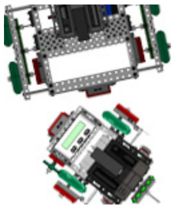
ROBOTC Natural Language - VEX PIC Reference:

Setup Functions:

Robot Type
Choose which robot you want to write a program for. Note that not including this command defaults to "`robotType(none);`" Also please note that this command should be the first thing in your "`task main()`".



Command:

```
robotType(type);
```

Parameters: `type`

Valid Robot Types for `type`:
`none` - this will not set up any motors and sensors for you (this is the default.)
`squarebot` - sets the motors to match a default Squarebot (NO sensors will be setup).

Usage without Parameters:

```
robotType();
```

This snippet of code will set the robot type to `none` by default, skipping the setup process. You must manually set the motors and sensors in the 'Motors and Sensors Setup' menu.

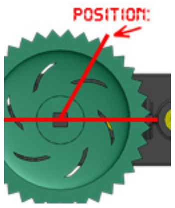
Usage with Parameters:

```
robotType(squarebot);
```

This snippet of code will set the robot type to `squarebot`. This will automatically set up the motor ports to match those of a default Squarebot.
(Note that no sensors are ever setup for the VEX PIC.)

Movement Functions:

Set Servo
Set a servo to a desired position.



Command:

```
setServo(servo, position);
```

Parameters: `servo`, `position`

Acceptable Motors for `servo`:
MOTOR ports 1 through 8 (and your names for them given in Motors and Sensors Setup.)

Valid Range Values for `position`:
-127 to 127.

Usage without Parameters:

```
setServo();
```

This snippet of code will set the servo on motor-port 8 to position 0 (center). The default motor-port is `port8` and the default position is `0` for `setServo()`.

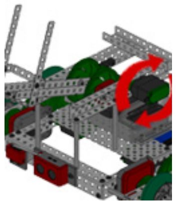
Usage with Parameters:

```
setServo(port7, 37);
```

This snippet of code will set the servo on motor-port 7 to position 37.

ROBOTC Natural Language - VEX PIC Reference:

Start Motor
Set a motor to a speed.



Command:

```
startMotor(motor, speed);
```

Parameters: `motor`, `speed`

Acceptable Motors for `motor`:
MOTOR ports 1 through 8 (and your names for them given in Motors and Sensors Setup.)

Valid Range Values for `speed`:
-127 (reverse) to 127 (forward) where 0 is stop.

Usage without Parameters:

```
startMotor();
wait();
stopMotor();
```

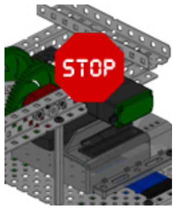
This snippet of code will run the motor in motor-port 6 at speed 95 for 1.0 seconds and then stop it. The default motor-port is `port6` and the default speed is `95` for `startMotor()`.

Usage with Parameters:

```
startMotor(port8, -32);
wait(0.5);
stopMotor(port8);
```

This snippet of code will run the motor in motor-port 8 at speed -32 for 0.5 seconds and then stop it.

Stop Motor
Stops a motor.



Command:

```
stopMotor(motor);
```

Parameters: `motor`

Acceptable Motors for `motor`:
MOTOR ports 1 through 8 (and your names for them given in Motors and Sensors Setup.)

Usage without Parameters:

```
startMotor();
wait();
stopMotor();
```

This snippet of code will run the motor in motor-port 6 at speed 95 for 1.0 seconds and then stop it. The default motor-port is `port6` for `stopMotor()`.

Usage with Parameters:

```
startMotor(port8, -32);
wait(0.5);
stopMotor(port8);
```

This snippet of code will run the motor in motor-port 8 at speed -32 for 0.5 seconds and then stop it.

ROBOTC Natural Language - VEX PIC Reference:

Wait Functions:

Wait

Wait an amount of time measured in seconds. The robot continues to do what it was doing during this time.

Command:

```
wait(time);
```

Parameters: `time`

Valid Range Values for `time`**:**
0 to 32766 (Must be whole numbers; VEX PIC does not support decimal "floating point" values.)

Usage without Parameters:

```
forward();
wait();
stop();
```

This snippet of code will run the robot forward for 1 second and then stop. The default time is `1` (second) for `wait()`.

Usage with Parameters:

```
forward(63);
wait(2);
stop();
```

This snippet of code will run the robot forward at half speed for 2 seconds and then stop.

Wait in Milliseconds

Wait an amount of time in milliseconds. The robot continues to do what it was doing during this time.

Command:

```
waitInMilliseconds(time);
```

Parameters: `time`

Valid Range Values for `time`**:**
0 to 32766.

Usage without Parameters:

```
forward();
waitInMilliseconds();
stop();
```

This snippet of code will run the robot forward for 1000 milliseconds (1.0 seconds) and then stop. The default time is `1000` (milliseconds) for `waitInMilliseconds()`.

Usage with Parameters:

```
forward(63);
waitInMilliseconds(2730);
stop();
```

This snippet of code will run the robot forward at half speed for 2730 milliseconds (2.73 seconds) and then stop.

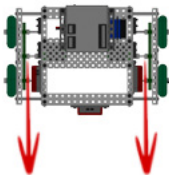
ROBOTC Natural Language - VEX PIC Reference:

Robot Movement Functions:

Note that for desirable results with the following set of functions, you must use the "`robotType();`" Setup Function with either `recbot` or `swervebot` in the beginning of your "`task main()`".

Forward

Both wheels rotate forward at the same speed, causing the robot to move forward.



Command:

```
forward(speed);
```

Parameters: `speed`

Valid Range Values for `speed`**:**
0 to 127 (however `forward()` will always move your robot forward.)

Usage without Parameters:

```
forward();
wait();
stop();
```

This snippet of code will run the robot forward for 1 second and then stop. The default speed is `95` for `forward()`.

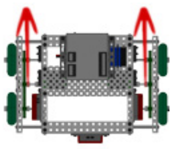
Usage with Parameters:

```
forward(63);
wait(2);
stop();
```

This snippet of code will run the robot forward at half speed for 2 seconds and then stop.

Backward

Both wheels rotate backward at the same speed, causing the robot to move backward.



Command:

```
backward(speed);
```

Parameters: `speed`

Valid Range Values for `speed`**:**
-127 to 0 (however `backward()` will always move your robot backward.)

Usage without Parameters:

```
backward();
wait();
stop();
```

This snippet of code will run the robot backward for 1 second and then stop. The default speed is `-95` for `backward()`.

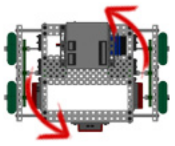
Usage with Parameters:

```
backward(-63);
wait(2);
stop();
```

This snippet of code will run the robot backward at half speed for 2 seconds and then stop.

ROBOTC Natural Language - VEX PIC Reference:

Point Turn
Both wheels rotate at the same speed but in opposite directions, causing the robot to turn in place.



Command:

```
pointTurn(direction, speed);
```

Parameters: `direction`, `speed`

Valid Directions for `direction`**:**
`left` and `right`.

Valid Range Values for `speed`**:**
-127 to 127.

Usage without Parameters:

```
pointTurn();
wait();
stop();
```

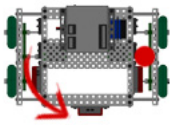
This snippet of code will make the robot turn right in place at speed 95 for 1 second and then stop. The default direction and speed are **`right`** and **95** for **`pointTurn()`**.

Usage with Parameters:

```
pointTurn(left, 63);
wait(2);
stop();
```

This snippet of code will make the robot turn left in place at half speed for 2 seconds.

Swing Turn
One wheel rotates while the other does not move, causing the robot to make a wide turn around the stopped wheel.



Command:

```
swingTurn(direction, speed);
```

Parameters: `direction`, `speed`

Valid Directions for `direction`**:**
`left` and `right`.

Valid Range Values for `speed`**:**
-127 to 127.

Usage without Parameters:

```
swingTurn();
wait();
stop();
```

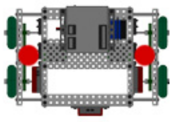
This snippet of code will make the robot make a wide right turn at speed 95 for 1 second and then stop. The default direction and speed are **`right`** and **95** for **`swingTurn()`**.

Usage with Parameters:

```
swingTurn(left, 63);
wait(2);
stop();
```

This snippet of code will make the robot make a wide left turn at half speed for 2 seconds.

Stop
Both wheels do not move, causing the robot to stop.



Command:

```
stop();
```

Parameters: N/A

Usage without Parameters:

```
forward();
wait();
stop();
```

This snippet of code will run the robot forward for 1 second and then stop. (Note that there are no parameters for **`stop()`**.

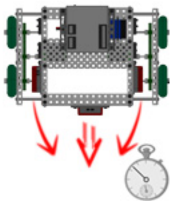
Usage with Parameters:

```
forward(63);
wait(2);
stop();
```

This snippet of code will run the robot forward at half speed for 2 seconds and then stop.

ROBOTC Natural Language - VEX PIC Reference:

Move Straight for Time
The robot will use encoders to maintain a straight course for a specified length of time in seconds.



Command:

```
moveStraightForTime(time, rightEncoder, leftEncoder);
```

Parameters: `time`, `rightEncoder`, `leftEncoder`

Valid Range Values for `time`:
0 to 32766 (Must be whole numbers; VEX PIC does not support decimal "floating point" values.)

Acceptable Sensors for `rightEncoder`, `leftEncoder`:
ANALOG / DIGITAL ports 1 through 16 (and your names for them given in Motors and Sensors Setup.)
* Don't forget the interrupt ports! *

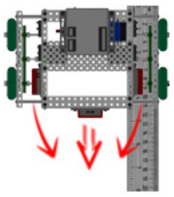
Parameters are required for VEX PIC sensor functions. There are NO defaults.

Usage with Parameters:

```
moveStraightForTime(7.5, in5, in3);
stop();
```

This snippet of code will make the robot move forward, maintaining a straight heading for 7.5 seconds using quadrature encoders in A/D-ports 5+interrupt and 3+interrupt, and then stop.

Move Straight for Rotations
The robot will use encoders to maintain a straight course for a specified distance in rotations.



Command:

```
moveStraightForRotations(time, rightEncoder, leftEncoder);
```

Parameters: `rotations`, `rightEncoder`, `leftEncoder`

Valid Range Values for `rotaions`:
0 to 32766 (Must be whole numbers; VEX PIC does not support decimal "floating point" values.)

Acceptable Sensors for `rightEncoder`, `leftEncoder`:
ANALOG / DIGITAL ports 1 through 16 (and your names for them given in Motors and Sensors Setup.)
* Don't forget the interrupt ports! *

Parameters are required for VEX PIC sensor functions. There are NO defaults.

Usage with Parameters:

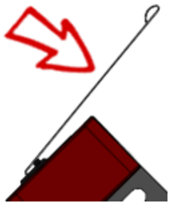
```
moveStraightForRotations(4.75, in5, in3);
stop();
```

This snippet of code will make the robot move forward, maintaining a straight heading for 4.75 rotations using quadrature encoders in A/D-ports 5+interrupt and 3+interrupt, and then stop.

ROBOTC Natural Language - VEX PIC Reference:

Until Functions:

Until Touch
The robot continues what it was doing until the touch sensor is pressed in.



Command:

```
untilTouch(sensorPort);
```

Parameters: `sensorPort`

Acceptable Sensors for `sensorPort`:
ANALOG / DIGITAL ports 1 through 16 (and your names for them given in Motors and Sensors Setup.)

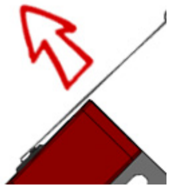
Parameters are required for VEX PIC sensor functions. There are NO defaults.

Usage with Parameters:

```
forward(63);
untilTouch(in10);
stop();
```

This snippet of code will run the robot forward at half speed until the touch sensor in A/D-port 10 is pressed, and then stop.

Until Release
The robot continues what it was doing until the touch sensor is released out.



Command:

```
untilRelease(sensorPort);
```

Parameters: `sensorPort`

Acceptable Sensors for `sensorPort`:
ANALOG / DIGITAL ports 1 through 16 (and your names for them given in Motors and Sensors Setup.)

Parameters are required for VEX PIC sensor functions. There are NO defaults.

Usage with Parameters:

```
forward(63);
untilRelease(in10);
stop();
```

This snippet of code will run the robot forward at half speed until the touch sensor in A/D-port 10 is released, and then stop.

Until Bump

The robot continues what it was doing until the touch sensor is pressed in and then released out.

Command:

```
untilBump(sensorPort);
```

Parameters: `sensorPort`

Acceptable Sensors for `sensorPort`**:**
ANALOG / DIGITAL ports 1 through 16 (and your names for them given in Motors and Sensors Setup.)

Parameters are required for VEX PIC sensor functions. There are NO defaults.

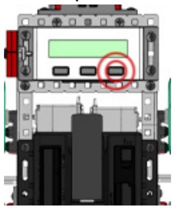
Usage with Parameters:

```
forward(63);
untilBump(in10);
stop();
```

This snippet of code will run the robot forward at half speed until the touch sensor in A/D-port 10 is pressed in and then released out, and then stop.

Until Button Press

The robot continues what it was doing until a specified button on the VEX LCD is pressed. *Connect the VEX LCD to UART-port 2.*



Command:

```
untilButtonPress(lcdButton);
```

Parameters: `lcdButton`

Valid LCD Buttons for `lcdButton`**:**
`centerBtnVEX` - VEX LCD center button
`rightBtnVEX` - VEX LCD right button
`leftBtnVEX` - VEX LCD left button

Usage without Parameters:

```
forward();
untilButtonPress();
stop();
```

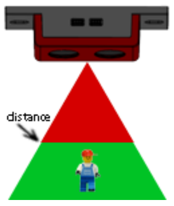
This snippet of code will run the robot forward until a button on the VEX LCD is pressed. The default button is `centerBtnVEX` for `untilBtnPress()`.

Usage with Parameters:

```
forward(63);
untilButtonPress(rightBtnVEX);
stop();
```

This snippet of code will run the robot forward at half speed until the right button on the VEX LCD is pressed.

Until Sonar Greater Than
The robot continues what it was doing until the sonar sensor reads a value greater than a set distance in centimeters.



Command:

```
untilSonarGreaterThan(distance, sensorPort);
```

Parameters: `distance, sensorPort`

Acceptable Values for `distance`**:**
0 to 255 (inches).

Acceptable Sensors for `sensorPort`**:**
ANALOG / DIGITAL ports 1 through 16 (and your names for them given in Motors and Sensors Setup.)
* Don't forget the interrupt ports! *

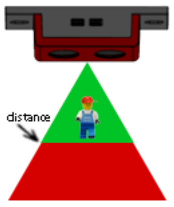
Parameters are required for VEX PIC sensor functions. There are NO defaults.

Usage with Parameters:

```
forward(63);
untilSonarGreatherThan(45, in2);
stop();
```

This snippet of code will run the robot forward at half speed until the sonar sensor in A/D-port 2+interrupt reads a value greater than 45 inches, and then stop.

Until Sonar Less Than
The robot continues what it was doing until the sonar sensor reads a value less than a set distance in centimeters.



Command:

```
untilSonarLessThan(distance, sensorPort);
```

Parameters: `distance, sensorPort`

Acceptable Values for `distance`**:**
0 to 255 (inches).

Acceptable Sensors for `sensorPort`**:**
ANALOG / DIGITAL ports 1 through 16 (and your names for them given in Motors and Sensors Setup.)
* Don't forget the interrupt ports! *

Parameters are required for VEX PIC sensor functions. There are NO defaults.

Usage with Parameters:

```
forward(63);
untilSonarLessThan(45, in2);
stop();
```

This snippet of code will run the robot forward at half speed until the sonar sensor in A/D-port 2+interrupt reads a value less than 45 inches, and then stop.

Until Potentiometer Greater Than

The robot continues what it was doing until the potentiometer sensor reads a value greater than a set position.

Command:

```
untilPotentiometerGreaterThan(position, sensorPort);
```

Parameters: `position`, `sensorPort`

Valid Range Values for `position`:
0 to 1023 (However due to mechanical stops, you may be limited to the range of 5 to 1018.)

Acceptable Sensors for `sensorPort`:
ANALOG / DIGITAL ports 1 through 16 (and your names for them given in Motors and Sensors Setup.)

Parameters are required for VEX PIC sensor functions. There are NO defaults.

Usage with Parameters:

```
startMotor(port8, 63);
untilPotentiometerGreaterThan(800, in4);
stop();
```

This snippet of code will run the motor on port 8 at speed 63 until the potentiometer in A/D-port 4 reaches a value greater than 800, and then stop.

Until Potentiometer Less Than

The robot continues what it was doing until the potentiometer sensor reads a value less than a set position.

Command:

```
untilPotentiometerLessThan(position, sensorPort);
```

Parameters: `position`, `sensorPort`

Valid Range Values for `position`:
0 to 1023 (However due to mechanical stops, you may be limited to the range of 5 to 1018.)

Acceptable Sensors for `sensorPort`:
ANALOG / DIGITAL ports 1 through 16 (and your names for them given in Motors and Sensors Setup.)

Parameters are required for VEX PIC sensor functions. There are NO defaults.

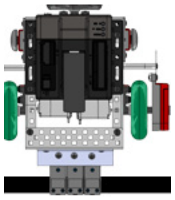
Usage with Parameters:

```
startMotor(port8, 63);
untilPotentiometerLessThan(40, in4);
stop();
```

This snippet of code will run the motor on port 8 at speed 63 until the potentiometer in A/D-port 4 reaches a value less than 40, and then stop.

Until Dark

The robot continues what it was doing until the line tracking sensor reads a value darker than a specified threshold.



Command:

```
untilDark(threshold, sensorPort);
```

Parameters: `threshold`, `sensorPort`

Valid Range Values for `threshold`:
(light) 0 to 1023 (dark)

Acceptable Sensors for `sensorPort`:
ANALOG / DIGITAL ports 1 through 16 (and your names for them given in Motors and Sensors Setup.)

Parameters are required for VEX PIC sensor functions. There are NO defaults.

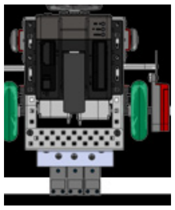
Usage with Parameters:

```
forward(63);
untilDark(1005, in4);
stop();
```

This snippet of code will run the robot forward at half speed until the line tracking sensor in A/D-port 4 reads a value darker than 1005, and then stop.

Until Light

The robot continues what it was doing until the line tracking sensor reads a value lighter than a specified threshold.



Command:

```
untilLight(threshold, sensorPort);
```

Parameters: `threshold`, `sensorPort`

Valid Range Values for `threshold`:
(light) 0 to 1023 (dark)

Acceptable Sensors for `sensorPort`:
ANALOG / DIGITAL ports 1 through 16 (and your names for them given in Motors and Sensors Setup.)

Parameters are required for VEX PIC sensor functions. There are NO defaults.

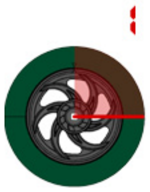
Usage with Parameters:

```
forward(63);
untilLight(1005, in4);
stop();
```

This snippet of code will run the robot forward at half speed until the line tracking sensor in A/D-port 4 reads a value lighter than 1005, and then stop.

ROBOTC Natural Language - VEX PIC Reference:

Until Rotations
The robot continues what it was doing until the quadrature encoder rotations reach the desired value.



Command:

```
untilRotations(rotations, sensorPort);
```

Parameters: `rotations`, `sensorPort`

Valid Range Values for `rotations`:
0 to 32766. (Due to hardware limitations of the VEX PIC, only whole rotations can be used -- no decimals.)

Acceptable Sensors for `sensorPort`:
ANALOG / DIGITAL ports 1 through 16 (and your names for them given in Motors and Sensors Setup.)
* Don't forget the interrupt ports! *

Parameters are required for VEX PIC sensor functions. There are NO defaults.

Usage with Parameters:

```
forward(63);
untilRotations(3, in3);
stop();
```

This snippet of code will run the robot forward at half speed for 3 rotations using a quadrature encoder in A/D-port 3+interrupt, and then stop.

Until Encoder Counts
The robot continues what it was doing until the quadrature encoder counts reach the desired value.

Command:

```
untilEncoderCounts(counts, sensorPort);
```

Parameters: `counts`, `sensorPort`

Valid Range Values for `counts`:
0 to 32766.

Acceptable Sensors for `sensorPort`:
ANALOG / DIGITAL ports 1 through 16 (and your names for them given in Motors and Sensors Setup.)
* Don't forget the interrupt ports! *

Parameters are required for VEX PIC sensor functions. There are NO defaults.

Usage with Parameters:

```
forward(63);
untilEncoderCounts(990, in3);
stop();
```

This snippet of code will run the robot forward at half speed for 990 encoder counts (2.75 rotations) using a quadrature encoder in A/D-port 3+interrupt, and then stop.

Special Functions:

LED ON
Turn an LED in a specified digital-port ON.

Command:

```
turnLEDOn(sensorPort);
```

Parameters: `sensorPort`

Acceptable Sensors for `sensorPort`:
ANALOG / DIGITAL ports 1 through 16 (and your names for them given in Motors and Sensors Setup.)
Note that you must set these digital-ports to "VEX LED".

Usage without Parameters:

```
turnLEDOn();
```

This snippet of code will turn an LED in A/D-port 2 ON.
The default sensor port is `in12` for `turnLEDOn()`.

Usage with Parameters:

```
turnLEDOn(in7);
```

This snippet of code will turn an LED in A/D-port 7 ON.

LED OFF
Turn an LED in a specified digital-port OFF.

Command:

```
turnLEDOff(sensorPort);
```

Parameters: `sensorPort`

Acceptable Sensors for `sensorPort`:
ANALOG / DIGITAL ports 1 through 16 (and your names for them given in Motors and Sensors Setup.)
Note that you must set these A/D-ports to "VEX LED".

Usage without Parameters:

```
turnLEDOff();
```

This snippet of code will turn an LED in A/D-port 2 OFF.
The default sensor port is `in12` for `turnLEDOff()`.

Usage with Parameters:

```
turnLEDOff(in7);
```

This snippet of code will turn an LED in A/D-port 7 OFF.

Flashlight ON
Turn a VEX Flashlight in a specified motor-port ON at a specified brightness.

Command:

```
turnFlashlightOn(motorPort, brightness);
```

Parameters: `motorPort`, `brightness`

Acceptable Motors for `motorPort`:
MOTOR ports 1 through 8 (and your names for them given in Motors and Sensors Setup.)

NOTE Brightness control only available in motor-ports 1 through 8 when connected to a VEX Motor Controller 29.)

Valid Range Values for `brightness`:
(off) 0 to 127 (bright)

Usage without Parameters:

```
turnFlashlightOn();
```

This snippet of code will turn a VEX Flashlight in motor-port 4 ON at brightness level 63 (half bright). The default motor port and brightness are **`port4`** and **`63`** for **`turnFlashlightOn()`**.

Usage with Parameters:

```
turnFlashlightOn(port8, 127);
```

This snippet of code will turn a VEX Flashlight in motor-port 8 ON at brightness level 127 (full bright).

Flashlight OFF
Turn a VEX Flashlight in a specified motor-port OFF.

Command:

```
turnFlashlightOff(motorPort);
```

Parameters: `motorPort`

Acceptable Motors for `motorPort`:
MOTOR ports 1 through 8 (and your names for them given in Motors and Sensors Setup.)

Usage without Parameters:

```
turnFlashlightOff();
```

This snippet of code will turn a VEX Flashlight in motor-port 4 OFF. The default motor port is **`port4`** for **`turnFlashlightOff()`**.

Usage with Parameters:

```
turnFlashlightOff(port8);
```

This snippet of code will turn a VEX Flashlight in motor-port 8 OFF.