K Force

The Kristin Robotics Team

Introductory Programming Tutorial 2014

For use with the teams squarebot training robots.

K Force – Squarebot Programming Course - 2014

Robot moves forward for two seconds

```
#pragma config(Motor,  port1,         leftwheel,     tmotorVex393HighSpeed, openLoop,)

#pragma config(Motor,  port2,         rightwheel,    tmotorVex393HighSpeed, openLoop, reversed )

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

//botgear robot no controller move forwards for 2 seconds

task main()                    //designates start of the programme

{                              //programme start bracket and white space

        wait1Msec(3000);       //allows time to switch the robot on and put it on the ground

        motor[leftwheel]=60;   //sets the left motor to 60

        motor[rightwheel]=60;  //sets the right motor to 60 - robot starts going forward

        wait1Msec(2000);       //wait 2 seconds - robot moves forwards for 2 seconds

        motor[leftwheel]=0;    //sets the left motor to 0

        motor[rightwheel]=0;   //sets the right motor to 0 - robots stops

}                              //programme end bracket and white space
```

It's good to start the programme with a comment briefly describing what it does

Use white space, comments and indents. This makes it easier to fault find, amend and understand your programme.

Important tips

The programme could be written as below with no structuring, white space or comments and it would still work. As you can see it is much harder to read and very difficult to fault find.

```
#pragma config(Motor,  port1, leftwheel, tmotorVex393HighSpeed, openLoop,) #pragma config(Motor,  port2, rightwheel, tmotorVex393HighSpeed, openLoop, , reversed ) task main(){motor[leftwheel]=60; motor[rightwheel]=60;wait1Msec(2000);motor[leftwheel]=0;motor[rightwheel]=0;}
```

You are advised to lay your programme out with structure, white space, comments for each line of code and a logical file name. /* A forward slash followed by an asterisk will isolate all the text after them in a paragraph so they are not read as code. Use these to make comments on your code and finish the paragraph with an asterisk and forward slash to complete the statement*/

// Two forward slashes will isolate the remaining text on one line only. No closing statement is required

Save your programmes regularly!!

Suggested file name structure: robot_programme_type_date_number

Examples

botgear_nc_fwd2secs_200214_2	(Botgear robot, no controller, forward for 2 seconds 20 Feb 2014, version 2)
2919a_kiwi chall_210514_8	(Team 2919A, Kiwi Challenge Competition, 21st May 2014, version 8)
marvin_3 sens_linetrk_300812-12	(Marvin, 3 sensor line tracking, 30 August 2012, version 12)

When you have completed your programme compile and down load it using the F5 key.
(Your programme must be correct!! All colons brackets and commands must be right.
The compiler will give you suggestions for errors and corrections)
Use the robot on and off switch to start your program or you can also use the start button in the Programme Debug window. Try using the stepping function which works through your programme step by step. This is very handy for fault finding.

Safety

1. **Ensure all wires are properly tied up before operating a robot**
   (Wires can get caught in gears and mechanisms which can cut them or cause short circuits resulting in expensive damage)
2. **Ensure you and your team mates are clear before operating your robot** (Fingers can get crushed in gears and mechanisms as well as eye hazards which can occur with moving parts such as lifts)
3. **Do not operate your robot on a bench where it can possibly fall** (If you are testing your robot on a bench ensure that someone is restraining it or it is on blocks. Robots can automatically go into autonomous routines on switch on. Falling onto the floor will almost certainly result in serious damage)

Robot moves forward continuously

```
#pragma config(Motor,  port1,         leftwheel,    tmotorVex393HighSpeed, openLoop)

#pragma config(Motor,  port2,         rightwheel,    tmotorVex393HighSpeed, openLoop, reversed )

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

//botgear robot no controller continuous forward

task main()                 //designates start of the programme

{                           //programme start bracket and white space

wait1Msec(3000);            //allows time to switch the robot on and put it on the ground

while(true)                 //While true, which is continuously, loop around between the brackets

{                           //loop start

motor[leftwheel]=60;        //sets the left motor to 60

motor[rightwheel]=60;       //sets the right motor to 60

}                           //loop end – go back to the start of the loop

}                           //designates end of the programme
```

Loops are very powerful tools in programming. The 'while(true)' loop will continue looping around indefinitely as there is nothing to make it false. As a result the programme is continually setting the motor speeds to the joystick settings with every loop. It does this thousands of times every second

Do not let your robot bash into anything

Robot moves forward when switched

```
#pragma config(Sensor, in2,    bumpswitch,     sensorTouch)

#pragma config(Motor,  port1,          leftwheel,     tmotorVex393HighSpeed, openLoop)

#pragma config(Motor,  port2,          rightwheel,    tmotorVex393HighSpeed, openLoop, reversed)

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

//botgear no controller forward and stop controlled by a bumper switch connected to A/D port 1

task main()         //designates start of the programme

{                   //designates start of programme

int state=0;        //creates and stores an integer called state which equals zero

                    //this is used to determine whether the motors are to be switched on or off

                    //when state=1 the robot will be running, when state=0 the robot will be stopped

int value=0;        //creates and stores an integer called value which equals zero

                    //this is used to record an activation of the bumper switch

                    //when switch contact is made (switch being held down)value will equal one

int oldvalue=0;     //creates and stores an integer called oldvalue which equals zero

                    //this variable is used to determine if the switch operation is changing the state


while(true)                                 //while true (continuous loop)

{                                           //start of while bracket - programme continuously loops to here

          value=(SensorValue(bumpswitch)); //sets 'value' to 1 if bump switch is pressed

          if(value==1 && oldvalue==0)       //bumpswitch is depressed  and old value equals zero

                                            //switch was not pressed during the previous loop

                    {                       //start of if statement

                    state = (1-state);      //change state to 1 or 0 so motors switch on or off (state changes)

                    wait1Msec(300);         //this is to 'debounce' the switch (ask your teacher about this)

                    }                       //end of if statement

          oldvalue = value;  //this will stop the state changing if the bumpswitch is pushed and held down

          if((state)==1)                    //if state equals 1

                    {                       //start of if statement
                    motor[leftwheel]=60;       //sets the left motor to 60

                    motor[rightwheel]=60;      //sets the right motor to 60 - robot starts going forward

                    }                          //end of if statement

          else

                    {                          //start of else statement

                    motor[leftwheel]=0;        //sets the left motor to 0

                    motor[rightwheel]=0;       //sets the right motor to 0 - robots stops

                    }                          //end of else statement

}                                              //end of while loop - programme repeats

}                                              //end of programme
```

This programme uses variables which we set at the beginning. Variables are very powerful tools in programming. We can use them to name things and remember things.

If we were controlling the robot with a simple on and off switch we could control the robot using a simple loop but because we are using a pushbutton switch we need to know whether the previous activation of the switch was to turn the robot on or off. We can use variables for this.

Do not let your robot bash into anything. Turn it off and on using the switch..

Robot moves forward controlled by a switch and potentiometer

```
#pragma config(Sensor, in1,   pot,  sensorPotentiometer)

#pragma config(Sensor, in2,  bumpswitch,  sensorTouch)

#pragma config(Motor,  port1,  leftwheel,  tmotorVex393HighSpeed, openLoop)

#pragma config(Motor,  port2, rightwheel, tmotorVex393HighSpeed, openLoop, reversed)

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

//botgear no controller forward and stop controlled by a bumper switch connected to A/D port 2

//potentiometer connected to A/D port 1 controls robot speed and forward and reverse

task main()

{                       //designates start of programme

int state=0;            //creates and stores an integer called state which equals zero

                        //this is used to determine whether the motors are to be switched on or off

                        //when state=1 the robot will be running, when state=0 the robot will be stopped

int value=0;            //creates and stores an integer called value which equals zero

                        //this is used to record an activation of the bumper switch

                        //when switch contact is made (switch being held down)value will equal one

int oldvalue=0;         //creates and stores an integer called oldvalue which equals zero

                        //this variable is used to determine if the switch operation is changing the state

while(true)             //while true

{                       //start of while bracket - programme continuously loops back to this point

value=(SensorValue(bumpswitch)); //sets 'value' to 1 if bump switch is pressed

if(value==1 && oldvalue==0)      //bumpswitch is depressed

                                 //switch was not pressed during the previous loop

{                                //start of if statement

state = (1-state);               //change state to 1 or 0 so motors switch on or off (state must change)

wait1Msec(300);                  //this is to 'debounce' the switch (ask your teacher about this)

}                                //end of if statement

     oldvalue = value;  //this will stop the state changing if the bumpswitch is pushed and held down

     if((state)==1)                  //if state equals 1
{                                       //start of if statement
          if(SensorValue(pot)<512)  //if the potentiometer (1-1024) is less than half way (512)
          {                             //start of if statement
          motor[leftwheel]=-128+((SensorValue(pot))/4); //left motor set to increasing -ve setting
          motor[rightwheel]=-128+((SensorValue(pot))/4);//right motor set to increasing -ve setting
          }                             //end of if statement
          else
          {                             //start of else statement
          motor[leftwheel]=-128+((SensorValue(pot))/4); //left motor set to increasing +ve setting
          motor[rightwheel]=-128+((SensorValue(pot))/4);//right motor set to increasing +ve setting
          }                             //end of else statement
}                                       //end of if statement
else
{                                       //start of else statement
motor[leftwheel]=0;                     //sets the left motor to 0
motor[rightwheel]=0;                    //sets the right motor to 0 - robots stops
}                                       //end of else statement
}                                       //end of while loop - programme repeats
}                                       //end of programme}
```

This programme uses a potentiometer connected to Analogue/Digital input 2.

This program also shows us how we can use basic maths in our programming to use an analogue sensor whose value is from 0-1024 and convert it to an analogue value from -127 - +127 which we can use to control the motors forward and in reverse.

Connect a potentiometer to A/D port 1 and put your robot on blocks so the wheels can spin freely..

A potentiometer has a limited range of rotation less than 360 degrees. It is used most commonly to measure angles. Do not connect a potentiometer on to a motor shaft or shaft that rotates more than 270 degrees otherwise you could destroy it.

Robot turns to the right

```
#pragma config(Motor,  port1,   leftwheel,   tmotorVex393HighSpeed, openLoop)

#pragma config(Motor,  port2,   rightwheel,   tmotorVex393HighSpeed, openLoop, reversed)

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


//botgear robot no controller move forwards and turns right in increasingly sharper turns

task main()                  //designates start of the programme

{                            //programme start bracket and white space

wait1Msec(3000);             //wait for 3 seconds after switch on for programme to start

motor[leftwheel]=60;         //sets the left motor to 60

motor[rightwheel]=60;        //sets the right motor to 60 - robot starts going forward

wait1Msec(2000);             //wait 2 seconds - robot moves forwards for 2 seconds

motor[leftwheel]=80;         //sets the left motor to 80

motor[rightwheel]=40;        //sets the right motor to 40 - robots turns gently to the right

wait1Msec(2000);             //wait 2 seconds - robot gently turns to the right for 2 second

motor[leftwheel]=80;         //sets the left motor to 80

motor[rightwheel]=00;        //sets the right motor to 00 - robot turns on its right wheel for 2 seconds

wait1Msec(2000);             //wait 2 seconds - robot turns on its right wheel for 2 seconds

motor[leftwheel]=80;         //sets the left motor to 80

motor[rightwheel]=-80;       //sets the right motor to -80 - robots spins on axis  to the right

wait1Msec(2000);             //wait 2 seconds - robot spins to the right on its axis for 2 seconds

}                            //programme end bracket
```

This programme shows how we can control the robots direction using the power to the two motors. The robot turns gently of on its axis using reverse power settings.

Do not let your robot bash into anything as it turns in increasingly tight turns.

Robot controlled by a controller

```
#pragma config(Motor, port1,  leftwheel, tmotorVex393HighSpeed, openLoop)

#pragma config(Motor, port2,  rightwheel, tmotorVex393HighSpeed, openLoop, reversed)

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

//botgear robot tank drive contol

task main()                        //designates start of the programme

{                                  //programme start bracket

while(true)                        //programme continually loops back to while statement

{                                  //start of loop

bIfiAutonomousMode = false;        //no autonomous mode

motor[leftwheel]=vexRT(Ch3);       //set the left motor to the left joystick vertical channel

motor[rightwheel]=vexRT(Ch2);      //set the right motor to the right joystick vertical channel

}                                  //end of loop

}                                  //end of programme
```

You can assign any channel or button on the controller to any motor or servo. The channel allocations can be found in the inventors guide or on line.

Do not let your robot bash into anything as you drive it around.

Robot controlled by a controller with autonomous routine

```
#pragma config(Sensor, in2,    bumpswitch,     sensorTouch)

#pragma config(Motor,  port1,          leftwheel,     tmotorVex393HighSpeed, openLoop)

#pragma config(Motor,  port2,          rightwheel,    tmotorVex393HighSpeed, openLoop, reversed)

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

//botgear competition control simulation

task main()

{                                       //programme start bracket

bIfiAutonomousMode = false;             //no automatic autonomous mode

int auton=0;                            //creates and stores an integer called auton which equals zero

                                        //records whether the autonomous bumpswitch has been pressed

                                        //when pressed the robot excecutes 20 seconds of autonomous code

int drivercont=0;                       //creates and stores an integer called drivercont which equals zero

                                        //used to start the 1 minute 40 second driver control period

                                        //the robot will stop at the end of this period

while(auton<1)                          //while autonomous button is not pressed

          {                             //start of while statement

          ClearTimer(T1);               //timer T1 is set to zero

          auton=(SensorValue(bumpswitch));//autonomous bumpswitch is checked

          }                             //end of while statement

while(time100[T1]<200)                  //while timer is less than 20 seconds(time100=time in 100msec intervals)

          {                             //start of while statement - place your autonomous code in here

                    motor[leftwheel]=60;//sets the left motor to 60

                    motor[rightwheel]=20;//sets the right motor to 20 - robot turns to the right for 20 seconds

          }                                       //end of while statement

while(drivercont==0)                              //while right button (Ch 6)  on controller is not pressed

          {                                       //start of while loop

                    motor[leftwheel]=0;           //sets the left motor to 0

                    motor[rightwheel]=0;          //sets the right motor to 0 - robot stopped
```

```
                ClearTimer(T2);             //timer T2 is set to zero

                drivercont=vexRT(Ch6);      //driver control switch  is checked (right switch)

                                            //either right switches will activate driver control

        }                                   //end of while loop

while(time100[T2]<1000)                     //while the timer is less than 100 seconds (1 minute 40 seconds)

                                            //time100=time in 100msec intervals

        {                                   //start of while bracket

                motor[leftwheel]=vexRT(Ch3);//set the left motor to the left joystick vertical channel

                motor[rightwheel]=vexRT(Ch2);//set the right motor to the right joystick vertical channel

        }                                   //end of while statement

while(true)                                 //while true - programme will remain in this loop

        {                                   //start of while statement

                motor[leftwheel]=0;         //sets the left motor to 0

                motor[rightwheel]=0;        //sets the right motor to 0 - robot stops - end of driver control

        }                                   //end of while statement

}                                           //end of programme
```

Use this programme template to simulate competition control in a match.

You can place your autonomous code in the highlighted area and it will run for 20 seconds when you press the push switch.

When either right button on the controller is pressed the robot will operate for 1 minute and 40 seconds simulating driver control after which it will stop. The robot will need to be turned off and back on to play further games.

Using quadrature shaft encoders

Using servomotors

```
#pragma config(Motor,  port8,  servo,  tmotorServoStandard, openLoop)

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


task main()

{

        while (true)

        {

                motor[servo] = 127;             //sets the servo to full deflection right

                wait1Msec(2000);                //wait 2 second

                motor[servo] = 0;               //sets the servo to centre

                wait1Msec(2000);                //wait 2 second

                motor[servo] = -127;            //sets the servo to full deflection left

                wait1Msec(2000);                //wait 2 second

        }
```

Servo motors only rotate through a fixed angle.

They are good for controlling things and are not used for heavy loads.

Servos use a value from -127 through to +127 to position the servo either side of centre.

Botgear Challenge Programme

```
#pragma config(Motor,  port1,  leftwheel,  tmotorVex393HighSpeed, openLoop)

#pragma config(Motor,  port2,  rightwheel, tmotorVex393HighSpeed, openLoop, reversed)

#pragma config(Motor,  port8,  servo,   tmotorServoStandard, openLoop)

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

//botgear challenge programme

task main()

{
                                         //programme start bracket

bIfiAutonomousMode = false;              //no autonomous mode

while(true)                              //programme continually loops back to while statement

{                                        //start of loop

motor[leftwheel]=vexRT(Ch3);             //set the left motor to the left joystick vertical channel

motor[rightwheel]=vexRT(Ch2);            //set the right motor to the right joystick vertical channel

motor[servo]=-(vexRT(Ch3)-vexRT(Ch2))/2; //set servo to difference between motors

}                                        //end of loop

}                                        //end of programme
```

Tile Trial Programme Template

```
#pragma config(Sensor, in2,    bumpswitch,     sensorTouch)

#pragma config(Motor,  port1,   leftwheel,  tmotorVex393HighSpeed, openLoop)

#pragma config(Motor,  port2,  rightwheel,  tmotorVex393HighSpeed, openLoop, reversed)

#pragma config(Motor,  port8,    servo,    tmotorServoStandard, openLoop)

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

//botgear competition control simulation

task main()

{                                   //programme start bracket

bIfiAutonomousMode = false;         //no automatic autonomous mode

int auton=0;                        //creates and stores an integer called auton which equals zero

                                    //records whether the autonomous bumpswitch has been pressed

                                    //when pressed the robot excecutes 20 seconds of autonomous code

int drivercont=0;                   //creates and stores an integer called drivercont which equals zero

                                    //used to start the 1 minute 40 second driver control period

                                    //the robot will stop at the end of this period

while(auton<1)                      //while autonomous button is not pressed

        {                           //start of while statement

        ClearTimer(T1);             //timer T1 is set to zero

        auton=(SensorValue(bumpswitch));//autonomous bumpswitch is checked

        }                           //end of while statement

while(time100[T1]<200)              //while timer is less than 20 seconds(time100=time in 100msec intervals)

        {                           //start of while statement - place your autonomous code in here

                motor[leftwheel]=60;//sets the left motor to 60

                motor[rightwheel]=20;//sets the right motor to 20 - robot turns to the right for 20 seconds

        }                                   //end of while statement

while(drivercont==0)                        //while right button (Ch 6)  on controller is not pressed

        {                                   //start of while loop

                motor[leftwheel]=0;         //sets the left motor to 0
                motor[rightwheel]=0;       //sets the right motor to 0 - robot stopped

                ClearTimer(T2);            //timer T2 is set to zero

                drivercont=vexRT(Ch6);     //driver control switch  is checked (right switch)

                                           //either right switches will activate driver control

        }                                  //end of while loop

while(time100[T2]<1000)                    //while the timer is less than 100 seconds (1 minute 40 seconds)

                                           //time100=time in 100msec intervals

        {                                  //start of while bracket

                motor[leftwheel]=vexRT(Ch3);//set the left motor to the left joystick vertical channel

                motor[rightwheel]=vexRT(Ch2);//set the right motor to the right joystick vertical channel

                motor[servo]=-(vexRT(Ch3)-vexRT(Ch2))/2;  //set servo to difference between motors

        }                                  //end of while statement

while(true)                                //while true - programme will remain in this loop

        {                                  //start of while statement

                motor[leftwheel]=0;        //sets the left motor to 0

                motor[rightwheel]=0;       //sets the right motor to 0 - robot stops - end of driver control

        }                                  //end of while statement

}                                          //end of programme
```

Use this programme template to simulate competition control in a match.

You can place your autonomous code in the highlighted area and it will run for 20 seconds when you press the push switch.

When either right button on the controller is pressed the robot will operate for 1 minute and 40 seconds simulating driver control after which it will stop. The robot will need to be turned off and back on to play further games.