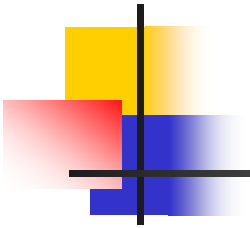
Oregon Robotics Tournament and Outreach Program

Getting Started with FTC Using RobotC

2008

Opening doors to the worlds of science and technology for Oregon's youth



Instructors

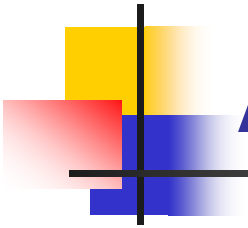
Coordinator	Ed C. Epp
Robot C for Tetrix	Dale Jordan
Advanced RobotC	Jeff McBride
Tetrix Hardware	John DeLacy
NXT-G for Tetrix	David Perry
Instructor	Leroy Knuckles

Getting Assistance

- For assistance, send an email to questions@ortop.org and request to be subscribed to the FTC community list serv
- You will then be placed on an unmoderated listserv at ortopftc@lists.ous.edu where you can ask questions

Today's Goal

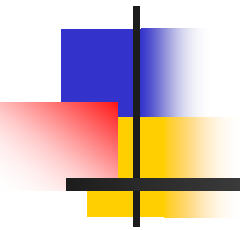
- Understand the Basics of getting started with the FTC kit using RobotC.
- Have enough reference information to help teams get started with the detailed work.



Agenda

- Introductions
- Resources
- Tetrix Platform
- Setting Robot up for RobotC
- Writing a Simple C Program
- Debugging with RobotC
- Bluetooth Communications

Introductions



Resources

- FIRST FTC Website
  http://www.usfirst.org/community/FTC/
- Oregon Robotics and Outreach Program
  http://www.ortop.org
- RobotC
  http://www.robotc.net
  http://www.robotc.net/content/lego_curric/pdfs/nxt_reference_guide.pdf
- FTC Training at CMU
  http://www.ftctraining.com

Tetrix Kit Components

- Tetrix Hardware
- Lego Mindstorms NXT Education Kit
- Software (RobotC, NXT-G and LabView)

Tetrix Hardware

- The Robot's maximum dimensions at start of challenge:
  - 18" W x 18" L x 18" H
- Tetrix kit (at registration): $900
  $450 for returning teams

Developed by Pitsco and LEGO
Over 500 parts per kit
Subset of the parts
pictured here

NXT Mindstorms Kit

- NXT Intelligent Brick
- Rechargeable lithium battery and charger
- 3 servo motors with built in rotation sensors
- 1 each: light sensor, sound sensor, ultrasonic sensor
- 2 touch sensors
- 3 converter cables
- 7 connector cables
- USB cable
- 100s of building elements

Software Packages

- RobotC (Carnegie Mellon University)
- NXT-G (Used by FLL teams)
- LabView (National Instruments)

- All languages have added support for Tetrix control and FTC.

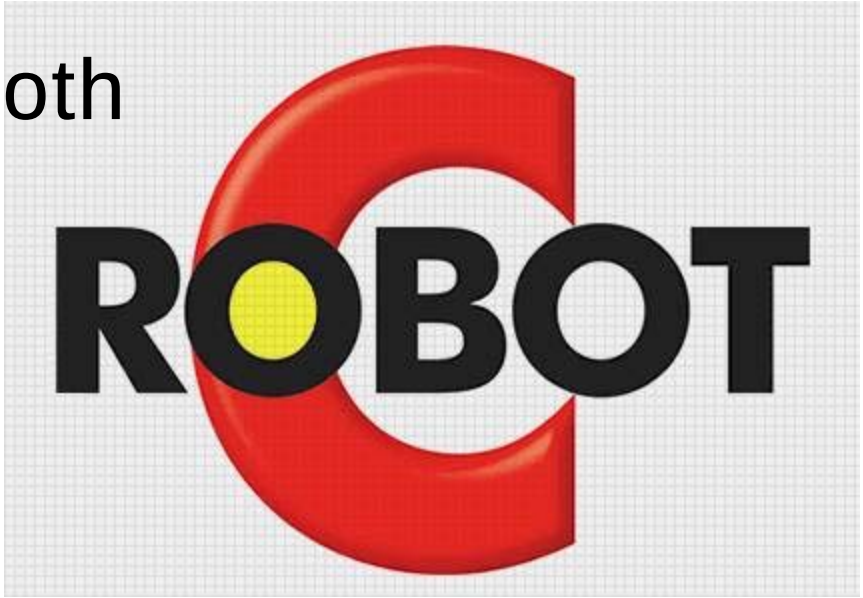
- This class focuses on using RobotC

Demo Robot

- NXT controller
- Differential Drive
- Sensors
  - Touch
  - Light
  - Ultra-sonic
  - Compass

Setting up to Use RobotC

- Installing RobotC on PC
- Configuring RobotC
- Downloading Firmware
- Setting up Bluetooth
- Remote Control



Installing RobotC

- 30 day free trial or online purchase from: www.robotc.net
- On CD as part of the FTC team kit
- System Requirements:
  - Windows XP service Pack 2
  - Windows Vista

- For problems in installation consult:
  http://www.robotc.net/content/lego_support/lego_support.html

Setting Preferences

- **Robot → Platform Type → FIRST Tech Challenge**
- **Window → Menu Level → expert**
- **View → Preferences**
  - Intrinsic Help: check "*Show intrinsic help in separate window*"
  - Compiler: check "*Start Debugger after download*"
- Click "*OK*"
- Exit RobotC and restart it to have all changes take affect (Platform Type)

Downloading Firmware

- Full details at:
  - www.robotc.net → quickstart → “download firmware.pdf”
- In a nutshell:
  - Connect NXT to PC with USB cable (not Bluetooth)
  - Start RobotC and turn on NXT
  - Go to menu **Robot → Download Firmware**
  - Select your **NXT** then **F/W Download**
  - Select the desired **.rfw** file followed by “**open**”
  - Wait for the firmware to download
  - Select **Rename NXT** (useful for Bluetooth)

RobotC Main Sections

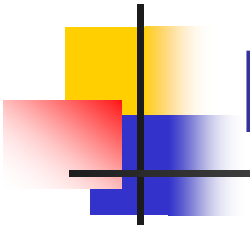
- Menu and Tool Bar
- Code Editor
- Code Template
  - Used to bring in predefined code snippets
  - Helpful in getting the names correct
  - Lets you see what's available in the system

Debugger Window

- **Robot → Debugger**
  - Brings up basic debug window and enables the debug selection.
  - Robot needs to be turned on and connected to PC.
- **Robot → Debug Windows →** (desired windows)
  - Select NXT Remote Screen

    Brings up a remote screen that emulates NXT display

Basic C Syntax

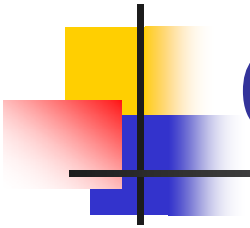


task main() {...}	Tasks and functions
int x;	Variable declaration
while (condition) {...}	Looping
if (condition) {...} else {...}	Conditional
x = expression;	Assignment
+ - * / %	Arithmetic operations
& | ~ ^	Bitwise logical
&& || == < <= > >=	Comparison
//... or /* ... */	Comment

Creating 1st Program

- Create the following program:

```
task main()
{
  eraseDisplay(); // clear the NXT display
  // display "Hello World"
  nxtDisplayCenteredBigTextLine(2, "Hello");
  nxtDisplayCenteredBigTextLine(4, "World!");
  while (true)    // loop until program is terminated so
    ;             //    that we can read the display
}
```



Creating 1st Program (cont)

Methods to create programs:

- Type in the program
- Cut and paste elements from an existing program
- Use the code template

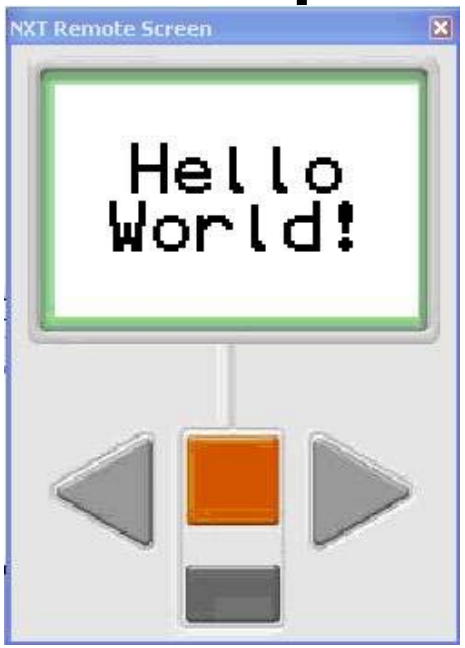
Creating 1st Program (cont)

Using Code Template (drag items from Template to Code Window then edit as appropriate)

- _C Constructs → Tasks/Subroutines → task taskname ...
  - Change **taskname** to **main**
  - Select and delete **body**
- Display → Intrinsics → eraseDisplay()
- Display → Intrinsics → nxtDisplayCenteredBigTextLine ()
- Display → Intrinsics → nxtDisplayCenteredBigTextLine ()
- Edit arguments for last two commands
  - Change **nLineNumber** to **2** and **4** respectively
  - Change **sString** to **"Hello"** and **"World!"** respectively
- _C Constructs → Control Structure → while
  - Change **conditional** to **true**
  - Change **{ body }** to **;**
- Add comments to describe what is happening (optional)
- File →Save As→Hello World.c (FTC Class directory on desktop)

Running 1st Program

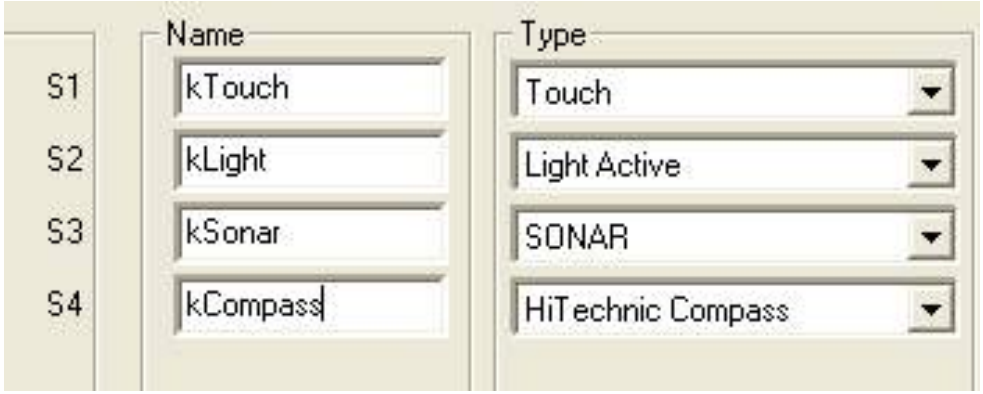
- **Robot → Compile and Download Program**
- Note: Pops up debugger
- Insure remote display is up
- While observing the remote display, Select **step into, step over, …, stop**
- Close Debug window to end session



Setting up Sensors & Motors

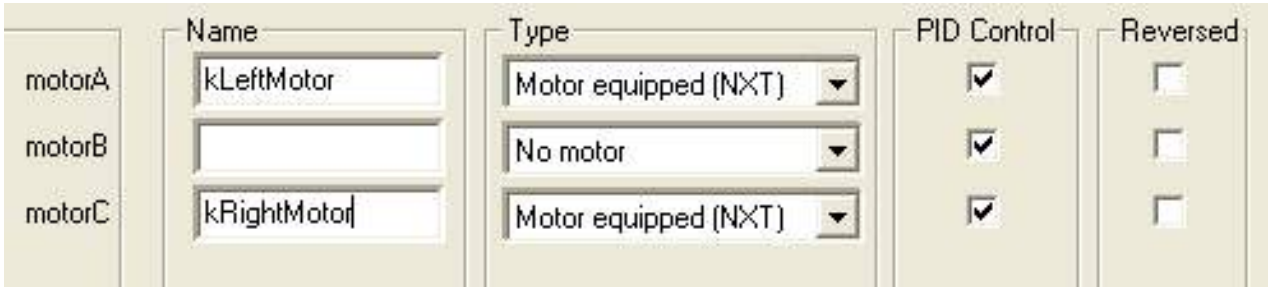
- **File → New**
- **Robot → Motors and Sensors Setup**
- Select “Device Mgmt NXT”

  Select “Allow 3rd Party Sensors from HiTechnic”
- Select “Sensors” (fill in and select the following)



Setting Up Sensors & Motors (cont)

- Select “Motors"
- Fill in the table to match the following:



- Select “OK”
- Look through selections for FTC specific items

Setting Up Sensors & Motors (cont)

- Note the generated code
- Compiler uses *#pragma's* to setup code that configures the Sensors before *task main()* is entered (prolog)
- Alternatively, *#define and SensorType[]* and *SensorSubType[]* can accomplish the same things
- With the complexity of FTC connections, it is recommended to let RobotC generate the sensor and motor configurations

```
#pragma config(Sensor, S1,     kTouch,                sensorTouch)
#pragma config(Sensor, S2,     kLight,                sensorLightActive)
#pragma config(Sensor, S3,     kSonar,                sensorSONAR)
#pragma config(Sensor, S4,     kCompass,              sensorI2CHiTechnicCompass)
#pragma config(Motor,  motorA,          kLeftMotor,    tmotorNormal, PIDControl, )
#pragma config(Motor,  motorC,          kRightMotor,   tmotorNormal, PIDControl, )
```

Reading Sensors

- To the sensor definition code add:

```
task main()
{
  eraseDisplay(); // clear the NXT display
  nxtDisplayTextLine(0, "Sensor     Value");
  nxtDisplayTextLine(1, "----------------");
  while (true)    // loop until the program is terminated by user
  {
      nxtDisplayString(2, "touch       %d", SensorValue[kTouch]);
      nxtDisplayString(3, "light       %d", SensorValue[kLight]);
      nxtDisplayString(4, "sonar       %d", SensorValue[kSonar]);
      nxtDisplayString(5, "compass     %d", SensorValue[kCompass]);
      nxtDisplayString(6, "left rot    %ld", nMotorEncoder[kLeftMotor]);
      nxtDisplayString(7, "right rot   %ld", nMotorEncoder[kRightMotor]);
      wait1Msec(3);
  }
}
```

- **File → Save As → DisplaySensors.c**
- **Robot → Compile and download**
- **Robot → Debugger**
- **Robot → Debug Windows → NXT Devices**
- **Click "*Start*"**
- View the motors and sensors in the Device Window

Using Motors & Sensors

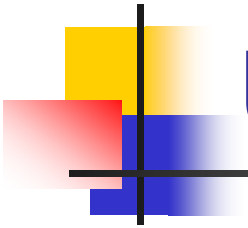
- Generate simple program to move robot forward for 2 seconds

```
task main()
{
  motor[kLeftMotor] = 100;    // Turn motors on
  motor[kRightMotor] = 100;

  wait1Msec(2000);            // Wait 2 seconds

  motor[kLeftMotor] = 0;      // Turn motors off
  motor[kRightMotor] = 0;

  // wait a little so we can read debug window
  wait1Msec(2000);
}
```



Using Motors & Sensors (cont)

To use sensors to determine when the robot stops, replace wait1Msec line with one of:

- Rotation sensor

```
while (nMotorEncoder[kLeftMotor] < kCount) ;
```

- Touch sensor

```
while (SensorValue[kTouch] == 0) ;
```

- Light sensor

```
while (SensorValue[kLight] > kThreshold) ;
```

- Ultra-sonic sensor

```
while (SensorValue[kSonar] > kDist) ;
```

Setting up Bluetooth

First time startup:

- Insure Bluetooth Dongle is installed and running on PC
  - Wait for Windows to indicate device is ready to use
- On NXT
  - Select Bluetooth Menu
  - Set visibility to visible
  - Use default Password
  - On/off to on

Setting up Bluetooth (cont)

First time startup (cont)

- Start RobotC
- **RobotC→NXT Brick→Link Setup**
- Check "*include Bluetooth in Search*"
- Click "Refresh Lists"
- Select Bluetooth device that is your NXT from the *"NXT Bricks Reachable via Bluetooth"* window
- Click "*Select"*
- When asked, Press orange button on NXT to send password

Setting up Bluetooth (cont)

Subsequent startup:

- **RobotC→NXT Brick→Link Setup**
- Select the device that is your NXT from the "*NXT Brick Connection History*"
- Click "*Select*"
- If the connection fails:
  - Make sure NXT Bluetooth is on
  - Exit and reenter RobotC
  - Go through the first time startup process again
- When RobotC needs to reconnect to the NXT it will try to do this automatically when the debug window is brought up.

Joystick Control

- Connect the Logitech Game Controller to the PC through a USB Port
- Wait for Windows to indicate device is ready to use
- **Robot → Debug** (NXT must be connected through BT)
- **Robot → Debug Windows → Joystick Control**

Joystick Control (cont)

- Load the program: Joystick.c

```
#pragma config(Motor,  motorA,          kLeftMotor,    tmotorNormal, PIDControl, )
#pragma config(Motor,  motorC,          kRightMotor,   tmotorNormal, PIDControl, )
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//
#include "JoystickDriver.c"
TJoystick joystick;

task main()
{
    int jLeft, jRight;
    while (true)
    {
        getJoystickSettings(joystick);    // read the current joystick settings
        jLeft = joystick.joy1_y1;
        jRight = joystick.joy1_y2;
        if (jLeft > -10 && jLeft < 10)    // core out the noise for near zero settings
            jLeft = 0;
        if (jRight > -10 && jRight < 10)
            jRight = 0;
        motor[kLeftMotor] = jLeft;        // set motors to joystick settings
        motor[kRightMotor] = jRight;
        wait1Msec(5);                     // let other tasks run
    }
}
```

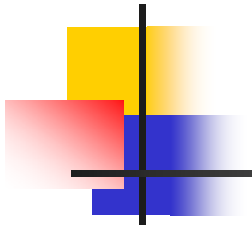
Example: Keep Robot Straight

- Download RobotDir.c
- Place Robot on a lazy-suzan
- Run program
- Swivel lazy-suzan
- Observe Robot correction

Example: Multiple Tasks

- RobotC supports running up to 10 tasks at a time.
- Priority based
- Round robin when same priority
- Support for creating, starting and stopping tasks
- Sample program:
  - Start with the *DisplaySensors.c* program
  - **File → Save As → DisplaySensorsAsTask.c**
  - Change *main* in **task main** to *displaySensors*
  - Add the following new task main:

```
task main()
{
  StartTask(displaySensors);
  while (true)
    ;    // insert code to do useful work here
}
```



Questions
Discussion
Wrap up