

Operator Tutorial

Hello and welcome to team 1073's Operator control tutorial. This tutorial will explain to you the basic parts of our main operator control file. Let us begin.

```
#include "main.h"
#include "utils.h"

void
MainRobot::OperatorControl ()
{

    // Set for operator mode control
    genesisRobot->SetDriverMode(true);    // Set for operator mode control
```

The previous code has several aspects. The two #includes at the top call in other header files, main.h, which has all of our motor and joystick definitions as well as other header files, and utils.h, which contains most of our constants and port locations.

In "void MainRobot::OperatorControl()", "MainRobot" is the class name, and OperatorControl () is the member function. The void means that we are not returning anything.

Next is "genesisRobot->SetDriverMode(true);". In the main.h we declared genesisRobot and in main.cpp we assigned it to a new GenesisDrive, (our epic drive system, sort of like Tank or ArcadeDrive), process. We use a "->" because it dereferences the pointer and assigns it to a class.

```
while (1)
{
    Wait(.010);

    if(leftStick->GetButton(Joystick::kTopButton)) {
        SetNextResolution();
    }

    if(rightStick->GetButton(Joystick::kTopButton)) {
        ScreenShot();
        Wait(.5);
    }

    genesisRobot->PeriodicService();

    ballLaunchHandler->OperateBallCollector();
```

The While (1) keeps the loop going as long as the robot is activated, once the function is called. We wait 10 milliseconds to prevent the loop from cascading out of control.

The next two if statements are getting button values from previously defined joysticks using the standard joystick.h. we tell it to get a button value, we then pass it in either a previously defined standard button, as we did, or you can pass it a button number as specified in joystick.h.

“genesisRobot->PeriodicService()” refers to the PeriodicService member function in GenesisDrive.cpp. In that process, the joystick x and y values are refreshed, and motors speeds reset depending on these new joystick values.

“ballLaunchHandler->OperateBallCollector()” is similar to the above mentioned line. OperateBallCollector is effectively the “PeriodicService” for the ball handler. It resets the ball launch motors depending on whether or not a certain button on a certain joystick is pressed.

Thank you for reading team 1073’s software tutorial, we hoped you learned something, and good luck in coding and at comp!