Encoder Programming Guide (ROBOTC®)

Introduction:

In this extension, motor encoders will be added to the wheels of the Ranger Bot. The Ranger Bot with Encoders will be programmed to move forward until it detects an object, turn 90°, and move forward until it detects a second object. The program will remember the encoder count distances from the start to the first object and from the first object to the second object. Using these encoder values, the robot will turn 90°, move a distance equal to the first encoder value, turn 90°, and finally move a distance equal to the second encoder value. From start to finish, this sequence of steps will make the robot move in a rectangle shape back to its starting point. This guide is for use with the ROBOTC® programming language.

Getting Started:

1. To start the program, type the **Main Task** function followed by an **opening brace**.

```
task main()
{
```

2. Declare six **integer variables** as seen in the code below. Set them to the values shown. After each variable there are two slashes (//) followed by what that variable is representing. These mark a comment. Notice that comments appear in green when code is typed into ROBOTC.

   Note: A comment is not part of the code that the NXT Brick will execute, but it helps the programmer by allowing the addition of notes to describe what has been done.

```
int object1Dist;      // Distance from start to first object
int object2Dist;      // Distance from first to second object
int stopDist = 30;    // Stopping distance before objects
int turnDist = 1380;  // Encoder counts that wheels rotate during each 90-degree turn
int fwdPower = 10;    // Motor power when robot drives forward
int turnPower = 15;   // Motor power when robot turns
```

3. Add the code below to set the encoder of the left motor to zero. A motor encoder is a sensor that determines the amount and speed of rotation of a motor. In this program we will be concentrating on using the motor encoder to control the movement of the robot.

```
nMotorEncoder[motorD] = 0;                  // Sets the motorD encoder to zero
```

4. Add a **While Loop** that executes while the Ultrasonic Sensor reading is greater than the **stopDist** variable (set to 30 cm). Set both DC motors' power to the **fwdPower** value while this condition remains True. This will make the robot move forward in a straight line towards the first object.

```
while(SensorValue(SonarSensor) > stopDist)    // Moves forward until within 30 cm (stopDist)
                                                 of first object
{
   motor[motorD]= fwdPower;
   motor[motorE]= fwdPower;
      }
```

Encoder Programming Guide (ROBOTC®)

5. After the robot is 30 cm from the first object, turn off the motors and make the robot wait for half a second.

```
motor[motorD] = 0;          // Stop motors and make the robot wait for half a second
motor[motorE] = 0;
wait1Msec(500);
```

6. The code below reads the current output value of the encoder sensor. This value is the number of encoder counts that make up the distance from the robot's starting point to the point 30 cm from the first object. The program needs to remember this value for later on in the program, so it writes it to the variable **object1Dist**.

```
object1Dist = nMotorEncoder[motorD];          // Sets distance from start to first object
```

7. The motor encoder needs to be set to zero again to read it correctly in the next step.

```
nMotorEncoder[motorD] = 0;          // Resets the motor encoder of motor D to zero
```

8. Add a **While Loop** that executes while the current encoder value is below the variable **turnDist**. This variable is an encoder count that is equal to the distance that a wheel rotates during a 90-degree turn. Set the DC motors' power to the **turnPower** variable while this condition remains True; however, the right motor will be set to the negative value of **turnPower**. This will make the robot turn on the spot.

   Note: You must adjust the **turnDist** value so the robot turns exactly 90 degrees. The preset value of 1380 shown in Step 2 is a good starting value; however, the robot turns different amounts on different ground surfaces. The value will probably have to be adjusted for a specific surface.

```
while (nMotorEncoder[motorD] < turnDist)          // Must tweak the turnDist encoder count
                                                     value so that robot turns 90 degrees
{
    motor[motorD] = turnPower;
    motor[motorE] = -turnPower;
}
```

9. As was done after the first While Loop, turn off the motors and make the robot wait for half a second. These three lines of code will be repeated after every While Loop.

```
motor[motorD] = 0;
motor[motorE] = 0;
wait1Msec(500);
```

10. Now the robot is ready to search for the second object. First, the motor encoder needs to be reset to zero.

```
nMotorEncoder[motorD] = 0;
```

Encoder Programming Guide (ROBOTC®)

11. As in Step 4, add a While Loop that drives the robot straight until it is within 30 cm of the second object. After the loop has been executed, turn off the motors and make the robot wait for half a second.

```
while (SensorValue[SonarSensor] > stopDist)     // Moves forward until within 30 cm (stopDist)
                                                   of second object
{
    motor[motorD] = fwdPower;
    motor[motorE] = fwdPower;
}


motor[motorD] = 0;
motor[motorE] = 0;
wait1Msec(500);
```

12. Similar to Step 6, the distance from the first object to the second object must be remembered and written to the variable **object2Dist**. After storing the encoder value, reset the encoder to zero.

```
object2Dist = nMotorEncoder[motorD];            // Sets distance from first to second object
nMotorEncoder[motorD] = 0;
```

13. Make the robot turn again using the same turning loop as in Step 8. Then, turn off the motors and make the robot wait for half a second.

```
while (nMotorEncoder[motorD] < turnDist)
{
    motor[motorD] = turnPower;
    motor[motorE] = -turnPower;
}


motor[motorD] = 0;
motor[motorE] = 0;
wait1Msec(500);
```

14. The robot will now begin traveling back to its starting point. First, reset the motor encoder to zero.

```
// Begins to travel back to the start


   nMotorEncoder[motorD] = 0;
```

15. Add a **While Loop** that drives the robot straight for the distance of the encoder count value stored in the variable **object1Dist**. This means the robot will travel the same distance as it did from its starting point to the first object. After the loop finishes, turn off the motors and make the robot wait for half a second.

```
while (nMotorEncoder[motorD] < object1Dist)
{
```

Encoder Programming Guide (ROBOTC®)

```
    motor[motorD] = fwdPower;
    motor[motorE] = fwdPower;
}

motor[motorD] = 0;
motor[motorE] = 0;
wait1Msec(500);
```

16. Reset the motor encoder to zero. Then make the robot turn 90 degrees for the third and last time. Use the same code section from the previous turns as displayed below.

```
nMotorEncoder[motorD] = 0;

while (nMotorEncoder[motorD] < turnDist)
{
    motor[motorD] = turnPower;
    motor[motorE] = -turnPower;
}

motor[motorD] = 0;
motor[motorE] = 0;
wait1Msec(500);
```

17. Reset the motor encoder. Similar to Step 15, add a **While Loop** that drives the robot straight for the distance of the encoder count value stored in the variable **object2Dist**. This means the robot will travel the same distance as it did from the first to the second object. Now the robot is back to its starting point, having traveled in a rectangle movement.

```
nMotorEncoder[motorD] = 0;

while (nMotorEncoder[motorD] < object2Dist)
{
    motor[motorD] = fwdPower;
    motor[motorE] = fwdPower;
}
```

18. Before ending the program, the robot should be instructed to stop the DC drive motors.

```
motor[motorD] = 0;
motor[motorE] = 0;
```

Encoder Programming Guide (ROBOTC®)

19. Add a **closing brace** to end the program.

```
}
```

Completed Code:

```
task main()
{
  int object1Dist;               // Distance from start to first object
  int object2Dist;               // Distance from first to second object
  int stopDist = 30;             // Stopping distance before objects
  int turnDist = 1380;           // Encoder counts that wheels rotate during each 90-degree turn
  int fwdPower = 10;             // Motor power when robot drives forward
  int turnPower = 15;            // Motor power when robot turns

                // Searching for the first object

  nMotorEncoder[motorD] = 0;                    // Sets the motorD encoder to zero

  while (SensorValue[SonarSensor] > stopDist)   // Moves forward until within 30 cm (stopDist) of first
                                                  object
  {
    motor[motorD] = fwdPower;
    motor[motorE] = fwdPower;
  }

  motor[motorD] = 0;                            // Stop motors and make the robot wait for half a
                                                  second
  motor[motorE] = 0;
  wait1Msec(500);

  object1Dist = nMotorEncoder[motorD];          // Sets distance from start to first object
  nMotorEncoder[motorD] = 0;                    // Resets the motor encoder of motor D to zero

  while (nMotorEncoder[motorD] < turnDist)      // Must tweak the turnDist encoder count value so
                                                  that robot turns 90 degrees
  {
    motor[motorD] = turnPower;
    motor[motorE] = -turnPower;
  }

  motor[motorD] = 0;
  motor[motorE] = 0;
  wait1Msec(500);
```

Encoder Programming Guide (ROBOTC®)

Completed Code (continued):

```
              // Searching for the second object

nMotorEncoder[motorD] = 0;

while (SensorValue[SonarSensor] > stopDist)   // Moves forward until within 30 cm (stopDist) of
                                                 second object
{
  motor[motorD] = fwdPower;
  motor[motorE] = fwdPower;
}

motor[motorD] = 0;
motor[motorE] = 0;
wait1Msec(500);

object2Dist = nMotorEncoder[motorD];          // Sets distance from first to second object
nMotorEncoder[motorD] = 0;

while (nMotorEncoder[motorD] < turnDist)
{
  motor[motorD] = turnPower;
  motor[motorE] = -turnPower;
}

motor[motorD] = 0;
motor[motorE] = 0;
wait1Msec(500);

              // Begins to travel back to the start

nMotorEncoder[motorD] = 0;

while (nMotorEncoder[motorD] < object1Dist)
{
  motor[motorD] = fwdPower;
  motor[motorE] = fwdPower;
}

motor[motorD] = 0;
motor[motorE] = 0;
wait1Msec(500);
```

Encoder Programming Guide (ROBOTC®)

Completed Code (continued):

```
  nMotorEncoder[motorD] = 0;

  while (nMotorEncoder[motorD] < turnDist)
  {
    motor[motorD] = turnPower;
    motor[motorE] = -turnPower;
  }

  motor[motorD] = 0;
  motor[motorE] = 0;
  wait1Msec(500);

              // Drives back to starting point

  nMotorEncoder[motorD] = 0;

  while (nMotorEncoder[motorD] < object2Dist)
  {
    motor[motorD] = fwdPower;
    motor[motorE] = fwdPower;
  }

  motor[motorD] = 0;
  motor[motorE] = 0;
}
```