~0.3 ms per call for any set* or enable* routines where the input has changed since previous call.

~0 ms per call for any set* or enable* routines where the inputs have not changed since previous call

~4 ms for any successful config* routines if non zero timeoutMs is passed. These should be done on robot boot.

~X ms for any timed out config* routines if X timeoutMs is passed. These should be done on robot boot.

~0.3 ms for any config* if zero timeoutMs is passed. These should be done in the robot loop, if at all (generally not necessary). Success is not determined since there is no wait-for-response checking.

~4 ms for any successful configGet*. These are generally not necessary in a typical robot application.

Phoenix 2018 - 2019 (Kickoff release)

Over the summer of 2018, we added further optimizations to improve this. For example calling getSelectedSensorPosition() twice in your loop will not cost 0.6ms (2 X the average call time).

This is because Phoenix knows how often signals are updated, and can judge when it is appropriate to perform the call. We had beta teams test this and they reported improved performance.

Similarly if you call getters on signals from the same signal group only one of them will experience the 0.3 ms cost, and the rest will return the buffered data (at least until enough time has elapsed to justify checking the CAN-bus).

This reduces the actual calls into "FRC NetComm" considerably.

Phoenix 2018 - 2019 (New Optional Release v5.14.1.2)

For these optimizations to work reliably, we have thresholds to determine when to start checking the bus again for fresh data.

The kickoff release had conservative thresholds. This was appropriate given that the performance results were improved and this was a new feature for this season. However, given the number of teams reaching the limits of the roboRIO performance, we've released a new version with more aggressive thresholds (5.14.1).

To be clear, there are no API changes in this release. We would not feel comfortable making those types of changes during the competition season. This is merely an optional update for teams looking for any means of squeezing more performance out of their roboRIO + CAN bus peripherals.

As with any software component update, teams should re-validate all base functionality of their robot. If this cannot be done, then do not feel obligated to apply this update as this is entirely optional.

In general we recommend that teams attempting to use software loops for time critical tasks to:

Directly measure your "dT" (time between loop calls) and compensate for the measured deviations. WPILIB and LabVIEW provide routines to measure time.

LabVIEW teams can leverage the built-in profiler to profile their robot applications.

Consider updating to 5.14.1 if roboRIO limitations are impacting robot performance.

Review the number and type of "get" calls being done per loop. For example, if retrieving current-draw and sensor position, use getSelectedSensorPosition() instead of the routines in getSensorCollection(), since selected sensor position and current-draw are in the same status group.

Consider using the hardware-accelerated features of our motor controllers (Talon SRX, Victor SPX).

Note

There is a motor controller firmware update for teams using low-resolution sensors and Motion-Magic. However if you are using the feature successfully, you likely do not need to update.

Note

Some teams have opted to use alternative platforms that do not have the same call limitations. An example of this would be using Phoenix on Raspberry-Pi/Jetson TX2. These devices function by leveraging a kernel-based CAN-bus solution (socket-can).

How To download

Windows users can download the v5.14.1 Installer.

Alternatively, users can download the individual components:

Release page on GitHub: https://github.com/CrossTheRoadElec/Phoenix-Releases/releases

Firmware can be downloaded from the product pages on http://www.ctr-electronics.com/

Additionally teams can pull the latest Phoenix API via the online method through VS Code, or via the non-Windows zip.

Download instructions can be found here.

Note

The online method refers to the “Check for updates (online)” feature. However this is not recommended as this requires a live Internet connection to use your FRC project.