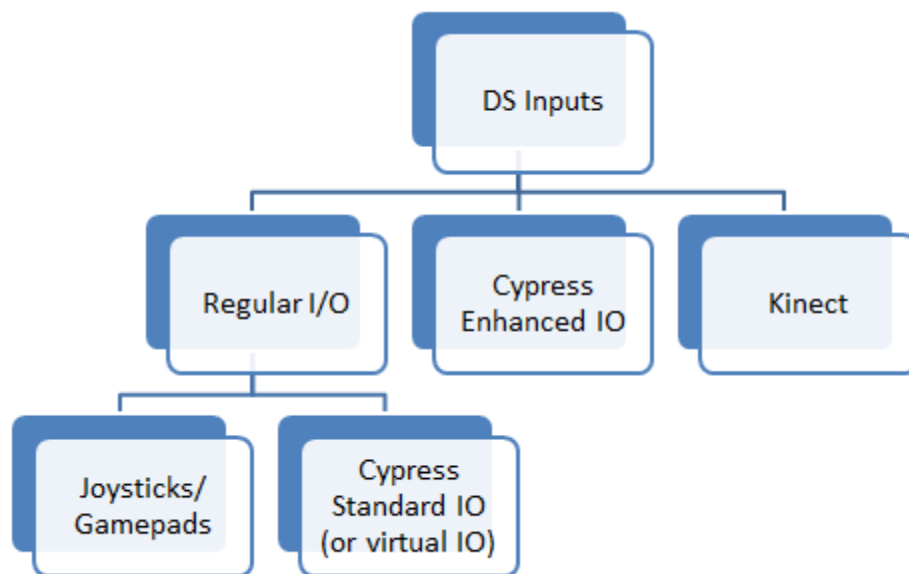


Driver Station Input Overview

The FRC Driver Station software serves as the interface between the human operators and the robot. The software takes input from a number of sources and forwards it to the robot where the robot code can act on it to control mechanisms.

Input types



The chart above shows the different types of inputs that may be transmitted by the DS software. The most common input is an HID compliant joystick or gamepad such as the Logitech Attack 3 or Logitech Extreme 3D Pro joysticks which have been provided in the Kit of Parts since 2009. In addition to these devices teams can also use the Cypress FirstTouch board to design custom IO solutions such as buttons potentiometers or other custom input. This custom IO can then be accessed using either the standard IO methods of the Driver Station class or by using the Enhanced IO Class if additional customization or advanced features are required. Note that a number of devices are now available which allow custom IO to be exposed as a standard USB HID device such as the [E-Stop Robotics CCI](#) or the [U-HID](#) device.

Driver Station Input Overview

Driver Station Class

```
DriverStation ds = DriverStation.getInstance();  
ds.someMethod();  
  
DriverStation.getInstance().someMethod();
```

The Driver Station class has methods for reading all of that "Regular I/O" as well as additional methods to access other information such as the robot mode, battery voltage, alliance color and team number. Note that while the Driver Station class has methods for accessing the joystick data, there is another class "Joystick" that provides a much more user friendly interface to this data. The DriverStation class is constructed as a singleton by the base class. To get access to the methods of the DriverStation object constructed by the base class, call DriverStation.getInstance() and either store the result in a DriverStation object (if using a lot) or call the method on the instance directly.

Robot Mode

```
exampleBool = isDisabled();  
exampleBool = isEnabled();  
  
exampleBool = isAutonomous();  
exampleBool = isOperatorControl();  
exampleBool = isTest();  
  
while(isOperatorControl() && isEnabled())  
{  
  
}
```

The Driver Station class provides a number of methods for checking the current mode of the robot, these methods are most commonly used to control program flow when using the [SimpleRobot base class](#). There are two separate pieces of information that define the current mode, the enable state (enabled/disabled) and the control state (autonomous, operator control, test). This means that exactly one method from the first group and one method from the second group should always return true. For example, if the Driver Station is currently set to Test mode and the robot is disabled the methods isDisabled() and isTest() would both return true.

Battery Voltage

```
voltage = DriverStation.getInstance().getBatteryVoltage();
```

In order to report the robot battery voltage back to the Driver Station software the DriverStation class runs a task which is constantly measuring and updating the battery voltage using the Analog

Driver Station Input Overview

Breakout with the jumper installed on the 9201 module in slot 1 of the cRIO. This information can be queried from the DriverStation class in order to perform voltage compensation or actively manage robot power draw by detecting battery voltage dips and shutting off or limiting non-critical mechanisms,

Alliance

```
DriverStation.Alliance color;
color = DriverStation.getInstance().getAlliance();
if(color == DriverStation.Alliance.kBlue){
}
```

The DriverStation class can provide information on what alliance color the robot is. When connected to FMS this is the alliance color communicated to the DS by the field. When not connected, the alliance color is determined by the Team Station dropdown box on the Operation tab of the DS software.

Location

```
int station;
station = DriverStation.getInstance().getLocation();
```

The getLocation() method of the Driver Station returns an integer indicating which station number the Driver Station is in (1-3). Note that the station that the DS and controls are located in is not typically related to the starting position of the robot so this information may be of limited use. When not connected to the FMS software this state is determined by the Team Station dropdown on the DS Operation tab.

Team Number

```
int team;
team = DriverStation.getInstance().getTeamNumber();
```

The getTeamNumber method returns an integer indicating the FRC team number the Driver Station software is currently set to. One example of using this information would be to distinguish at runtime between multiple robots with identical code but different constants/tuning parameters.

Match Time

```
double time;
time = DriverStation.getInstance().getMatchTime();
```

Driver Station Input Overview

This method returns the approximate match time in seconds. Note that this time is derived by starting a timer at 0 when the enable signal is received for Autonomous and setting the timer to 15 seconds when the enable signal is received for Teleop. This is not an official time sent from the FMS. Another consequence of this is that if the controller reboots or disconnects from the DS during the match, then reconnects, this time will be incorrect.

Custom IO Methods

The DriverStation class also contains methods for accessing custom IO on the Cypress FirstTouch board in compatibility mode. If a Cypress board is not connected to the DS these inputs can be used as virtual IO and set with the keyboard and mouse inside the Driver Station software on the I/O tab. Additional information on accessing this data can be found in the [Custom I/O article](#).