

Using the motor safety feature

Motor Safety is a mechanism in WPILib that takes the concept of a watchdog and breaks it out into one watchdog (Motor Safety timer) for each individual actuator. Note that this protection mechanism is in addition to the System Watchdog which is controlled by the Network Communications code and the FPGA and will disable all actuator outputs if it does not receive a valid data packet for 125ms.

Motor Safety Purpose

The purpose of the Motor Safety mechanism is the same as the purpose of a watchdog timer, to disable mechanisms which may cause harm to themselves, people or property if the code locks up and does not properly update the actuator output. Motor Safety breaks this concept out on a per actuator basis so that you can appropriately determine where it is necessary and where it is not. Examples of mechanisms that should have motor safety enabled are systems like drive trains and arms. If these systems get latched on a particular value they could cause damage to their environment or themselves. An example of a mechanism that may not need motor safety is a spinning flywheel for a shooter. If this mechanism gets latched on a particular value it will simply continue spinning until the robot is disabled. By default Motor Safety is enabled for RobotDrive objects and disabled for all other speed controllers and servos.

Motor Safety Operation

The Motor Safety feature operates by maintaining a timer that tracks how long it has been since the feed() method has been called for that actuator. Code in the Driver Station class initiates a comparison of these timers to the timeout values for any actuator with safety enabled every 5 received packets (100ms nominal). The set() methods of each speed controller class and the set() and setAngle() methods of the servo class call feed() to indicate that the output of the actuator has been updated.

Enabling/Disabling Motor Safety

C++

```
exampleJaguar->SetSafetyEnabled(true);  
exampleJaguar->SetSafetyEnabled(false);
```

Using the motor safety feature

Java

```
exampleJaguar.setSafetyEnabled(true);  
exampleJaguar.setSafetyEnabled(false);
```

Motor safety can be enabled or disabled on a given actuator, potentially even dynamically within a program. However, if you determine a mechanism should be protected by motor safety, it is likely that it should be protected all the time.

Configuring the Safety timeout

C++

```
exampleJaguar->SetExpiration(.1);
```

Java

```
exampleJaguar.setExpiration(.1);
```

Depending on the mechanism and the structure of your program, you may wish to configure the timeout length of the motor safety (in seconds). The timeout length is configured on a per actuator basis and is not a global setting. The default (and minimum useful) value is 100ms.