Common Command Functions

Here we have listed some of the most basic and common functions we use when defining commands to control the robot. This is not a comprehensive list. You can read more about this API and discover more functions at the links below.

- The full API documentation for the `Command` class
- The full API documentation for the `Commands` utility class
- The full API documentation for the `Trigger` class
- WPILib docs on more command compositions
- WPILib docs on structuring command based projects

Base command functions

These functions work great as a base for a sequence of actions you want to define.

- `run(action)`
  - Repeatedly runs an action until interrupted
- `runEnd(runAction, endAction)`
  - Repeatedly runs an action until interrupted, then runs a second action
- `runOnce(action)`
  - Runs an action once. that's it
- `startEnd(startAction, endAction)`
  - Runs an action once and another action when interrupted

Examples

> A function that takes in a number representing the power to run the arm motor at. When the command ends, the motor will stop. **Note** that there is nothing actually specifying *when* the command ends, only *what* happens when it does.

```
// ArmSubsystem.java
public Command moveAndStop(double power) {
    return runEnd(() -> armKraken.setControl(new DutyCycleOut(power)), armKraken::stopMotor);
}
```

Modifier command functions

These functions work great to enhance a command's functionality. They can refine when a command ends and join a command with others.

- `andThen(command)`
  - called on a command to run another command when the first one finishes
- `alongWith(command)`
  - called on a command to run another command in parallel with the first one
- `until(condition)`
  - called on a command to end it when the condition is true
- `withTimeout(seconds)`
  - called on a command to end it after a specified number of seconds

Example

> We have modified the function from the previous example with a timeout. Now we have specified *when* the command should end. In this case, after two seconds.

```
public Command moveAndStop(double power) {
    return runEnd(() -> armKraken.setControl(new DutyCycleOut(power)), armKraken::stopMotor).withTimeout(2);
}
```

Binding commands to buttons

These functions use the `Trigger` api to start commands when some condition is true. The most common use case is starting a command when a button is pressed. You can read more about triggers here.

- `onTrue(condition)`
  - Starts a command when the condition changes from false to true
- `whileTrue(condition)`
  - Starts a command when the condition changes from false to true, and then cancels the command when the condition changes back to false. You can use this when you want a command to run only while holding a button.
- `onFalse(condition)`
  - Starts a command when the condition changes from true to false
- `whileFalse(condition)`
  - Starts a command when the condition changes from true to false, and then cancels the command when the condition changes back to true

Example

> Line 3 of this example shows how to run the command we made in the previous example when button 1 on the joystick is pressed. Since we pass in 0.5 into the `moveAndStop` method, the arm will move at 50% power. The command will then

> stop after 2 seconds just as we defined it. **Note** that the `configureBindings` method is one that is preexisting in `RobotContainer`. We do not make this for every binding.

```
// RobotContainer.java
private void configureBindings() {
    joystick.button(1).onTrue(arm.moveAndStop(0.5));
}
```

Revision #3
Created 23 July 2025 00:26:15 by Sam Perlmutter
Updated 23 July 2025 01:46:10 by Sam Perlmutter