RobotC Reference Sheet (robotC.h Version 0.96)

Header File

```
#include "robotC.h"
```

This line must be included in each program right below the #include "robot-config.h"

Motor Movement Commands

```
startMotor(LeftMotor, 40);
```

Turns on a motor at a specified percent power. The percentage can be between -100 and 100.

In this example the motor "LeftMotor" is running forward at 40% speed.

```
stopMotor(RightMotor);
```

Stops a specified motor.

In this example the motor "RightMotor" is stopped.

```
moveRelative(RightMotor,-720,75);
```

Tells a motor to move for a specified distance, at a specified power. Power can range from -100 to 100

"RightMotor" will run in reverse for 720 degrees at 75% speed.

This command does not wait to finish before going to the next line.

```
moveAbsolute(RightMotor, 90, 25);
```

Tells a motor to move to a specific location, moving at a certain power. Power ranges from -100 to 100

"RightMotor" will run until it reaches position 90 degrees. It will run at 25% speed to get there.

This command does not wait to finish before going to the next line.

```
waitForMotor(ArmMotor);
```

Pauses the program until a specified motor is done moving

In this example the program will wait until "ArmMotor" is done moving before going to the next line.

Motor Settings

```
setCoast(RightMotor);
setBrake(RightMotor);
setHold(RightMotor);
```

Each motor can be set to do different things when it is not running with these commands.

SetCoast lets the motor coast to a stop, and it can be turned by outside forces.

SetBrake actively tries to stop the motor when it is unpowered and resists moving.

SetHold not only stops the motor, but apply power to resist moving once there.

```
setMaxTorque(ArmMotor, 75);
```

This command specifies the percentage of a motors torque that you will allow it to output (in percent).

In this example "ArmMotor" will only use up to 75% of its maximum torque.

Motor Sensing

```
getMotorCurrent(LeftMotor);
getMotorPower(LeftMotor);
getMotorTorque(LeftMotor);
```

These functions return the current of the motor in Amps, the power in watts, or the torque in newton*meters.

In these examples they are returning the values for the motor "LeftMotor".

```
getMotorEff(RightMotor);
getMotorTemp(RightMotor);
```

These functions return a value between 0-100. In the first example the function returns the efficiency of as a percent. In the second example the function returns the motor temperature as a percentage of the overheat value. At 100 the motor is overheating, and power to it will be automatically reduced.

In these examples data for the motor "RightMotor" is being called.

```
getIfMotorSpinning(LeftMotor);
```

This function tells you if a motor is spinning. It returns a 0 if the motor is not spinning, and a 1 if it is.

Time Commands

```
wait(3.25);
```

This command cause the robot to pause before going to the next line. The number is in seconds.

This example pauses 3.25 seconds.

Note that decimal numbers can be used here.

```
setTime(0);
```

The robot brain has a built in timer that is constantly running in the background.

This command is used to set that timer to a certain time. The time is in seconds and decimals are okay.

In this example the time is reset to 0 seconds. Although other times can be used as well.

```
getTime();
```

This function returns the current value of the timer.

Sensor Reading and Setting

```
getAnalog(LightSensor);
```

This returns the value of any other analog sensor. The values range from 0-1024

```
getDigital(Phototransistor);
```

This returns the value of any other digital sensor. The possible values are *true & false*

```
getSonar(Sonar1);
getSonarMm(Sonar1);
getSonarIn(Sonar1);
```

Returns the distance of a sonar reading. getSonar & getSonarMm are in mm. getSonarIn is in inches. If the sonar is connected in correctly a negative value will be displayed.

```
getGyro(Gyro1);
```

This returns the current heading from the Gyro. The reading in in the range of 0-360.

```
setEncoder(LeftMotor, 270);
```

This command changes the encoder value for an encoder built into a motor.

In this example the encoder on the motor “LeftMotor” is set to 270

```
setQuadEncoder(Encoder1, 0);
```

This command changes the encoder value for a quadrature encoder using the 3 wire ports.

In this example the encoder named “Encoder1” is reset to 0

```
getEncoder(LeftMotor);
```

Returns the value of an encoder built into a motor. In this case the value from the encoder in LeftMotor.

```
getQuadEncoder(Encoder1);
```

Returns the value for a quadrature encoder using the 3 wire ports. In this case the value from Encoder1.

LEGACY MOTOR COMMANDS

USING THE BRAIN TOUCHSCREEN

```
clearScreen();
```

Erases the Brain touchscreen. This does *not* move the cursors location however.

```
setScreenCursor(2,4);
```

Moves the cursor to a X,Y position. The location of the cursor is where text is next wrote on the screen.

In this example the cursor is 2 spaces from the left, and 4 lines from the top.

```
Brain.Screen.print("Put text Here");
Brain.Screen.print(variable);
Brain.Screen.print("The highest reading is %d", highReading);
```

This prints to the screen. Text can be printed to the screen directly by enclosing it in quotation marks (1st example), or a variable can be printed writing it without quotes (2nd example), or a place holder can be used to write text combined with variables (3rd example)

```
getIfScreenTouched()
```

Return either a 0 (screen is not being touched), or a 1 (screen is being touched).

```
getScreenTouchX();
getScreenTouchY();
```

Returns the X or Y position of the last place the screen was touched.

```
drawLine(x1,y1,x2,y2);
drawRectangle(x1,y1,width,height);
drawCircle(x1,y1, radius);
```

Draws a shape on the screen.

Example 1 draws a line from (x1,y1) to (x2,y2).

Example 2 draws a rectangle with one corner at (x1,y1) and with a specified width and height.

Example 3 draws a circle centered at (x1,y1) and with a specified radius.

CAMERA

In order for the camera functions to work the Vision Sensor must keep the name "Vision1"

```
#include "camera.h"
```

This line must be included in the program right below the other #include lines at top

```
blobX(1,3);
blobY(2,4);
```

These functions return the X or Y location of a color object (or blob) for a given color signature. The first number is the color signature, and the second number is the number of the blob of that color ranked by size (1 is the largest). If the blob doesn't exist the value -1 will be returned.

```
blobHeight(2,1);
blobWidth(1,2);
blobSize(2,3);
```

These functions return the Height, Width, and Size of the rectangular bounding box in pixels. The first number is the color signature, and the second number is the number of the blob of that color ranked by size (1 is the largest). If the blob doesn't exist the value -1 will be returned.

```
blobCount(4);
```

This function tells you how many blobs are detected for a given color signature.

```
blobExists(1,3);
```

This function tells you if a blob exists. This example check to see if there is a third largest blob in color signature 1. If there is the function returns 1, if not it returns 0.

```
setLedColor(255,0,255);
```

This changes the color of the LED on top of the camera. You can specify on a scale of 0-255 how much Red, Green, and Blue light should be combined to create the color of the LED. This example sends 255 to Red and Blue, and 0 to Green making a purple light.

CONTROLLER BUTTONS & JOYSTICKS

In order for the functions to work the Controller must keep the name "Controller1"

```
#include "radio.h"
```

This line must be included in the program right below the other #include lines at top

All of the button function (getXbutton() , getUpButton() , getL1Button , etc) return a value of 1 when pressed and a value of 0 when not pressed.

The stick functions return a value of -100 to 100. It'll return a value of approximately 0 when centered.

For the horizontal/X axis (getLeftStickX() & getRightStickX()) -100 is full Left and 100 is full Right.
For the vertical/Y axis (getLeftStickY() & getRightStickY()) -100 is full Down and 100 is full Up.

BATTERY SENSING

```
getBattLife();
```

Returns a value giving the battery charge percentage. Ranges from 0-100.

```
getBattTemp();
```

Returns the battery temperature as a percent of maximum temperature. Ranges from 0-100.

COMPETITION