#pragma config(Sensor, in1, Power, sensorAnalog)

#pragma config(Sensor, dgtl1, rightEncoder, sensorQuadEncoder)

#pragma config(Sensor, dgtl3, leftEncoder, sensorQuadEncoder)

#pragma config(Motor, port1, , tmotorVex393\_HBridge, openLoop)

#pragma config(Motor, port2, frontRight, tmotorVex393\_MC29, openLoop)

#pragma config(Motor, port3, backRight, tmotorVex393\_MC29, openLoop)

#pragma config(Motor, port4, frontLeft, tmotorVex393\_MC29, openLoop, reversed)

#pragma config(Motor, port5, backLeft, tmotorVex393\_MC29, openLoop, reversed)

#pragma config(Motor, port6, clawMotor, tmotorVex393\_MC29, openLoop)

#pragma config(Motor, port7, armMotor, tmotorVex393\_MC29, openLoop, encoderPort, dgtl1)

#pragma config(Motor, port8, arm2Motor, tmotorVex393\_MC29, openLoop, encoderPort, dgtl3)

#pragma config(Motor, port9, liftMotor, tmotorVex393\_MC29, openLoop)

//\*!!Code automatically generated by 'ROBOTC' configuration wizard !!\*//

/\*

Code Chooser

ROBOTC on VEX 2.0 Cortex

This program uses the Display functions of ROBOTC on the VEX 2.0 Cortex platform.

It allows the user to choose from 4 different pieces of code using the left and right buttons

on the VEX LCD. Once the center button is pressed, the code corresponding with the choice is run.

This code can be adapted for competition based settings - just place the code for the first

switch case in the pre\_auton function, and the code for the second switch in the autonomous task.

Replace the basic movement behaviors in the second switch with your own autonomous routines.

ROBOT CONFIGURATION

MOTORS & SENSORS:

[I/O Port] [Name] [Type] [Description]

UART Port 2 none VEX LCD VEX LCD Screen

Motor Port 2 rightMotor VEX 3-wire module Right side motor

Motor Port 3 leftMotor VEX 3-wire module Left side motor

\*/

const short leftButton = 1;

const short centerButton = 2;

const short rightButton = 4;

//Wait for Press--------------------------------------------------

void waitForPress()

{

while(nLCDButtons == 0){}

wait1Msec(5);

}

//----------------------------------------------------------------

//Wait for Release------------------------------------------------

void waitForRelease()

{

while(nLCDButtons != 0){}

wait1Msec(5);

}

//----------------------------------------------------------------

task main()

{

bLCDBacklight = true; // Turn on LCD Backlight

//Declare count variable to keep track of our choice

int count = 0;

//------------- Beginning of User Interface Code ---------------

//Clear LCD

clearLCDLine(0);

clearLCDLine(1);

//Loop while center button is not pressed

while(nLCDButtons != centerButton)

{

//Switch case that allows the user to choose from 4 different options

switch(count){

case 0:

//Display first choice

displayLCDCenteredString(0, "Autonomous 1");

displayLCDCenteredString(1, "< Enter >");

waitForPress();

//Increment or decrement "count" based on button press

if(nLCDButtons == leftButton)

{

waitForRelease();

count = 3;

}

else if(nLCDButtons == rightButton)

{

waitForRelease();

count++;

}

break;

case 1:

//Display second choice

displayLCDCenteredString(0, "Autonomous 2");

displayLCDCenteredString(1, "< Enter >");

waitForPress();

//Increment or decrement "count" based on button press

if(nLCDButtons == leftButton)

{

waitForRelease();

count--;

}

else if(nLCDButtons == rightButton)

{

waitForRelease();

count++;

}

break;

case 2:

//Display third choice

displayLCDCenteredString(0, "Autonomous 3");

displayLCDCenteredString(1, "< Enter >");

waitForPress();

//Increment or decrement "count" based on button press

if(nLCDButtons == leftButton)

{

waitForRelease();

count--;

}

else if(nLCDButtons == rightButton)

{

waitForRelease();

count++;

}

break;

case 3:

//Display fourth choice

displayLCDCenteredString(0, "Autonomous 4");

displayLCDCenteredString(1, "< Enter >");

waitForPress();

//Increment or decrement "count" based on button press

if(nLCDButtons == leftButton)

{

waitForRelease();

count--;

}

else if(nLCDButtons == rightButton)

{

waitForRelease();

count = 0;

}

break;

default:

count = 0;

break;

}

}

//------------- End of User Interface Code ---------------------

//------------- Beginning of Robot Movement Code ---------------

//Clear LCD

clearLCDLine(0);

clearLCDLine(1);

//Switch Case that actually runs the user choice

switch(count){

case 0:

//If count = 0, run the code correspoinding with choice 1

displayLCDCenteredString(0, "Autonomous 1");

displayLCDCenteredString(1, "is running!");

wait1Msec(2000); // Robot waits for 2000 milliseconds

// Move forward at full power for 3 seconds

motor[clawMotor] = -127;

wait(1.15);

stopMotor(clawMotor);

motor[frontRight] = -127; // Motor on port2 is run at full (127) power forward

motor[backRight] = -127;

motor[frontLeft] = -127;

motor[backLeft] = -127;

wait(1.89);

stopMotor(frontRight);

stopMotor(backRight);

stopMotor(frontLeft);

stopMotor(backLeft);

motor[arm2Motor] = 127;

motor[armMotor] = -127;

wait(1);

stopMotor(arm2Motor);

stopMotor(armMotor);

motor[clawMotor] = 127;

wait(1);

stopMotor(clawMotor);

motor[arm2Motor] = -127;

motor[armMotor] = 127;

wait(1);

stopMotor(arm2Motor);

stopMotor(armMotor);

motor[frontRight] = 127; // Motor on port2 is run at full (127) power forward

motor[backRight] = 127;

motor[frontLeft] = 127;

motor[backLeft] = 127;

wait(1.89);

stopMotor(frontRight);

stopMotor(backRight);

stopMotor(frontLeft);

stopMotor(backLeft);

// Motor on port3 is run at full (127) power forward

wait1Msec(3000); // Robot runs previous code for 3000 milliseconds before moving on

break;

case 1:

//If count = 1, run the code correspoinding with choice 2

displayLCDCenteredString(0, "Autonomous 2");

displayLCDCenteredString(1, "is running!");

wait1Msec(2000); // Robot waits for 2000 milliseconds

// Move reverse at full power for 3 seconds

motor[frontRight] = 127; // Motor on port2 is run at full (127) power forward

motor[backRight] = 127;

motor[frontLeft] = 127;

motor[frontRight] = 127;

// Motor on port3 is run at full (-127) power reverse

wait1Msec(3000); // Robot runs previous code for 3000 milliseconds before moving on

break;

case 2:

//If count = 2, run the code correspoinding with choice 3

displayLCDCenteredString(0, "Autonomous 3");

displayLCDCenteredString(1, "is running!");

wait1Msec(2000); // Robot waits for 2000 milliseconds

//Turn right for 3 seconds

motor[frontRight] = 127; // Motor on port2 is run at full (127) power forward

motor[backRight] = 127;

motor[frontLeft] = 127;

motor[frontRight] = 127;

break;

case 3:

//If count = 3, run the code correspoinding with choice 4

displayLCDCenteredString(0, "Autonomous 4");

displayLCDCenteredString(1, "is running!");

wait1Msec(2000); // Robot waits for 2000 milliseconds

//Turn left for 3 seconds

motor[frontRight] = 127; // Motor on port2 is run at full (127) power forward

motor[backRight] = 127;

motor[frontLeft] = 127;

motor[backLeft] = 127;

wait(2);

stopMotor(frontRight);

stopMotor(backRight);

stopMotor(frontLeft);

stopMotor(backLeft);

motor[liftMotor] = -127;

wait(2);

stopMotor(liftMotor);

wait1Msec(3000); // Robot runs previous code for 3000 milliseconds before moving on

default:

displayLCDCenteredString(0, "No valid choice");

displayLCDCenteredString(1, "was made!");

bLCDBacklight = true; // Turn on LCD Backlight

string mainBattery, expanderBattery;

while(true) // An infinite loop to keep the program running until you terminate it

{

clearLCDLine(0); // Clear line 1 (0) of the LCD

clearLCDLine(1); // Clear line 2 (1) of the LCD

displayLCDString(0, 0, "Primary: ");

sprintf(mainBattery, "%1.2f%c", nImmediateBatteryLevel/1000.0,'V'); //Build the value to be displayed

displayNextLCDString(mainBattery);

string expanderBattery;

//\* Expander Battery Displaying \*//

if(SensorValue[Power]<2240){ //If Expander Battery is below 8V (2240 Value)//

clearLCDLine(1); //Clear LCD line 1 (Bottom Line)//

displayLCDString(1, 0, "Exp. Low: "); //Display Expander Low//

sprintf(expanderBattery, "%1.2f%c", SensorValue[Power]/280.0, 'V'); //Build the value to be displayed//

displayNextLCDString(expanderBattery); //Display the string "expanderBattery"//

//Short delay for the LCD refresh rate

wait1Msec(100);

SensorValue[rightEncoder] = 0; // Set the encoder so that it starts counting at 0

SensorValue[leftEncoder] = 0; // Set the encoder so that it starts counting at 0

//Loop Forever

while(1 == 1)

{

//Remote Control Commands

motor[frontRight] = vexRT[Ch3]+ vexRT[Ch1];

motor[backRight] = vexRT[Ch3]+ vexRT[Ch1];

motor[frontLeft] = vexRT[Ch3]- vexRT[Ch1];

motor[backLeft] = vexRT[Ch3]- vexRT[Ch1];

//Arm Control

if(vexRT[Btn8R] == 1)

{

motor[liftMotor] = -127;

}

else if(vexRT[Btn8D] == 1)

{

motor[liftMotor] = 127;

}

else

{

motor[liftMotor] = 0;

}

//Arm Control

if(vexRT[Btn6D] == 1)

{

motor[armMotor] = 97;

motor[arm2Motor] = -127;

}

else if(vexRT[Btn6U] == 1)

{

motor[armMotor] = -97;

motor[arm2Motor] = 127;

}

else

{

motor[armMotor] = 0;

motor[arm2Motor] = 0;

}

if(vexRT[Btn5U] == 1)

{

motor[clawMotor] = -127;

}

else if(vexRT[Btn5D] == 1)

{

motor[clawMotor] = 127;

}

else

{

motor[clawMotor] = 0;

//Arm Control

if(vexRT[Btn6D] == 1)

{

if(SensorValue[rightEncoder] == SensorValue[leftEncoder]) // If rightEncoder has counted the same amount as leftEncoder:

{

// Move Forward

motor[armMotor] = 97; // Right Motor is run at power level 80

motor[arm2Motor] = -127; // Left Motor is run at power level 80

}

else if(SensorValue[rightEncoder] > SensorValue[leftEncoder]) // If rightEncoder has counted more encoder counts

{

// Turn slightly right

motor[armMotor] = 97; // Right Motor is run at power level 60

motor[arm2Motor] = -127; // Left Motor is run at power level 80

}

else // Only runs if leftEncoder has counted more encoder counts

{

// Turn slightly left

motor[armMotor] = 97; // Right Motor is run at power level 80

motor[arm2Motor] = -127; // Left Motor is run at power level 60

}

}

else if(vexRT[Btn6U] == 1)

{

if(SensorValue[rightEncoder] == SensorValue[leftEncoder]) // If rightEncoder has counted the same amount as leftEncoder:

{

// Move Forward

motor[armMotor] = -97; // Right Motor is run at power level 80

motor[arm2Motor] = 127; // Left Motor is run at power level 80

}

else if(SensorValue[rightEncoder] > SensorValue[leftEncoder]) // If rightEncoder has counted more encoder counts

{

// Turn slightly right

motor[armMotor] = -97; // Right Motor is run at power level 60

motor[arm2Motor] = 127; // Left Motor is run at power level 80

}

else // Only runs if leftEncoder has counted more encoder counts

{

// Turn slightly left

motor[armMotor] = -97; // Right Motor is run at power level 80

motor[arm2Motor] = 127; // Left Motor is run at power level 60

}

}

else

{

// Turn slightly left

motor[armMotor] = 0; // Right Motor is run at power level 80

motor[arm2Motor] = 0; // Left Motor is run at power level 60

}

}

//------------- End of Robot Movement Code -----------------------

}

}

}

}

}