VEX C++ Autonomous Blocks

Created by Ben Cornelius and Orin Selby

What are they?

The VEX C++ Autonomous Blocks (VCABs) are a series of coding segments designed to carry out basic autonomous robot functions. Their purpose is to shorten the amount of time writing a VEX C++ program takes. They also appear on the controller screen as they are carried out!

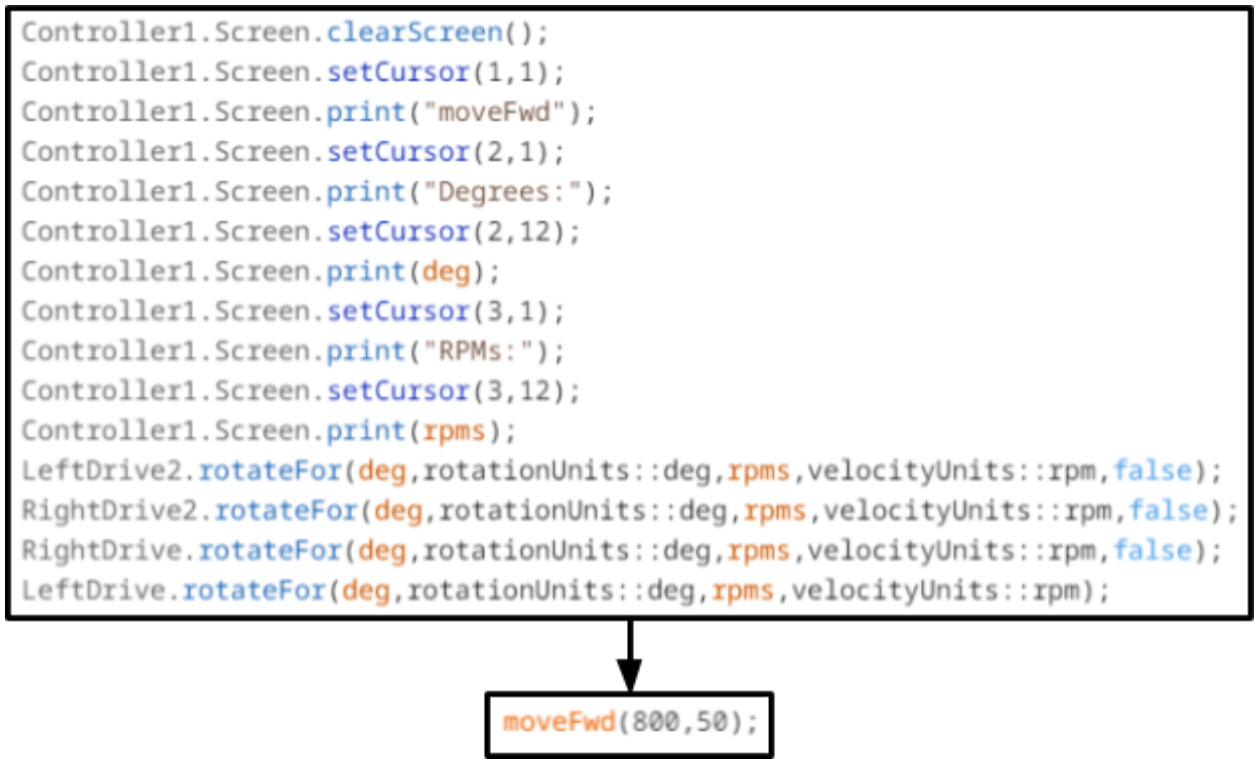

Figure 1.1: *A group of 15 commands was shortened to one simple callback function with two parameters:* deg *and* rpms.

How do I get access to them?

1) Download all files
2) Open the file (either the Competition Template or just VCABs)
3) Write your autonomous program!

Is there any formatting I need to worry about?

Yes, you will need to do the following:

- Name your drive motors as follows:
    - LeftDrive (your front left motor)
    - RightDrive (your front right motor)
    - RightDrive2 (your rear right motor)
        - Only use if your robot is 4-wheel-drive
    - LeftDrive2 (your rear left motor)
        - Only use if your robot is 4-wheel-drive
    - *Port numbers don't matter; VCS uses motor names for commands, rather than port numbers.*

- If your robot is 4-wheel-drive, you will need to uncomment (delete "//" before the command) the LeftDrive2 and RightDrive2 commands for each set of blocks
    - If your robot is 2-wheel-drive, leave the statements commented
- These blocks utilize the "rotateFor();" command, so the deg value is independent of other blocks. This means that your robot will move for the amount of degrees you specify, rather than moving to a specific degree value.
    - Do not input negative values for Fwd or Bwd functions. The moveBwd() and blockBwd() functions automatically negate the values you provide, meaning that inputting negative values will cause the robot to move forward for these functions.
        - Negative turns are fine.

What are the blocks?

moveFwd(deg, rpms);

This block will move your robot forward for the given amount of degrees (deg) at the given speed (rpms)

moveBwd(deg, rpms);

This block will move your robot backward for the given amount of degrees (deg) at the given speed (rpms)

zRight(deg, rpms);

This block will make your robot perform a zero-point, right turn for the given amount of degrees (deg) at the given speed (rpms)
You can use the variable nDegZ in place of deg to perform 90-degree zero-point turns, if you have tuned nDegZ to your robot

zLeft(deg, rpms);

This block will make your robot perform a zero-point, right turn for the given amount of degrees (deg) at the given speed (rpms)
You can use the variable nDegZ in place of deg to perform 90-degree zero-point turns, if you have tuned nDegZ to or your robot

wideRight(deg, rpms);

This block will make your robot perform a wide right turn (left wheels move forward, right wheels lock) for the given amount of degrees (deg) at the given speed (rpms).
You can use the variable nDegW in place of deg to perform 90-degree wide turns, if you have tuned nDegW to your robot

wideRight(deg, rpms);

This block will make your robot perform a wide left turn (right wheels move forward, left wheels lock) for the given amount of degrees (deg) at the given speed (rpms).
You can use the variable nDegW in place of deg to perform 90-degree wide turns, if you have tuned nDegW to your robot

blockFwd(rpms);

This block will move your robot forward for one block/field tile (oneBlock) at the given speed (rpms)
You will need to change the variable oneBlock if you are *not* using 4-inch wheels (5-inch wheels = 812 deg) (3.25-inch wheels = 528 deg)

blockBwd(rpms);

This block will move your robot backward for one block/field tile (oneBlock) at the given speed (rpms)
You will need to change the variable oneBlock if you are *not* using 4-inch wheels (5-inch wheels = 812 deg) (3.25-inch wheels = 528 deg)

How do I use them?

Using the blocks is simple: call the function, and input the parameters. For example, to make your robot move forward for 800 degrees at 200 rpms, you would write: moveFwd(800,200);

The maximum rpm values for the different motor cartridges are:

- High Speed: 600rpms
- Regular: 200rpms
- High Torque: 100rpms

Note: Make sure you format your motors in Robot Settings!

How do I get the degree values?

To get degree values from motors, you *could* go into the Brain settings and access the motor info, but that would be a tedious task when you have a complex autonomous.
Instead, we created a program that shows you the degree values on the controller with the simple press of a button!
To get this program:

1) Open “Values Program.vex”
2) Change the motor ports in the template to match the ports your motors use
3) Download the program to your bot in an empty program slot

How do I use the Values Program?

The Values program displays the rotation (in deg) of different motors when you press different buttons. Here are the defaults:

- **LeftDrive** (*port2*): **ButtonA**
- **RightDrive** (*port20*): **ButtonX**
- **LeftDrive2** (*port19*): **ButtonB**
- **RightDrive2** (*port3*): **ButtonY**

The unused buttons are ButtonUp, ButtonDown, ButtonLeft, and ButtonRight.
To add another motor to the Values program, use this format:

```
if(Controller1.Button[up/down/left/right].pressing())
{
    Controller1.Screen.clearScreen();
    Controller1.Screen.setCursor(1,1);
    Controller1.Screen.print("[motorName]:");
    Controller1.Screen.setCursor(2,1);
    Controller1.Screen.print("%f degrees",[motorName].rotation(rotationUnits::deg));
}
```