File: C:\Users\Citi\Documents\CitiGalsCode\CitiGalsV6RobotC.c

```
#pragma config(Motor,  port1,           leftBack,      tmotorVex393_HBridge, ope
#pragma config(Motor,  port2,           leftTop,       tmotorVex393_MC29, openLo
#pragma config(Motor,  port3,           rightBack,     tmotorVex393_MC29, openLo
#pragma config(Motor,  port4,           mobileLift,    tmotorVex393_MC29, openLo
#pragma config(Motor,  port5,           leftScissor,   tmotorVex393_MC29, openLo
#pragma config(Motor,  port6,           rightScissor,  tmotorVex393_MC29, openLo
#pragma config(Motor,  port7,           arm,           tmotorVex393_MC29, openLo
#pragma config(Motor,  port8,           claw,          tmotorVex393_MC29, openLo
#pragma config(Motor,  port10,          rightTop,      tmotorVex393_HBridge, ope
//*!!Code automatically generated by 'ROBOTC' configuration wizard

/*---------------------------------------------------------------------------*/
/*                                                                           */
/*        Description: Competition template for VEX EDR                      */
/*                                                                           */
/*---------------------------------------------------------------------------*/

// This code is for the VEX cortex platform
#pragma platform(VEX2)

// Select Download method as "competition"
#pragma competitionControl(Competition)

//Main competition background code...do not modify!
#include "Vex_Competition_Includes.c"

#define C1LX vexRT[Ch1]
#define C1LY vexRT[Ch2]
#define C1RX vexRT[Ch4]

#define FULLPWR 127
#define HALFPWR 63


void sideDrive(int drivespd_direction, int time)
{
  //this function drives to the left or the right at a chosen speed
  motor[rightTop] = drivespd_direction;
  motor[leftBack] = drivespd_direction;
  wait1Msec(time); //1000 miliseconds is 1 second
  stopAllMotors();
}

void frontDrive(int drivespd_direction, int time)
{
  //this function drives forward of backward at a chosen speed
  motor[rightBack] = drivespd_direction;
  motor[leftTop] = drivespd_direction;
  wait1Msec(time);
}

void turnNinetyNinety(int drivespd_direction, int time)
{
  motor[rightBack] = drivespd_direction;
  motor[leftBack] = drivespd_direction;
  motor[rightTop] = drivespd_direction;
  motor[leftTop] = drivespd_direction;
  wait1Msec(2000); //test to get a 180 degree turn
}
```

```
void scissorLift (int spd)
{
  motor[leftScissor] = spd;
  motor[rightScissor] = spd;
  wait1Msec(1000); //still have to find out how long it takes to lift the scissc
}

void mobileLiftDrive (int spd)
{
  motor[leftTop] = spd;
  motor[rightBack] = spd;
  int count = 0;
  while (count <1 && motor[leftTop] == spd && motor[rightBack] == spd)
  {
    motor[mobileLift] = FULLPWR;
    wait1Msec(2000);
    count--;
  }
}

/*---------------------------------------------------------------------------*/
/*                           Pre-Autonomous Functions                        */
/*                                                                           */
/*  You may want to perform some actions before the competition starts.      */
/*  Do them in the following function.  You must return from this function   */
/*  or the autonomous and usercontrol tasks will not be started.  This       */
/*  function is only called once after the cortex has been powered on and    */
/*  not every time that the robot is disabled.                               */
/*---------------------------------------------------------------------------*/

void pre auton()
{
  // Set bStopTasksBetweenModes to false if you want to keep user created tasks
  // running between Autonomous and Driver controlled modes. You will need to
  // manage all user created tasks if set to false.
  bStopTasksBetweenModes = true;

  // Set bDisplayCompetitionStatusOnLcd to false if you don't want the LCD
  // used by the competition include file, for example, you might want
  // to display your team name on the LCD in this function.
  // bDisplayCompetitionStatusOnLcd = false;

  // All activities that occur before the competition starts
  // Example: clearing encoders, setting servo positions, ...
}

/*---------------------------------------------------------------------------*/
/*                                                                           */
/*                              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()
{
```

```
  // .....................................................................
  sideDrive(

  // .....................................................................

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

/*---------------------------------------------------------------------------*/
/*                                                                           */
/*                              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.

    //motor power levels range from -127 (full reverse) to 0 (stopped) to 127 (f
    // .....................................................................
    //sets each of the wheel/chassis motors to a channel on the joystick
    motor[leftTop] = -C1LY - C1LX - C1RX;
    motor[rightTop] =  C1LY - C1LX - C1RX;
    motor[rightBack] =  C1LY + C1LX - C1RX;
    motor[leftBack] = -C1LY + C1LX - C1RX;

    //claw moves one direction when button 7D pressed
    if (vexRT[Btn7D] == 1)
    {
      motor[claw] = HALFPWR; //motor forward for about half power
    }
    else if (vexRT[Btn7L] == 1)
    {
     motor[claw] = -HALFPWR; //claw moves in opposite direction when button 7L p
    }
    else
    {
      motor [claw] = 0;
    }

    //scissor lift moves up when 6U pressed
    if (vexRT [Btn6U] == 1)
    {
      motor[leftScissor] = -HALFPWR;
      motor[rightScissor] = HALFPWR;
    }
    else if (vexRT[Btn6D] == 1) //if 6D is pressed then scissor lift down
    {
```

```
      motor[leftScissor] = HALFPWR;
      motor[rightScissor] = -HALFPWR;
    }
    else
    {
      motor[leftScissor] = 0;
      motor[rightScissor] = 0;
    }

    //goal lift moves up when 5U pressed
    if (vexRT [Btn5U] == 1)
    {
      motor[mobileLift] = HALFPWR;
    }
    else if (vexRT [Btn5D] == 1)
    {
      motor[mobileLift] = -HALFPWR;
    }
    else
    {
      motor[mobileLift] = 0;
    }
    // .....................................................................

    // Remove this function call once you have "real" code.


    //UserControlCodePlaceholderForTesting();
  }
}
```