

Choosing a Base Class

The base class is the framework that the robot code is constructed on top of. WPILib offers two different base classes, as well as a third option which is not technically a separate base class.

Simple Robot

```
public class RobotTemplate extends SimpleRobot {  
    /**  
     * This function is called once each time the robot enters autonomous mode.  
     */  
    public void autonomous() {  
        while(isAutonomous() && isEnabled())  
        {  
            //Put code here  
            Timer.delay(.05);  
        }  
    }  
  
    /**  
     * This function is called once each time the robot enters operator control.  
     */  
    public void operatorControl() {  
        while(isOperatorControl() && isEnabled())  
        {  
            //Put code here  
            Timer.delay(.05);  
        }  
    }  
}
```

The SimpleRobot class is the simplest to understand as most of the state flow is directly visible in your program. Your robot program overrides the operatorControl() and autonomous() methods that are called by the base at the appropriate time. Note that these methods are called only once each time the robot enters the appropriate mode and are not automatically terminated. Your code in the operatorControl method must contain a loop that checks the robot mode in order to keep running and taking new input from the Driver Station. The autonomous code shown uses a similar loop.

Choosing a Base Class

Iterative Robot

```
public class RobotTemplate extends IterativeRobot {  
    /**  
     * This function is run when the robot is first started up and should be  
     * used for any initialization code.  
     */  
    public void robotInit() {  
  
    }  
  
    /**  
     * This function is called once each time the robot enters autonomous  
     */  
    public void autonomousInit() {  
  
    }  
  
    /**  
     * This function is called periodically during autonomous  
     */  
    public void autonomousPeriodic() {  
  
    }  
}
```

The Iterative Robot base class assists with the most common code structure by handling the state transitions and looping in the base class instead of in the robot code. For each state (autonomous, teleop, disabled, test) there are two methods that are called:

- Init methods - The init method for a given state is called each time the corresponding state is entered (for example, a transition from disabled to teleop will call teleopInit()). Any initialization code or resetting of variables between modes should be placed here.
- Periodic methods - The periodic method for a given state is called each time the robot receives a Driver Station packet in the corresponding state, approximately every 20ms. This means that all of the code placed in each periodic method should finish executing in 20ms or less. The idea is to put code here that gets values from the driver station and updates the motors. You can read the joysticks and other driverstation inputs more often, but you'll only get the previous value until a new update is received. By synchronizing with the received updates your program will put less of a load on the cRIO CPU leaving more time for other tasks such as camera processing.

Choosing a Base Class

Command Based Robot

While not strictly a base class, the Command based robot model is a method for creating larger programs, more easily, that are easier to extend. There is built in support with a number of classes to make it easy to design your robot, build subsystems, and control interactions between the robot and the operator interface. In addition it provides a simple mechanism for writing autonomous programs. The command based model is described in detail in the [Command Based Programming manual](#).