ROBOTC Programming Competition Templates

This document is part of a software inspection guide for VEX v0.5 (75 MHz crystal) and VEX v1.5 (VEXnet Upgrade) microcontroller-based robots. Use this document to learn how to use or troubleshoot the "Competition Template" or "Driver Skills Template" included with ROBOTC.

You will need:

- A computer with ROBOTC for IFI VEX 2.0.2 or later installed (available at www.robotc.net)

Normal Programming vs. Competition Programming

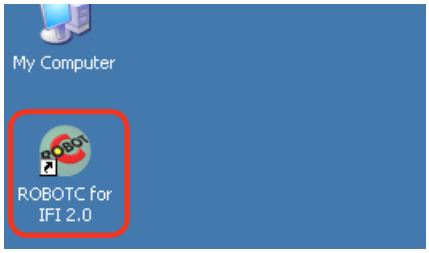
In ROBOTC, every program is usually based around **`task main()`**, as shown below:

```
task main()
{
  //User code goes here
}
```

In a VEX competition, however, the robots need to communicate with the field control, so programming is a little different. To keep things simple, ROBOTC comes with built-in Competition and Driver Skills templates. The templates contains three main sections, each mapped to specific portions of the competition, where teams can place their code.

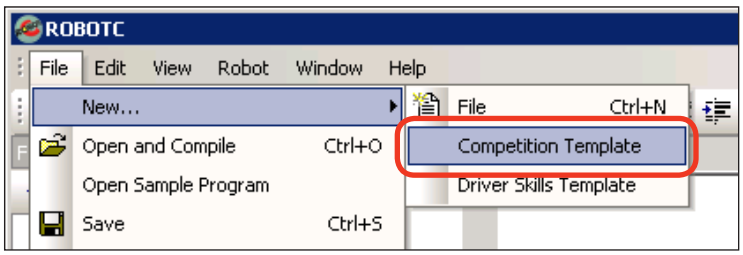
Part I. Using the ROBOTC Competition Template

1. Open *ROBOTC for IFI*.



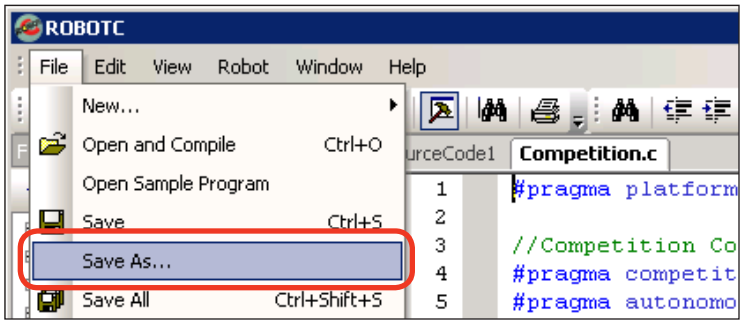
1. ***Open ROBOTC for IFI***
   Open *ROBOTC for IFI* from your Desktop or Start Menu.

2. To begin programming in a Competition Template, go to *File* > *New...* and select *Competition Template*.



ROBOTC Programming Competition Templates *(cont.)*

3. A new file named ***Competition.c*** will appear. Before making any changes to the template, go to ***File***, select ***Save As...*** and save this program in a location and under a name you will remember.



Shown below are the contents of the ***Competition.c*** file with brief descriptions, but with the comments removed and additional spacing added. Additional details are available further down.

```
#pragma platform(VEX)
```

This **pragma** statement specifies the VEX as the platform type.

```
#pragma competitionControl(Competition)
```

This **pragma** statement enables competition control.

```
#pragma autonomousDuration(20)
#pragma userControlDuration(120)
```

These **pragma** statements specify the duration of the autonomous and user control portions of the competition (20 and 120 second defaults) for robots using the 75 MHz crystals. If your robot is using the 75 MHz crystals, do NOT remove these lines of code. If your robot is using the VEXnet Upgrade, these lines of code can be commented-out or deleted.

```
#include "Vex_Competition_Includes.c"
```

This **include** file contains the functionality necessary for the robot to operate with the VEX field control. Do NOT modify the include file or remove this line of code.

```
void pre_autonomous()
{
  //Place pre-autonomous code here
}
```

Any initialization code, such as setting servo positions or clearing encoder values can be placed within the **pre_autonomous()** function.

```
task autonomous()
{
  //Place autonomous code here
  AutonomousCodePlaceholderForTesting();
}
```

All code for the autonomous portion of the competition should be placed within the **autonomous task**. The **AutonomousCodePlaceholderForTesting** function can be removed once you place your own code within the task.

```
task usercontrol()
{
  while(true)
  {
    //Place user control code here
    UserControlCodePlaceholderForTesting();
  }
}
```

All code for the user control portion of the competition should be placed within the **while()** loop of the **usercontrol task**. The while() loop repeats all commands within its curly brackets for the duration of the user control portion of the competition, ensuring that the transmitter data and any other values are up-to-date. The **UserControlCodePlaceholderForTesting** function can be removed once you place your own code within the task.

ROBOTC Programming Competition Templates *(cont.)*

Setting up Competition Control and Timing

In ROBOTC, competition programming is completely user customizable, so no jumpers are required for testing. By adjusting the following commands, the templates can be adapted to work in any VEX supported competition.

```
#pragma competitionControl(Competition)
#pragma autonomousDuration(20)
#pragma userControlDuration(120)
```

`#pragma competitionControl(competition_mode)` - Controls the competition mode that the VEX is in. There are two different competition modes that you can pass:

`OFF` - No competition control.

`Competition` - The VEX will respond to field control commands and switch between Autonomous and User Control modes at the competition-specified times. Use this mode for competitions.

`#pragma autonomousDuration(time_in_seconds)` -
Defines the duration of the autonomous phase of a VEX competition for robots using the 75 MHz crystals. To use the competition template in a User Control-only competition, set the duration to zero.

`#pragma userControlDuration(time_in_seconds)` -
Defines the duration of the user control phase of a VEX competition for robots using the 75 MHz crystals. To use the competition template in an Autonomous-only competition, set the duration to zero.

Important Information - Timing Tips

For robots using the 75MHz crystals, the user control duration can be increased beyond the actual length of the round to compensate for any possible delays in the system. For example, changing **`userControlDuration(120)`** to **`userControlDuration(180)`** would ensure that the robot remains active until the Field Control system ends the user control period.

The durations of robots using the VEXnet Upgrade are determined solely by the Field Control system. When programming robots using the VEXnet Upgrade, you can comment-out or delete the **`autonomousDuration()`** and **`userControlDuration()`** pragma statements.

Pre-Autonomous Period

Place your initialization code inside this function. During the pre-autonomous period, code can be executed to configure your robot before the actual competition begins. Valid code for this section includes tasks such as clearing encoders, reflecting motors, and setting initial servo positions.

```
void pre_autonomous()
{
  //Place pre-autonomous code here
}
```

Note: This code executes only once and runs before the competition begins.

ROBOTC Programming Competition Templates *(cont.)*

Autonomous Period

Place your autonomous code inside this task. During the autonomous period, the robot performs the pre-programmed actions for the length of time specified in the **`autonomousDuration(time)`** pragma statement, or until it is disabled. The **`AutonomousCodePlaceholderForTesting();`** function is only a placeholder and **should be replaced** with your own code.

```
task autonomous()
{
  //Place autonomous code here
  AutonomousCodePlaceholderForTesting();
}
```

Important Information - Transmitter Signal

The robot cannot accept commands from the Radio Control Transmitter during the autonomous period, but it requires that its signal be present as a safety precaution. The autonomous period during a competition cannot be skipped by shutting the transmitter off; doing so with a robot using the 75 MHz crystals will pause the VEX's internal timers, potentially causing the robot to enter the User Control period later than it should.

Also note that the **`AutonomousCodePlaceholderForTesting();`** function contains a **`while(true)`** loop that can prevent user-written autonomous code from executing. Remove this function to avoid any unexpected behavior during the autonomous period.

User Control Period

Place your user control code inside this task. During the user controlled period, the robot accepts commands from the Radio Control Transmitter. This segment of code typically executes immediately after the autonomous period ends. The **`UserControlCodePlaceholderForTesting();`** function is only a placeholder and should be removed once you place your own code inside of the **`while(true)`** loop.

```
task usercontrol()
{
  while(true)
  {
     //Place user control code here
     UserControlCodePlaceholderForTesting();
  }
}
```

Important Information - while(true) loop

When programming for the user control period, place all commands inside of the **`while(true)`** loop. Failing to do so will result in the commands only running once, preventing you from remotely controlling your robot.

Also note that shutting the Radio Control Transmitter off during the user control period does not increase the duration; the field control system determines and controls the maximum length of the user control period.

ROBOTC Programming Competition Templates *(cont.)*

Sample Competition Template with User Code

```
#pragma config(Sensor, in1, bumper, sensorTouch)
//*!!Code automatically generated

#pragma platform(VEX)

#pragma competitionControl(Competition)

#pragma autonomousDuration(20)
#pragma userControlDuration(120)

#include "Vex_Competition_Includes.c"

void forwardUntilTouch()
{
 while(SensorValue[bumper] == 0)
 {
    motor[port2] = 63;
    motor[port3] = 63;
 }
 motor[port2] = 0;
 motor[port3] = 0;
}

void pre_autonomous()
{
 bMotorReflected[port2] = true;
}

task autonomous()
{
 forwardUntilTouch();
 motor[port6] = 31;
 wait1Msec(250);
 motor[port6] = 0;
}

task usercontrol()
{
 while(true)
 {
    motor[port2] = vexRT[Ch2];
    motor[port3] = vexRT[Ch3];
    motor[port6] = vexRT[Ch6]/4;
 }
}
```

Motors and Sensors configured using the ROBOTC *Motors and Sensors* setup window will automatically be added as **`pragma`** statements at the top of the program.

While it's not technically required, it's good programming practice to place user-defined functions and/or global variables below the pragma and include statements, and above the template-included functions and tasks.

This sets the motor connected to VEX Port 2 to spin in reverse.

Commands in the autonomous period are executed once (assuming there are no loops in the code). Also, notice that the **`AutonomousCodePlaceholderForTesting`** function has been removed.

Commands in the while(true) loop of the user control period are executed until the end of the match. Also, notice that the **`UserControlCodePlaceholderForTesting`** function has been removed from the loop.

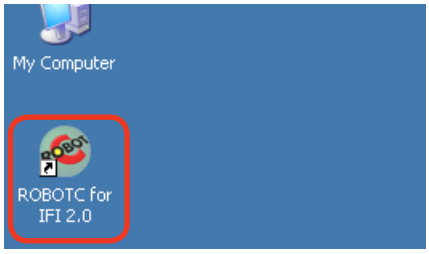
ROBOTC Programming Competition Templates *(cont.)*

Part II. Using the ROBOTC Driver Skills Template

Using the ROBOTC Driver Skills Template is very similar to using the ROBOTC Competition Template. The two templates are nearly identical; the only difference between the two are comments in the code, the autonomous duration is set to zero seconds by default, the user control duration is set to sixty seconds by default, and basic radio control commands are already placed within the while(true) loop of the user control section of the program.

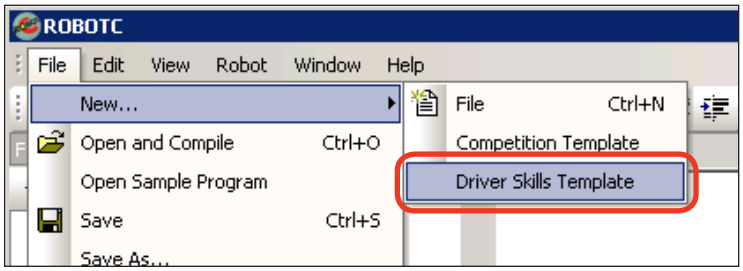
---

1. Open *ROBOTC for IFI*.

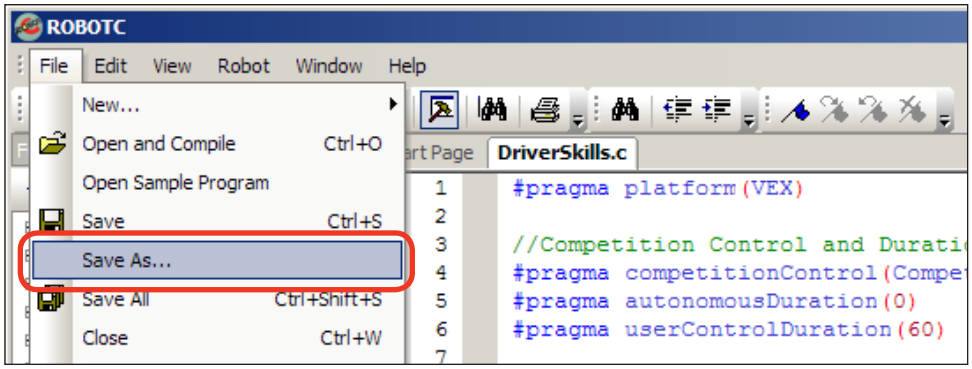


1. ***Open ROBOTC for IFI***
   Open *ROBOTC for IFI* from your Desktop or Start Menu.

2. To begin programming in a Driver Skills Template, go to *File* > *New...* and select *Driver Skills Template*.



3. A new file named *DriverSkills.c* will appear. Before making any changes to the template, go to *File*, select *Save As...* and save this program in a location and under a name you will remember.



ROBOTC Programming Competition Templates *(cont.)*

Shown below are the contents of the ***DriverSkills.c*** file with brief descriptions, but with the comments removed and additional spacing added. Additional details can be found in **Part I** of this document.

```
#pragma platform(VEX)

#pragma competitionControl(Competition)

#pragma autonomousDuration(0)
#pragma userControlDuration(60)




#include "Vex_Competition_Includes.c"

void pre_autonomous()
{
  //Place pre-autonomous code here
}

task autonomous()
{
  //Leave this section alone
  AutonomousCodePlaceholderForTesting();
}

task usercontrol()
{
  while(true)
  {
     //Place user control code here
     UserControlCodePlaceholderForTesting();
     motor[port2] = vexRT[Ch1];
     motor[port3] = vexRT[Ch4];
  }
}
```

These **`pragma`** statements specify the duration of the autonomous and user control portions of the competition (0 and 60 second defaults) for robots using the 75 MHz crystals.

If your robot is using the 75 MHz crystals, do NOT remove these lines of code. If your robot is using the VEXnet Upgrade, these lines of code can be commented-out or deleted.

Any initialization code, such as setting servo positions or clearing encoder values can be placed within the **`pre_autonomous()`** function.

The **`autonomous task`** can be ignored when programming for a Driver Skills competition, but do NOT delete it.

All code for the user control portion of the competition should be placed within the **`while()`** loop of the **`usercontrol task`**. The while() loop repeats all commands within its curly brackets for the duration of the user control portion of the competition, ensuring that the transmitter data and any other values are up-to-date.

The pre-existing code within the loop can be replaced with your own code.