Getting Started with ROBOTC – Creating a program for competition

STEP 1 - Creating a new program

Go to the File menu, click New... and then choose "New Competition Template"
Now go to the Robot menu, click Platform Type and make sure that "VEX 2.0 Cortex" is selected.
Finally, go to the Robot menu, click VEX Cortex Communication Mode and ensure "Competition (VEXnet)" is selected.

We are now ready to start the first program but before we start coding, it is important to understand the structure of a VEX Competition program.

There are three main sections to the program which are described below. As you scroll down through the Competition Template program, you will see these three sections. At an official VEX Competition, the competition field controls which mode your robot is in so that all teams start and stop at exactly the same time.

Pre-Autonomous
This code runs as soon as the robot is turned on. It is used to initialise sensors and other parts of the robot. It is not something that you will use with your first programs but as you become more advanced, it becomes a vital part of a successful VEX competition robot.
Pre-Autonomous code runs even when the robot is in "Disable" mode – this allows initialisation to occur before a match begins.

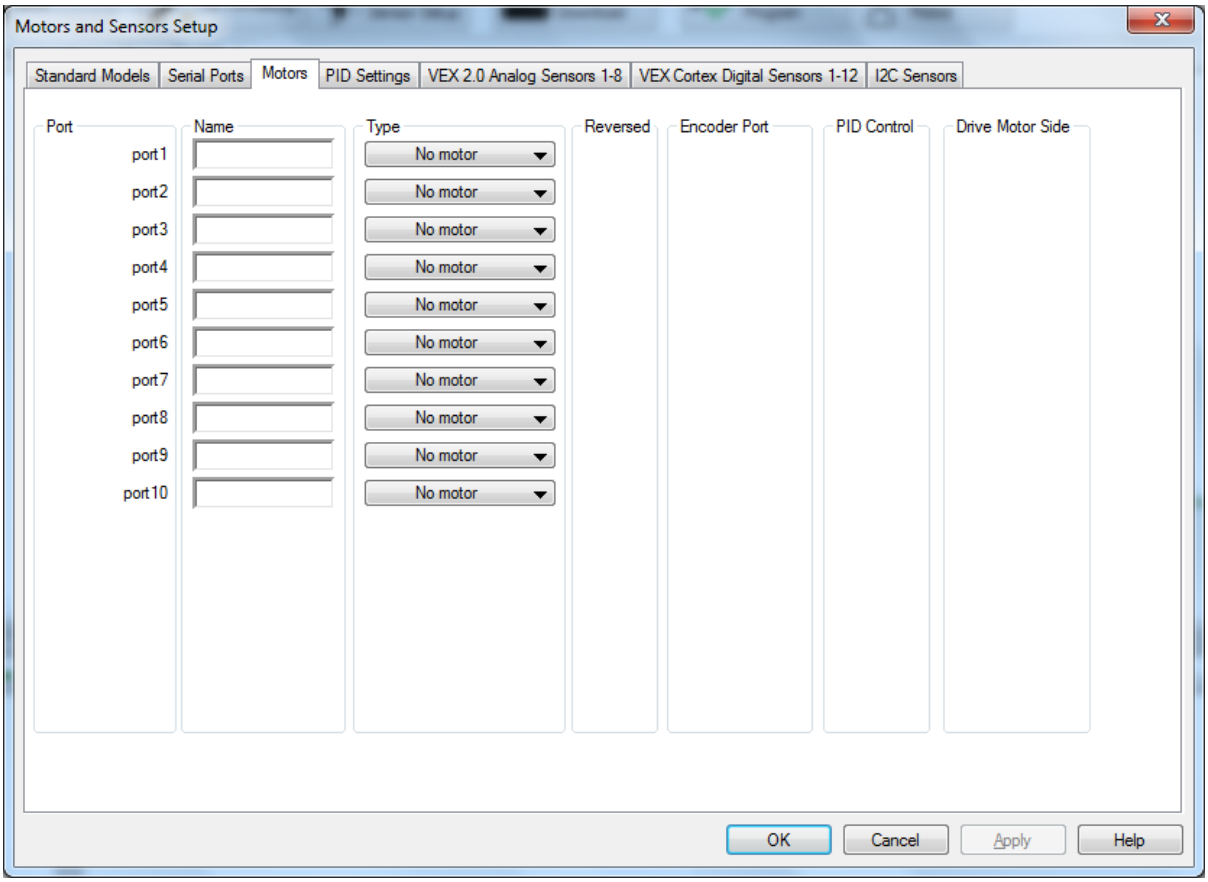
Autonomous
This code is executed when the robot is in both Autonomous and Enable mode. As the name suggests, the joystick functions cannot be used in Autonomous as the robot is operating by itself. If the robot is in Disable mode, the Autonomous code cannot run.

User Control
This code is executed during the User Control period (usually referred to as Driver Control) where you can use the joystick to control your robot as well as any sensors etc. that you have used. The robot must be in Enable mode for User Control to work.

STEP 2 – Install the ROBOTC firmware on the Cortex
You should have already updated the Firmware on your Cortex, Joystick and VEXnet 2.0 keys. The last thing we need to do is load the ROBOTC Firmware onto the Cortex to allow it to understand ROBOTC programs.
To do this, connect the Cortex to your USB port and click the Firmware Download button at the top of ROBOTC. This takes a few seconds to complete and remains on the Cortex. The only time you will need to re-download the ROBOTC Firmware is if you update the Master Firmware on the Cortex.

STEP 3 – Configure your motors and sensors

Click on Motor and Sensor Setup at the top of the screen. This dialogue box is where you can name your various motors and sensors. Click on the Motors tab, it should look something like this:

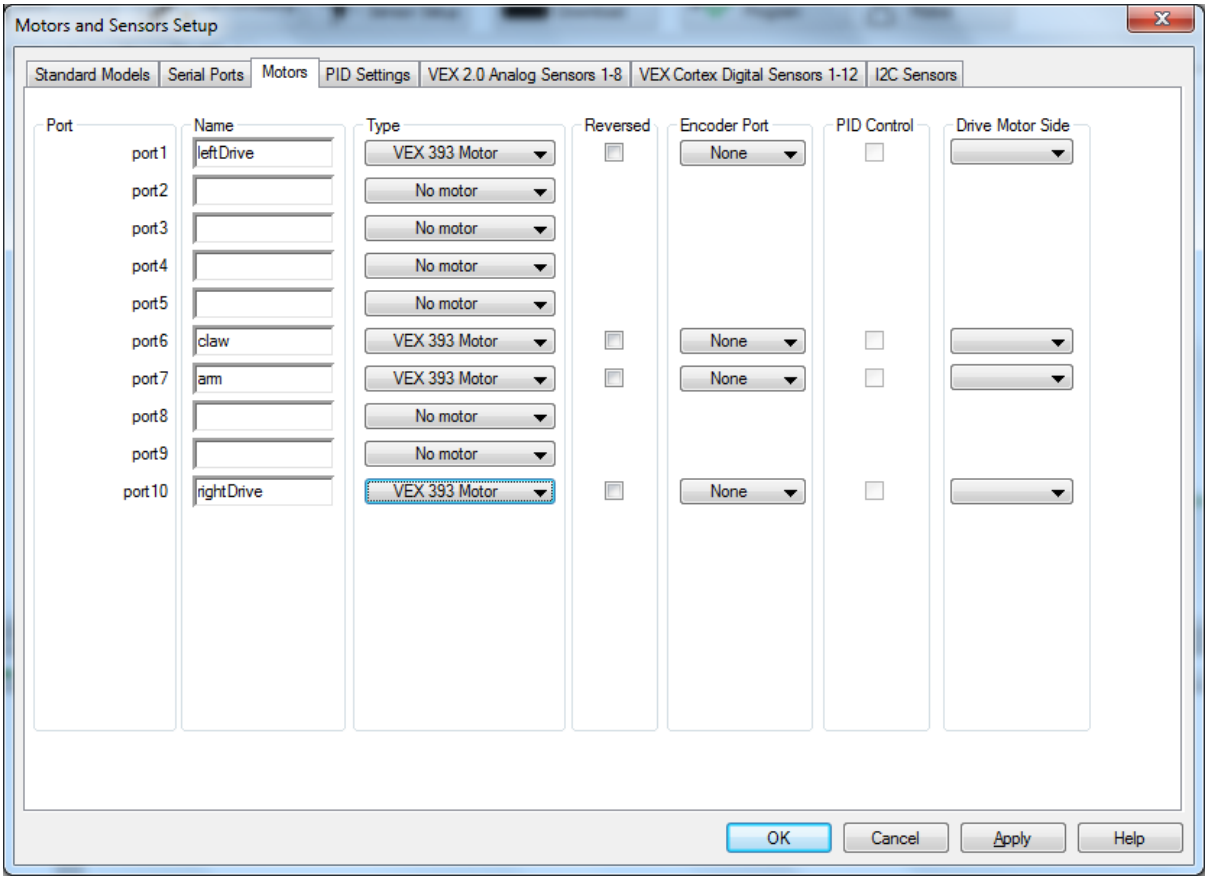


For the standard Clawbot, we have 4 motors in total. We are going to give them names here so we can refer to them by these names within the program. This is case sensitive so take that into account. Also, don't use spaces or symbols in your names. If you have built the Clawbot as per the instructions, your motors will be connected to the ports shown in the table below. The ROBOTC name column is what we are going to call that motor in ROBOTC.

Motor	Cortex Port	ROBOTC Name
Left Wheels	1	leftDrive
Right Wheels	10	rightDrive
Arm	7	arm
Claw	6	claw

Name the motors accordingly and choose VEX 393 Motor from the Type drop down list.

It should now look something like this:



Click OK to return to the program. You will notice that ROBOTC has created 4 #pragma statements at the top of the screen, don't edit these.

STEP 4 – Creating your first Program

Now we are going to make the robot move forward in Autonomous mode. Scroll down to around line 35 until you find this section:

```
/////////////////////////////////////////////////////////////////////////////////////////
//
//                          Autonomous Task
//
// This task is used to control your robot during the autonomous phase of a VEX Competition.
// You must modify the code to add your own robot specific commands here.
//
/////////////////////////////////////////////////////////////////////////////////////////

task autonomous()
{
  // ..........................................................................
  // Insert user code here.
  // ..........................................................................

        AutonomousCodePlaceholderForTesting();  // Remove this function call once you have "real" code.
}
```

Any lines preceded by // turn green – these are comment lines and are ignored in the program. Comments help you add notes to your code to make it easier to remember what you have done.

Re-write the code so that it looks like this:

```
/////////////////////////////////////////////////////////////////////////////////////////
//
//                          Autonomous Task
//
// This task is used to control your robot during the autonomous phase of a VEX Competition.
// You must modify the code to add your own robot specific commands here.
//
/////////////////////////////////////////////////////////////////////////////////////////

task autonomous()
{
        motor[leftDrive] = 127; //set the left drive motor to full speed forward
        motor[rightDrive] = 127; //set the right drive motor to full speed forward
        wait1Msec(2000);  //wait for 2000 milliseconds, or 2 seconds
        motor[leftDrive] = 0; //stop the left drive motor
        motor[rightDrive] = 0; //stop the right drive motor
}
```

The motor command allows us to set the power of a motor. The name of the motor we want to set is in the square brackets. Remember, it is case sensitive so make sure you get the name correct. The power value can be anything from -127 (full speed backwards) through 0 (stopped) up to 127 (full speed forwards)
The wait1Msec command makes the program pause at that line for the time specified inside the brackets. The time is specified in milliseconds.

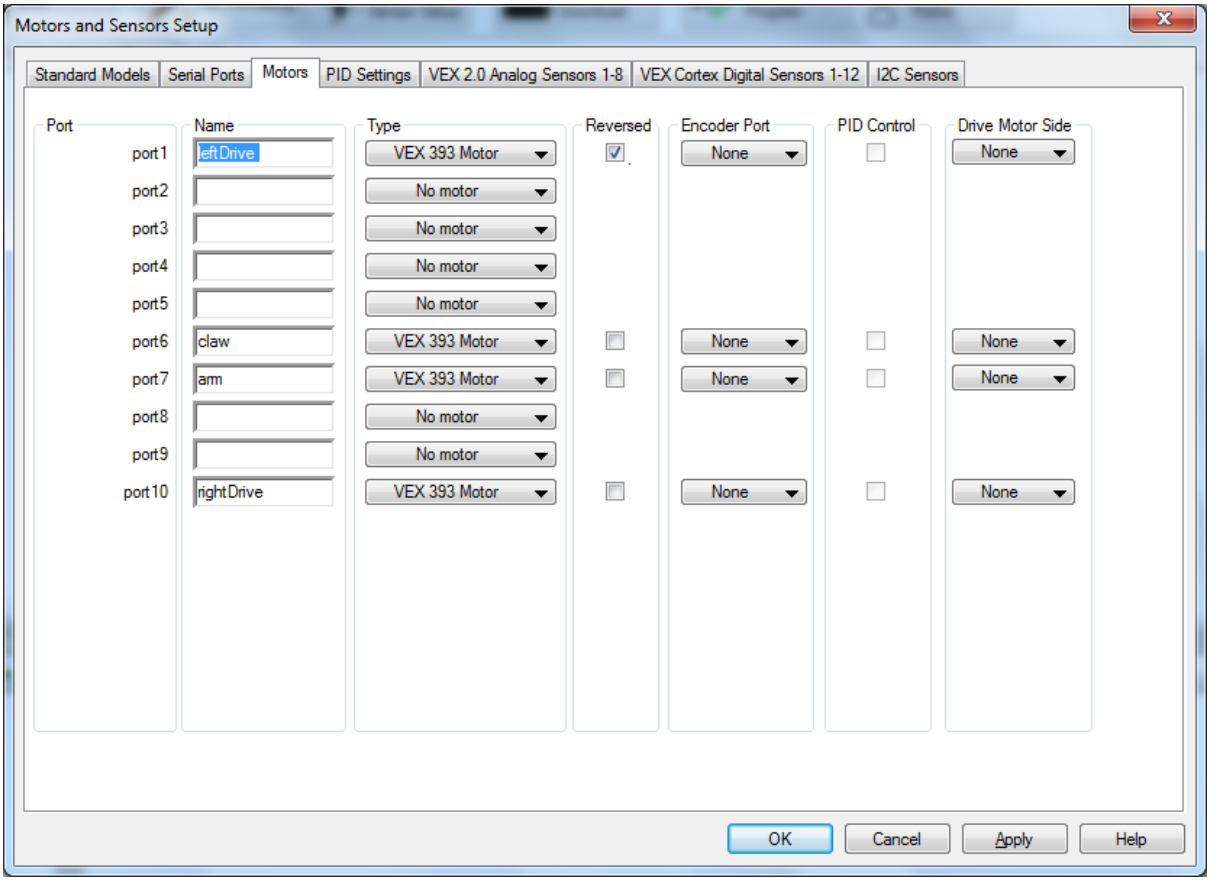
So this program will switch both motors on and 2 seconds later, will switch them off again.

Save your program and then download the program to the Cortex by connecting the USB cable and clicking on the Download to Robot button.

Now it is time to test the program! Disconnect the USB cable and plug the VEXnet Key into your Cortex. Make sure the other VEXnet Key is plugged in to your joystick.
Now connect your VEXnet Competition Switch to the Competition Port on the top of the Joystick and check that the Enable/Disable switch is set to Disable and the Driver/Autonomous switch is set to Autonomous.
Place the robot on the floor and switch on both your robot and your joystick and wait for them to connect to each other. Once they are connected, switch the VEXnet Competition Switch to Enable and your robot should start to move.
Depending on how your robot is wired, you might find that it goes forwards, backwards or spins on the spot (i.e. one motor is going forward and one is going backwards) as per the below table:

Robot movement	**Cause**
Forward	Both motors are set up correctly, all is fine!
Backwards	Both motors are reversed
Spins right (clockwise)	The right motor is reversed
Spins left (anti-clockwise)	The left motor is reversed

To fix this, click on the Motors and Sensors Setup box and tick the Reversed box for the relevant motor. For example, if your robot was rotating left, you will need to reverse the left motor as below:



Repeat the download procedure but be sure to set your VEXnet Switch to Disable before you switch the robot back on so that it does not move until you want it to.
Your robot should now go forward for 2 seconds and then stop.

STEP 4 – Adding Driver Control

Now we are going to implement some Driver Control joystick commands. Scroll down to the User Control section that looks like this:

```
/////////////////////////////////////////////////////////////////////////////////////////
//
//                          User Control Task
//
// This task is used to control your robot during the user control phase of a VEX Competition.
// You must modify the code to add your own robot specific commands here.
//
/////////////////////////////////////////////////////////////////////////////////////////

task usercontrol()
{
        // User control code here, inside the loop

        while (true)
        {
         // This is the main execution loop for the user control program. Each time through the loop
         // your program should update motor + servo values based on feedback from the joysticks.

         // ........................................................................
         // Insert user code here. This is where you use the joystick values to update your motors, etc.
         // ........................................................................

         UserControlCodePlaceholderForTesting(); // Remove this function call once you have "real" code.
        }
}
```

We are now going to add some commands to control the robot using the joysticks. First, it is important to understand the button and axis on the joystick and what they are called.
The analogue joysticks have 2 axis each (up/down and left/right). A legend is printed on the joystick to show a number for each axis

Axis	Channel number	Value returned
Right Joystick Up/Down	2	Up = 1 to 127 Down = -1 to -127 Centre = 0
Right Joystick Left/Right	1	Right = 1 to 127 Left = -1 to -127 Centre = 0
Left Joystick Up/Down	3	Up = 1 to 127 Down = -1 to -127 Centre = 0
Left Joystick Left/Right	4	Right = 1 to 127 Left = -1 to -127 Centre = 0

Now change the code to look like this:

```
/////////////////////////////////////////////////////////////////////////////////////////
//
//                          User Control Task
//
// This task is used to control your robot during the user control phase of a VEX Competition.
// You must modify the code to add your own robot specific commands here.
//
/////////////////////////////////////////////////////////////////////////////////////////

task usercontrol()
{
        // User control code here, inside the loop

        while (true)
        {
                motor[leftDrive] = vexRT[Ch2];
                motor[rightDrive] = vexRT[Ch1];
        }
}
```

This program will set the left motor to the value of the left joystick up/down (Ch2) and the right motor will be set to the value of the right joystick up/down (Ch1).
while(true) is an infinite loop, so these values will constantly be updated.

Download this to the robot. Use the competition switch to try the autonomous program and then switch to Driver to see if your joystick controls work.

And there you have the basis of a competition program! To see examples of how to code other functions, click on File and then Open Sample Program
Try to increase the complexity of the Autonomous program and add controls for the Arm and Claw into your Operator Control Program.