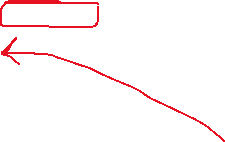
*Prerequisites to beginning to code using the latest version*

1. Install Visual Studio Code (latest version unless code version mentions otherwise).
2. Go to the “Extensions” tab and install the extension “PROS”. Any issues with installation, please visit <https://pros.cs.purdue.edu/> Subsequently, install the extension “VEX ROBOTICS” (backup).

A screenshot of a computer

Description automatically generated



1. Download all Code related files when making a new project which will be labelled in the format: *Violent\_Explosion\_DDMMYY\_VERSIONNUMBER*

*Any* changes which will be made which need to be explained or additions which need to be explained will be added to the GitHub Repository.

1. Unzip all Code related files in a newly created folder to hold all of it, and open Visual Studio.
2. Navigate to FILE > OPEN FOLDER > to the folder you have just made.
3. Make sure you are connected to an internet source with suitable strength to ensure the following message pops up: “The CLI and Toolchain are up to date”, otherwise the build and upload functionality to the VEX Brain *will not work* (some functions must be downloaded from the internet.

A screenshot of a computer

Description automatically generated

1. Go to explorer (CTRL + SHIFT + E) and navigate to the file name “main.cpp” which is where the bulk of your code editing will happen.

*CHANGING MOTOR AND PERIPHERAL SETTINGS*

A screen shot of a computer program

Description automatically generated

The LemLib Constructor is used to alter the autonomous port assignments for all the peripherals (motors, sensors etc.)

A screenshot of a computer program

Description automatically generated

The EZ Template Constructor is by far the more important one – follow the comments to adjust all the settings it mentions. Play with the settings until you reach the ones which are the best performing and let me know what exactly you have changed.

*CHANGING DRIVER CONTROL SETTINGS*

A computer screen shot of text

Description automatically generated

The function “opcontrol” is controls driver control. The images of the code will contain examples of what is being mentioned for driver control.

The most important things to use to code driver control are:

* Motor Declaration – Adding a new motor? Do this before the while loop and within the function’s curly brackets, in the format (italicized indicates editable sections):

pros::Motor *motorname(port, rotationdirection)*

rotationdirection is limited to Boolean true or false values which indicate direction of rotation of the motor.

* Pneumatic Declaration – Do this before the while loop and within the function’s curly brackets, in the format (italicized indicates editable sections):

pros::ADIDigitalOut *pneumaticname(port, beginningstate)*

beginning state is limited to Boolean true or false values which indicate whether air is added to the pneumatic cylinder or not.

* Motor movement (basic) – Do this during within the limits of the while loop and indented in the format (italicized indicates editable sections):

*motorname*.move(*potential difference*)

The potential difference provided to the motor must be between -127 and 127 (those are the min and max voltage limits which can be provided to motor)

A screen shot of a computer program

Description automatically generated

* Button press structure – Do this during within the limits of the while loop and indented (in the format (italicized indicates editable sections):

if (master.get\_digital\_new\_press(DIGITAL\_*button*)){

*code to be executed upon pressing button.*

      }

* Debugging code – ensure you check everything has been fully closed off with the correct brackets, don’t forget to include semicolons and generally abide by C++ syntax. Make sure there are no typos or similar as these can also trigger errors
* PROS and EZ Template further list of important functions:

<https://pros.cs.purdue.edu/>