

Default Commands

In some cases you may have a subsystem which you want to always be running a command no matter what. So what do you do when the command you are currently running ends? That's where default commands come in.

What is the default command?

Each subsystem may, but is not required to, have a default command which is scheduled whenever the subsystem is idle (the command currently requiring the system completes). The most common example of a default command is a command for the drivetrain that implements the normal joystick control. This command may be interrupted by other commands for specific maneuvers ("precision mode", automatic alignment/targeting, etc.) but after any command requiring the drivetrain completes the joystick command would be scheduled again.

Setting the default command

C++

```
#include "ExampleSubsystem.h"

ExampleSubsystem::ExampleSubsystem()
{
    // Put methods for controlling this subsystem
    // here. Call these from Commands.
}

ExampleSubsystem::InitDefaultCommand()
{
    // Set the default command for a subsystem here.
    SetDefaultCommand(new MyDefaultCommand());
}
```

Default Commands

Java

```
public class ExampleSubsystem extends Subsystem {

    // Put methods for controlling this subsystem
    // here. Call these from Commands.

    public void initDefaultCommand() {
        // Set the default command for a subsystem here.
        setDefaultCommand(new MyDefaultCommand());
    }

}
```

All subsystems should contain a method called `initDefaultCommand()` which is where you will set the default command if desired. If you do not wish to have a default command, simply leave this method blank. If you do wish to set a default command, call `setDefaultCommand` from within this method, passing in the command to be set as the default.