```
#pragma config(Sensor, in1,    AutonDial,      sensorPotentiometer)
#pragma config(Sensor, in2,    LiftPot,        sensorPotentiometer)
#pragma config(Sensor, in3,    RightClawPot,   sensorPotentiometer)
#pragma config(Sensor, in4,    LeftClawPot,    sensorPotentiometer)
#pragma config(Sensor, in5,    AutonSide,      sensorPotentiometer)
#pragma config(Sensor, in6,    Expander,       sensorAnalog)
#pragma config(Sensor, dgtl9,  LiftEnc,        sensorQuadEncoder)
#pragma config(Sensor, dgtl11, DistanceSensor, sensorQuadEncoder)
#pragma config(Motor,  port1,           Right1,        tmotorVex393TurboSpeed_HBridge, openLoop, reversed)
#pragma config(Motor,  port2,           Right2,        tmotorVex393TurboSpeed_MC29, openLoop, reversed)
#pragma config(Motor,  port3,           Left1,         tmotorVex393TurboSpeed_MC29, openLoop, reversed)
#pragma config(Motor,  port4,           Left2,         tmotorVex393TurboSpeed_MC29, openLoop, reversed)
#pragma config(Motor,  port5,           LiftR1,        tmotorVex393HighSpeed_MC29, openLoop)
#pragma config(Motor,  port6,           LiftR2,        tmotorVex393HighSpeed_MC29, openLoop)
#pragma config(Motor,  port7,           LiftL1,        tmotorVex393HighSpeed_MC29, openLoop)
#pragma config(Motor,  port8,           LiftL2,        tmotorVex393HighSpeed_MC29, openLoop)
#pragma config(Motor,  port9,           ClawR,         tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port10,          ClawL,         tmotorVex393_HBridge, openLoop)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

  /////////////////////////////////
 //Platforming And Program Setup//
////////////////////////////////////////////////////////////////////////////////////////////////////
#pragma platform(VEX)//Make the platform be VEX//
//Competition Control and Duration Settings//
#pragma competitionControl(Competition)//Go into the control of the competition//
#pragma autonomousDuration( 15/*Time in seconds*/ )//Amount of seconds that are in the auton duration//
#pragma userControlDuration(  105/*Time in seconds*/  )//Amount of seconds that are in the user control duration//
#include "Vex_Competition_Includes.c"   //Main competition background code...do not modify!//
////////////////////////////////////////////////////////////////////////////////////////////////////

        // / //////////////////////////////////////////////////////////////////////////////// / //
       //   ///    __  _______  __    _______     ___          ____   ____               ///   //
      // / ///   /_ \ \  ____\ \ \   \__  _\  |   \      \_  _\  \_  _\              /// / //
     //   ///   // \ \ \ \__      \ \      \ \  | |\ \       \ \    \ \             ///   //
    // / ///   //   \ \ \  __\     \ \      \ \  |  _  \       \ \    \ \           /// / //
   //   ///   //_____\ \ \ \____    \ \___   \ \ | | \  \     _\ \_  _\ \_         ///   //
  // / ///    \_________/  \______\ \_____\  \_\ |_|  \__\     \_____\ \_____\    /// / //
 //   ///                                                                      ///   //
// / ///////////////////////////////////////////////////////////////////////////////// / //

       /////////////////////////////////////////////////////////////////////////////////////////
      //
     //                          Pre-Autonomous Functions
    //
```

```
   // You may want to perform some actions before the competition starts. Do them in the
  // following function.
 //
////////////////////////////////////////////////////////////////////////////////////////
void pre_auton()
{
  bLCDBacklight=true;//Turn the LCD backlight on to show selected autons//
  //Set bStopTasksBetweenModes to false if you want to keep user created tasks running between
  //Autonomous and Tele-Op modes. You will need to manage all user created tasks if set to false.
  bStopTasksBetweenModes = true;//Stops all tasks between any mode//

    ////////////////////////////
   //First-Time pre-auton setup//
  ////////////////////////////////////////////////////////////////
  clearLCDLine( 0/*0=Top Line, 1=Bottom Line*/  );//Clear the LCD screen to display any needed data//
  clearLCDLine( 1/*0=Top Line, 1=Bottom Line*/  );//Clear the LCD screen to display any needed data//
  displayLCDCenteredString( 0/*0=Top Line, 1=Bottom Line*/  , "Sens Clring..."/*Text To Be Displayed*/  );//Display "Sen
  wait1Msec(2000);//Wait 2 seconds//
  SensorValue[LiftEnc]=0;//Clear The Lift Encoder//
  SensorValue[DistanceSensor]=0;//Clear the Distance Sensor//
  ////////////////////////////////////////////////////////////////

    ////////////////////////
   //Infinite self-check//
  ////////////////////////////////////////////////////////
  while(true){//Create a while loop to loop the clearing//

     ////////////////////////////
    //Distance Sensor Checking//
    ////////////////////////////////////////////////////////////////////////////////////////////////////
    if(SensorValue[DistanceSensor]  <-0.1/*Wheel Rotation In Degrees*/  ||SensorValue[DistanceSensor] >0.1/*Wheel Rotati
      clearLCDLine( 0/*0=Top Line, 1=Bottom Line*/  );//Clear the LCD screen to display any needed data//
      displayLCDCenteredString( 0/*0=Top Line, 1=Bottom Line*/  , "Sens Clring..."/*Text To Be Displayed*/  );//Display
      SensorValue[DistanceSensor]=0;//Clear the Distance Sensor//
      wait1Msec(2000);//Wait 2 seconds//
      SensorValue[DistanceSensor]=0;//Clear the Distance Sensor//
    }
    ////////////////////////////////////////////////////////////////////////////////////////////////////

     ////////////////////////
    //Lift Encoder Checking//
    ////////////////////////////////////////////////////////////////////////////////////////////////////
    else if(SensorValue[LiftEnc]  <-0.1/*Lift Rotation In Degrees*/ ||SensorValue[LiftEnc]  >0.1/*Lift Rotation In Degre
      clearLCDLine( 0/*0=Top Line, 1=Bottom Line*/  );//Clear the LCD screen to display any needed data//
      displayLCDCenteredString( 0/*0=Top Line, 1=Bottom Line*/  , "Sens Clring..."/*Text To Be Displayed*/  );//Display
```

```
      SensorValue[LiftEnc]=0;//Clear The Lift Encoder//
      wait1Msec(2000);//Wait 2 seconds//
      SensorValue[LiftEnc]=0;//Clear The Lift Encoder//
    }
    ////////////////////////////////////////////////////////////////////////////////////////////////////

    clearLCDLine( 0/*0=Top Line, 1=Bottom Line*/  );//Clear the LCD screen to display any needed data//
    //Display that everything is checked and cleared when all done//
    displayLCDCenteredString( 0/*0=Top Line, 1=Bottom Line*/  , "Sens Cleared"/*Text To Be Displayed*/  );//Display "Sen
    wait1Msec(500);//Wait 0.5 seconds//

  }
//////////////////////////////////////////////////////

}

  ///////////////////////////////////
 //Tasks To Be Ran For Autonomous//
////////////////////////////////////////////////////////////////////////////////////////////////////////////

task downposition(){//Down Position For Autonomous//
  while(true){//Create An Infinite Running Loop//
    motor[ClawL]=((SensorValue[LeftClawPot] -1200/*Width From Starting Pos.*/ ) /10/*Sensitivity*/  ) ^2/*Slope*/ ;//Lef
    motor[ClawR]=((SensorValue[RightClawPot]  -1200/*Width From Starting Pos.*/ ) /10/*Sensitivity*/  ) ^2/*Slope*/ ;//R
    //Lift PI control to position//
    motor[LiftL1]=(SensorValue[LiftEnc] -2/*Height From Starting Pos.*/ ) *4/*Sensitivity*/ ;//Left lift being ran by th
    motor[LiftL2]=(SensorValue[LiftEnc] -2/*Height From Starting Pos.*/ ) *4/*Sensitivity*/ ;//Left lift being ran by th
    motor[LiftR1]=(SensorValue[LiftEnc] -2/*Height From Starting Pos.*/ ) *4/*Sensitivity*/ ;//Right lift being ran by t
    motor[LiftR2]=(SensorValue[LiftEnc] -2/*Height From Starting Pos.*/ ) *4/*Sensitivity*/ ;//Right lift being ran by t
  }
}

task closeclaw(){//Close Claw Position For Autonomous//
  while(true){//Create An Infinite Running Loop//
    motor[ClawL]=((SensorValue[LeftClawPot] -2700/*Width From Starting Pos.*/ ) /5/*Sensitivity*/ );//Left claw being ra
    motor[ClawR]=((SensorValue[RightClawPot]  -2700/*Width From Starting Pos.*/ ) /5/*Sensitivity*/ );//Right claw being
    //Lift PI control to position//
    motor[LiftL1]=(SensorValue[LiftEnc] -2/*Height From Starting Pos.*/ ) *4/*Sensitivity*/ ;//Left lift being ran by th
    motor[LiftL2]=(SensorValue[LiftEnc] -2/*Height From Starting Pos.*/ ) *4/*Sensitivity*/ ;//Left lift being ran by th
    motor[LiftR1]=(SensorValue[LiftEnc] -2/*Height From Starting Pos.*/ ) *4/*Sensitivity*/ ;//Right lift being ran by t
    motor[LiftR2]=(SensorValue[LiftEnc] -2/*Height From Starting Pos.*/ ) *4/*Sensitivity*/ ;//Right lift being ran by t
  }
}

task intakeposition(){//Intake(hold up) Position For Autonomous//
```

```
  while(true){//Create An Infinite Running Loop//
    //Make sure claw is closed//
    motor[ClawL]=((SensorValue[LeftClawPot] -2700/*Width From Starting Pos.*/ ) /5/*Sensitivity*/ );//Left claw being ra
    motor[ClawR]=((SensorValue[RightClawPot]  -2700/*Width From Starting Pos.*/ ) /5/*Sensitivity*/ );//Right claw being
    //Lift PI control to position//
    motor[LiftL1]=(SensorValue[LiftEnc] +50/*Height From Starting Pos.*/  ) *4/*Sensitivity*/ ;//Left lift being ran by
    motor[LiftL2]=(SensorValue[LiftEnc] +50/*Height From Starting Pos.*/  ) *4/*Sensitivity*/ ;//Left lift being ran by
    motor[LiftR1]=(SensorValue[LiftEnc] +50/*Height From Starting Pos.*/  ) *4/*Sensitivity*/ ;//Right lift being ran by
    motor[LiftR2]=(SensorValue[LiftEnc] +50/*Height From Starting Pos.*/  ) *4/*Sensitivity*/ ;//Right lift being ran by
  }
}

task dumpposition(){//Dump Position For Autonomous//
  while(true){//Create An Infinite Running Loop//
      if((-SensorValue[LiftEnc])<110){//If the lift is below dump position, keep claw closed//
        motor[ClawL]=((SensorValue[LeftClawPot] -2700/*Width From Starting Pos.*/ ) /5/*Sensitivity*/ );//Left claw bein
        motor[ClawR]=((SensorValue[RightClawPot]  -2700/*Width From Starting Pos.*/ ) /5/*Sensitivity*/ );//Right claw b
      }
      else{//When at or above dump position, open claw//
        motor[ClawL]=((SensorValue[LeftClawPot] -1200/*Width From Starting Pos.*/ ) /10/*Sensitivity*/  ) ^2/*Slope*/ ;/
        motor[ClawR]=((SensorValue[RightClawPot]  -1200/*Width From Starting Pos.*/ ) /10/*Sensitivity*/  ) ^2/*Slope*/
      }
      //Lift PI control to position//
      motor[LiftL1]=(SensorValue[LiftEnc] +145/*Height From Starting Pos.*/ ) *4/*Sensitivity*/ ;//Left lift being ran b
      motor[LiftL2]=(SensorValue[LiftEnc] +145/*Height From Starting Pos.*/ ) *4/*Sensitivity*/ ;//Left lift being ran b
      motor[LiftR1]=(SensorValue[LiftEnc] +145/*Height From Starting Pos.*/ ) *4/*Sensitivity*/ ;//Right lift being ran
      motor[LiftR2]=(SensorValue[LiftEnc] +145/*Height From Starting Pos.*/ ) *4/*Sensitivity*/ ;//Right lift being ran
  }
}
////////////////////////////////////////////////////////////////////////////////////////////////////

       ////////////////////////////////////////////////////////////////////////
      //
     //                                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.
```

```
}

       ////////////////////////////////////////////////////////////////////////////////////////
      //
     //                              LCD Auton Chooser And Voltage Task
    //
   // This task is used to display the selected autons during the user control phase of a VEX Competition.
  // You must modify the code to add your own robot specific commands here.
 //
////////////////////////////////////////////////////////////////////////////////////////
task showautons(){//Show the selected mode for the auton; starts in driver control mode//
  string backupBattery, primBattery;//Create a string that can be ran for the voltage displaying//
  while(true){//Create an infinite-running loop(obviously runnung for only driver control)//
    wait1Msec(100);//Slow down refresh rate to use less CPU as well as decrease glitches//
    clearLCDLine( 0/*0=Top Line, 1=Bottom Line*/  );//Clear the LCD screen to display any needed data//
    clearLCDLine( 1/*0=Top Line, 1=Bottom Line*/  );//Clear the LCD screen to display any needed data//

      ////////////////////
     //Voltage Warnings//
    ////////////////////////////////////////////////////////////////////////////////////////
    //Expander Battery Displaying//
    if(SensorValue[Expander]<2240){//If Expander Battery is below 8V (2240 Value)//
      displayLCDString( 0/*0=Top Line, 1=Bottom Line*/  , 0/*0=Uncentered, 1=Centered*/ , "Exp. Low:");//Display Expande
      sprintf(backupBattery, "%1.2f%c", SensorValue[Expander]/280.0, 'V');//Build the value to be displayed//
      displayNextLCDString(backupBattery);//Display the string "backupBattery"//
    }
    //Primary Battery Displaying//
    else if(nImmediateBatteryLevel<8000){//If Primary Battery is below 8V (7000 Value)//
      displayLCDString( 0/*0=Top Line, 1=Bottom Line*/  , 0/*0=Uncentered, 1=Centered*/ , "Prim. Low:");//Display Primar
      sprintf(primBattery, "%1.2f%c", nImmediateBatteryLevel/1000.0, 'V');//Build value to be displayed//
      displayNextLCDString(primBattery);//Display the string "primBattery"//
    }
    else{//When all are above 8V//
      displayLCDCenteredString( 0/*0=Top Line, 1=Bottom Line*/, "Batt Above 8V"/*Text To Be Displayed*/ );//Display "Bat
    }
    ////////////////////////////////////////////////////////////////////////////////////////

      //////////////
     //Left Side//
    ////////////////////////////////////////////////////////////////////////////////////////
    if(SensorValue[AutonSide] <1000/*SensorValue Starting W/ 0 From the Left*/  ){//When "autonside" is on the left side
      if(SensorValue[AutonDial] <800/*SensorValue Starting W/ 0 From the Left*/ ){//The farthest left side. Display no s
        displayLCDCenteredString( 1/*0=Top Line, 1=Bottom Line*/  , "No 2nd Cub|L"/*Text To Be Displayed*/  );//Display
      }
      else if(SensorValue[AutonDial]  <1600/*SensorValue Starting W/ 0 From the Left*/  &&SensorValue[AutonDial]  >800/*
```

```
        displayLCDCenteredString( 1/*0=Top Line, 1=Bottom Line*/  , "2nd Cub Left|L"/*Text To Be Displayed*/  ); //Displ
      }
      else if(SensorValue[AutonDial]  <2400/*SensorValue Starting W/ 0 From the Left*/  &&SensorValue[AutonDial]  >1600/
        displayLCDCenteredString( 1/*0=Top Line, 1=Bottom Line*/  , "2nd Cub Center|L"/*Text To Be Displayed*/  ); //Dis
      }
      else if(SensorValue[AutonDial]  <3200/*SensorValue Starting W/ 0 From the Left*/  &&SensorValue[AutonDial]  >2400/
        displayLCDCenteredString( 1/*0=Top Line, 1=Bottom Line*/  , "2nd Cub Right|L"/*Text To Be Displayed*/ ); //Displ
      }
      else if(SensorValue[AutonDial]  >3200/*SensorValue Starting W/ 0 From the Left*/  ){//Farthest right side, have no
        displayLCDCenteredString( 1/*0=Top Line, 1=Bottom Line*/  , "No Auton|L"/*Text To Be Displayed*/  ); //Display "
      }
    }
    ////////////////////////////////////////////////////////////////////////////////////////////

      ///////////////
     //Right Side//
    ////////////////////////////////////////////////////////////////////////////////////////////
    else{//When "autonside" is not on the left(basically on the right side)//
      if(SensorValue[AutonDial] <800/*SensorValue Starting W/ 0 From the Left*/ ){//The farthest left side. Display no s
        displayLCDCenteredString( 1/*0=Top Line, 1=Bottom Line*/  , "No 2nd Cub|R"/*Text To Be Displayed*/  );//Display
      }
      else if(SensorValue[AutonDial]  <1600/*SensorValue Starting W/ 0 From the Left*/  &&SensorValue[AutonDial]  >800/*
        displayLCDCenteredString( 1/*0=Top Line, 1=Bottom Line*/  , "2nd Cub Left|R"/*Text To Be Displayed*/  ); //Displ
      }
      else if(SensorValue[AutonDial]  <2400/*SensorValue Starting W/ 0 From the Left*/  &&SensorValue[AutonDial]  >1600/
        displayLCDCenteredString( 1/*0=Top Line, 1=Bottom Line*/  , "2nd Cub Center|R"/*Text To Be Displayed*/  ); //Dis
      }
      else if(SensorValue[AutonDial]  <3200/*SensorValue Starting W/ 0 From the Left*/  &&SensorValue[AutonDial]  >2400/
        displayLCDCenteredString( 1/*0=Top Line, 1=Bottom Line*/  , "2nd Cub Right|R"/*Text To Be Displayed*/ ); //Displ
      }
      else if(SensorValue[AutonDial]  >3200/*SensorValue Starting W/ 0 From the Left*/  ){//Farthest right side, have no
        displayLCDCenteredString( 1/*0=Top Line, 1=Bottom Line*/  , "No Auton|R"/*Text To Be Displayed*/  ); //Display "
      }
    }
    ////////////////////////////////////////////////////////////////////////////////////////////

  }
}

      ////////////////////////////////////////////////////////////////////////////////
     //
    //                                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.
```

File: E:\delta 3.0 program for robot with pre auton.c

```
 //
////////////////////////////////////////////////////////////////////////////////
task usercontrol()
  {
  bLCDBacklight=true;//Turn the LCD backlight on to show selected autons//
  startTask(showautons);//Start task "showautons" in order to display the currently selected auton//
  while (true)//Create an infinite-running loop//
  {

      ////////////////////////////////////////////////
     //Lift and Claw Control(A tad bit more advanced)//
    ////////////////////////////////////////////////

      /////////////////
     //Dump Position//
    /////////////////////////////////////////////////
    if(vexRT[ Btn6U/*Corresponding Button*/ ]== 1/*1=Pressed, 0=Released*/  ){//Btn 6U(Right-top bumper)//
      if((-SensorValue[LiftEnc])<110){///If the lift is below dump position, keep claw closed//
      motor[ClawL]=((SensorValue[LeftClawPot] -2700/*Width From Starting Pos.*/ ) /5/*Sensitivity*/ );//Left claw being
      motor[ClawR]=((SensorValue[RightClawPot]  -2700/*Width From Starting Pos.*/ ) /5/*Sensitivity*/ );//Right claw bei
      }
      else{//When at or above dump position, open claw//
        motor[ClawL]=((SensorValue[LeftClawPot] -1200/*Width From Starting Pos.*/ ) /10/*Sensitivity*/  ) ^2/*Slope*/ ;/
        motor[ClawR]=((SensorValue[RightClawPot]  -1200/*Width From Starting Pos.*/ ) /10/*Sensitivity*/  ) ^2/*Slope*/
      }
      //Lift PI control to position//
      motor[LiftL1]=(SensorValue[LiftEnc] +145/*Height From Starting Pos.*/ ) *4/*Sensitivity*/ ;//Left lift being ran b
      motor[LiftL2]=(SensorValue[LiftEnc] +145/*Height From Starting Pos.*/ ) *4/*Sensitivity*/ ;//Left lift being ran b
      motor[LiftR1]=(SensorValue[LiftEnc] +145/*Height From Starting Pos.*/ ) *4/*Sensitivity*/ ;//Right lift being ran
      motor[LiftR2]=(SensorValue[LiftEnc] +145/*Height From Starting Pos.*/ ) *4/*Sensitivity*/ ;//Right lift being ran
    }
    /////////////////////////////////////////////////

      ////////////////////
     //Hold Up Position//
    /////////////////////////////////////////////////
    else if(vexRT[  Btn6D/*Corresponding Button*/ ]== 1/*1=Pressed, 0=Released*/  ){//Btn 6D(Right-bottom bumper)//
      //Make sure claw is closed//
      motor[ClawL]=((SensorValue[LeftClawPot] -2700/*Width From Starting Pos.*/ ) /5/*Sensitivity*/ );//Left claw being
      motor[ClawR]=((SensorValue[RightClawPot]  -2700/*Width From Starting Pos.*/ ) /5/*Sensitivity*/ );//Right claw bei
      //Lift PI control to position//
      motor[LiftL1]=(SensorValue[LiftEnc] +50/*Height From Starting Pos.*/  ) *4/*Sensitivity*/ ;//Left lift being ran b
      motor[LiftL2]=(SensorValue[LiftEnc] +50/*Height From Starting Pos.*/  ) *4/*Sensitivity*/ ;//Left lift being ran b
      motor[LiftR1]=(SensorValue[LiftEnc] +50/*Height From Starting Pos.*/  ) *4/*Sensitivity*/ ;//Right lift being ran
      motor[LiftR2]=(SensorValue[LiftEnc] +50/*Height From Starting Pos.*/  ) *4/*Sensitivity*/ ;//Right lift being ran
```

```
    }
    ////////////////////////////////////////////////

      ////////////////////////////////
     //Disable Lift and Claw motors//
    ////////////////////////////////////////////////
    else if(vexRT[  Btn7R/*Corresponding Button*/ ]== 1/*1=Pressed, 0=Released*/  ){//Btn 7R(Left buttonpad, right butto
      motor[ClawL]=0;//Stop Motor//
      motor[ClawR]=0;//Stop Motor//
      motor[LiftL1]=0;//Stop Motor//
      motor[LiftL2]=0;//Stop Motor//
      motor[LiftR1]=0;//Stop Motor//
      motor[LiftR2]=0;//Stop Motor//
    }
    ////////////////////////////////////////////////

      /////////////////
     //Down Position//
    ////////////////////////////////////////////////
    else{//When no lift buttons are pressed//
      if(vexRT[ Btn5U/*Corresponding Button*/ ]== 1/*1=Pressed, 0=Released*/  ){//Btn 5U(Left-top bumper)//
      motor[ClawL]=((SensorValue[LeftClawPot] -2700/*Width From Starting Pos.*/ ) /5/*Sensitivity*/ );//Left claw being
      motor[ClawR]=((SensorValue[RightClawPot]  -2700/*Width From Starting Pos.*/ ) /5/*Sensitivity*/ );//Right claw bei
      }
      else{//When no button is pressed, open the claw//
        motor[ClawL]=((SensorValue[LeftClawPot] -1200/*Width From Starting Pos.*/ ) /10/*Sensitivity*/  ) ^2/*Slope*/ ;/
        motor[ClawR]=((SensorValue[RightClawPot]  -1200/*Width From Starting Pos.*/ ) /10/*Sensitivity*/  ) ^2/*Slope*/
      }
      //Lift PI control to position//
      motor[LiftL1]=(SensorValue[LiftEnc] -2/*Height From Starting Pos.*/ ) *4/*Sensitivity*/ ;//Left lift being ran by
      motor[LiftL2]=(SensorValue[LiftEnc] -2/*Height From Starting Pos.*/ ) *4/*Sensitivity*/ ;//Left lift being ran by
      motor[LiftR1]=(SensorValue[LiftEnc] -2/*Height From Starting Pos.*/ ) *4/*Sensitivity*/ ;//Right lift being ran by
      motor[LiftR2]=(SensorValue[LiftEnc] -2/*Height From Starting Pos.*/ ) *4/*Sensitivity*/ ;//Right lift being ran by
    }
    ////////////////////////////////////////////////

      ////////////////////////////////////////
     //Drivetrain Code(Simple and easy)      //
    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////
    motor[Left1]=vexRT[ Ch3/*Corresponding Channel(F/B)*/ ]+vexRT[  Ch1/*Corresponding Channel(L/R)*/ ];     //The front
    motor[Left2]=vexRT[ Ch3/*Corresponding Channel(F/B)*/ ]+vexRT[  Ch1/*Corresponding Channel(L/R)*/ ];    //The back-l
    motor[Right1]=vexRT[  Ch3/*Corresponding Channel(F/B)*/ ]-vexRT[  Ch1/*Corresponding Channel(L/R)*/ ]; //The front-r
    motor[Right2]=vexRT[  Ch3/*Corresponding Channel(F/B)*/ ]-vexRT[  Ch1/*Corresponding Channel(L/R)*/ ];//The back-rig
    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  }
```

```
}
```