

# Using the CAN subsystem with the RoboRIO

Using CAN with the RoboRIO has many advantages over previous connection methods between the robot controller and peripheral devices.

1. CAN connections are through a single wire that is daisy-chained between all the devices so *home run* wiring isn't required.
2. Since this is protocol-based signaling the devices can be smart and accept higher level commands besides start, stop and set speed.
3. Devices can report status back to the robot controller making it possible to have much better control algorithms with devices that use CAN.

There are a number of CAN devices supported in the FRC control system:

1. Jaguar speed controllers
2. CAN-Talon speed controllers
3. The Power Distribution Panel (PDP)
4. The Pneumatics Control Module (PCM)

The devices are typically connected to the RoboRIO CAN bus using twisted pair wiring (except the Jaguar which uses modular phone-style connectors).

## CAN bus topology and termination

The CAN bus must be *terminated* at each end of the bus, that is bridged with a termination resistor of 120 ohms. Conveniently both the RoboRIO (start of bus) and the PDP board can supply termination. So a CAN bus that starts at the RoboRIO, goes through several devices, and ends at the PDP board (with the termination jumper installed) will provide the correct termination. Nothing else has to be done.

## CAN operation and timing

It's helpful to understand the timing of the messages sent and received by your program to and from the CAN bus. With this information it will be easier to write and debug CAN-based robot programs.

Completely rewritten CAN subsystem. All the CAN code is now non-blocking for the program except in the constructor. When creating new CAN Jaguars the system waits for a short period of time to receive version and some status information back from the device being created. This delay is 50ms maximum, but will continue on as soon as the data is received. If no information is

# Using the CAN subsystem with the RoboRIO

received within 50ms, the device creation will fail (either an exception in Java or an error status in C++).

- Setpoints that the program sets are updated to the CAN devices (Jaguars for now) every 20ms regardless of how often your program sets them. That is to say, once your program sets a set point, it will continue to be sent to the Jaguar in the background.
- Device status is returned every 20ms for each device automatically - the program does not need to request those updates. Whenever the program requests status from a CAN device it will be no older than 20ms.
- Configuration data only needs to be sent once. The software will verify that the actual values match the configured values and resend as necessary. This means that a Jaguar rebooting while the robot is running should have all of it's configuration values restored within a few updates after coming back.
- The background updating of values happens whenever any parameter is set on the Jaguar. Each time a value is set, the Jaguar objects will scan for any unverified data and request updates to that data. This means that the restoring of data will only happen when the program is updating some value (any value) in the Jaguar. This is a little non-obvious, but it means that there are no background threads and blocking calls required. The assumption is that the program will periodically set at least the set point, triggering the update to happen.