

C++ Programming For 2010-2011 FRC Teams



Mike Anderson (manderson13@cox.net) and Greg Smith (gnsmith@cox.net)



Herndon High School FRC Team #116

What We'll Talk About

- Goals
- Why C/C++?
- The development environment
- Talking to the cRIO
- Making it move
- Resources
- Summary



Goals

- The goal of this presentation is to help you understand how to use C/C++ in the development of your robot
- We clearly can't explain all of the aspects because we only have 75 minutes
- But, you should leave here with a better understanding of the process



Why C/C++?

 C/C++ is a standard in embedded systems programming for over 30 years

- It's still the most predominant language in the embedded, real-time operating system (RTOS) world
 - This gives your team valuable real-world experience
- It's compiled to native machine code
 - No virtual machine interpreters
 - No pausing due to garbage collection
 - It's fast
- It's the native language of the VxWorks RTOS
 - The environment is written in C and Assembler
 - You get easy, direct access to the underlying O/S
- C++ is object oriented
 - Full support from WPILib



Why Not C/C++?

- C/C++ is compiled
 - This adds complexity to the build
- C/C++ is textual
 - ► There are no cutesy GUIs with lots of obscure symbols and squiggly lines ☺
- There is no hand-holding VM to catch your mistakes
 - The syntax is similar to Java, but it's definitely not Java
- C/C++ has pointers
 - Objects can be referenced in many different ways
 - This concept can be too much for some developers

Getting Your cRIO Ready

 Before you can start development, you'll need to make sure that your cRIO has the proper operating system image on it

This is accomplished using the cRIO imaging tool

CompactRIO Imaging Tool Select CompactRIO Device MAC address Name Current IP 00802F118A0C FRC-cRIO-116 10.1.16.2 Development Environment Always run deployed code at startup Wind River Workbench (C++) Java Technology Format Controller Select Image FRC_2010_v20.zip Device name FRC-cRIO-116 Team ID 116 Version Number Apply Close 2009.12.11.00



The Development Environment

- The FIRST provided platform is the Wind River Systems Workbench tool
 - IDE is based on the open-source Eclipse tool
 - The compiler is the open-source GNU compiler
 - But, the front-end is licensed and requires a key

 Provided by FIRST
- The compiler is actually a *cross-compiler*
 - We are building on an x86 for a PowerPC
 - Again, this is a standard approach for commercial, embedded development



Development Environment #2

- Workbench runs under Windows and Linux
 FIRST only supplies the Windows installation
- Workbench runs under Windows XP, Vista or Windows 7 (2011 season)
 - You can also run it in a virtual machine
 - For all you Mac OS/X and Linux fans
 - When you install it, there will be a key file that needs to be placed in c:\windriver\license



The WRS Workbench



FRC C++-Introduction-9



Creating A Project

- Workbench collects all of the files related to building a piece of code into a subdirectory called a project folder
 - It's normally stored under the c:\windriver\workspace directory
 - You can put the project elsewhere when you open the workbench tool
- You can also import and export projects
 - This allows you to create a .zip of the project for archival purposes

New Project -- Simple Robot

New Project Image: Comparison of Compariso		
Wizards: type filter text	New Project Sample	
VXWorks 6.x WXWorks Boot Loader / BSP Project VXWorks Downloadable Kernel Module Project VXWorks Image Project VXWorks Real Time Process Project VXWorks Shared Library Project VXWorks Shared Library Project VXWorks Complex Native Sample Project VXWorks Real Time Process Sample Project Show All Wizards.	Sample Project Template Select a sample project template. Available Examples:	Information Display the simplest sample program that implements the full field control and shows the use of the watchdog timer. This is an excellent starting point for your programs. Display the program simply drives forward for 2 seconds in the duo nomous period and does simple arcade driving during the Operator Control period.
	0	

$E\Delta$ - FRC Team #116

New Project Result

```
🖸 MyRobot.cpp 🛛 🔪 📄 sample.txt
   */
 class RobotDemo : public SimpleRobot
  {
      RobotDrive myRobot; // robot drive system
      Joystick stick; // only joystick
 public:
      RobotDemo(void):
         myRobot(1, 2), // these must be initialized in the same order
                        // as they are declared above.
         stick(1)
     {
         GetWatchdog().SetExpiration(0.1);
      }
      / * *
      * Drive left & right motors for 2 seconds then stop
      */
      void Autonomous (void)
      {
         GetWatchdog().SetEnabled(false);
         myRobot.Drive(0.5, 0.0); // drive forwards half speed
         Wait(2.0); // for 2 seconds
         myRobot.Drive(0.0, 0.0); // stop robot
      - }
      / * *
      * Runs the motors with arcade steering.
      */
      void OperatorControl(void)
      -{
         GetWatchdog().SetEnabled(true);
         while (IsOperatorControl())
         -{
             GetWatchdog().Feed();
             myRobot.ArcadeDrive(stick); // drive with arcade style (use right stick)
             Wait(0.005);
                              // wait for a motor update time
         }
      з
 );
```



Build the Project



FRC C++-Introduction-13

ЕД- FRC leam #116

Attach to the Target

- Before you can load code to the robot, you need to create a target connection
 - This will also create a "target server" and "registry" instance
- It is possible to have multiple users attached to the same target
 - Not recommended if you're both running robot code though ;-)





Attach to the Target #2

New Connection	New Connection
Select Remote System Type	Target Server Options Review and customize the target server options.
System type: type filter text General General General Constraints General Constra	Backend settings Backend: wdbrpc Processor: (default from target) Select Target name / IP address: 10.1.16.2 Check Port:
⑦ < Back Next > Finish Cancel Image: Concel Image: Concel Image: Concel Image: Concel	 File path from target (if available) File: Browse Bypass checksum comparison Advanced target server options Verbose target server output Options: -R C:/windriver/workspace -RW -Bt 3 -A Command Line:
✓ ▲ NN N S Connect ↓ Connect ↓ Coal ↓ Local Files ↓ Local Shells ↓ Wind Biver Registries	tgtsvr -V -R C;/windriver/workspace -RW -Bt 3 -A 10.1.16.2 Image: Comparison of the system of the
FRC C++-Introduction-15	ΕΔ- FRC Tear

Attached Target

- Once the robot is attached to the target server, you'll see all of the tasks running on the target in the "Remote Systems" window
- If you create your own threads, they'll show up in the list as well



Running Code

- In order to run your robot code, you'll need to have the target attached
 - You'll need to create a "run config"

	🕲 Run	X
Download	Create, manage, and run configurations Please connect target to change settings.	
Task	Image: Second	Name: FRC_UserProgram_StartupLibraryInit - SimpleTemplate.out - VxWorks6x_10.1.16.2
	Kernel Task Kernel Task Serrer StartupLibraryInit - SimpleTempl Serrer StartupLibraryInitit - SimpleTempl Serrer StartupLibraryInit -	Connection Connection to use: VxWorks6x_10.1.16.2 (localhost) Properties Add Connect VxWorks6x_10.1.16.2 is not connected. Kernel Task to Run
	RTP on Target	Entry Point: FRC_UserProgram_StartupLibraryInit Browse Arguments:
		Stack size: 0x20000 Advanced Options: Edit
		Apply
	Filter matched 7 of 7 items	Run Close

 $E\Delta$ - FRC Team #116



Handling Unloading

Workbench has the option to automatically unload previously loaded code and replace it with the new code

- Not the default behavior though
- There are a series of changes that need to be made to the run configuration to support automatic unloading



Unload Settings

🚳 Run

Create, manage, and run configurations

Image: Second secon	Name: FRC_UserProgram_StartupLibraryInit - SimpleTemplate.out - VxWorks6x_10.1.16.2 Image: Main Image: Downloads Image: Common	
C/C++ Local Application Kernel Task FRC_UserProgram_StartupLibraryInit - SimpleTemplate.out - VxWorksf	Downloads:	
Launch Control Antive Application Process on Target DED on Target	File Symbols Reload Options /SimpleTemplate/PPC603gnu/SimpleTemplate/De Image: Symbols -plugin:nounload	Add
RIP on larget		Remove Up
	Highlight and select "Edit"	Down
Filter matched 7 of 7 items	Apply	levert
0	Run	Close



×

FRC C++-Introduction-19

Unload Settings #2

	CLoad Option
🛿 Download	O load from target server.
	 use auto-loading method (default).
	O load from tool.
	Common Symbols Matching Strategy
	 search both modules and core file for matching symbols (default).
to: MPC5200 (VxWorks6x_10.1.16.2)	 always allocate, no symbol search.
Options	○ exclude core file from symbol search.
✓ Load Symbols to Debug Server	Symbol Loading
Download even when file not modified	 load global symbols only (default).
Advanced Options	🔿 load no symbols.
	🔿 load all symbols.
OK Cancel	○ load local symbols only.
	hide the module from dfwserver (default is to not hide the module).
	no relocation is required (default is to relocate the module).
X	onot unload the existing module (default is unload).
	manually call the C++ constructor (default is to automatically call the C++ constructor).
Change this setting to auto unload	Set Defaults OK Cancel

Advanced Download Options

.



×

Running Code on the Robot

- Once your code is loaded, the entry point (FRC_UserProgram_StartupLibraryInit) Will be called and your code will start running
 - Make sure that your Classmate is attached and you should be able to control the robot
- There is an option to disable the "STOP" button on the UI
 - Use with caution...
- You can then test Autonomous or Teleop code segments

Disabling the Stop Button



Click the LED and you'll get a dialog that allows you to disable the STOP button





Debugging Code

New Select Working Set Deselect Working Set Edit Active Working Set Go Into	•	 In order to debug code, you'll need to create a deb Check the debug option for automatically connect tasks 	oug launcher to spawned
Copy Copy Paste Coper in New Window Paste Copy Copy Copy Copy Copy Copy Copy Copy	Ctrl+C Ctrl+V Delete F2	Create, manage, and run configurations Please connect target to change settings.	Ŕ
Project References Import Import Index Build Project Rebuild Project Clean Project Build Options Download Run Kernel Task Debug Kernel Task	Ctrl+Shift+A	Image: Second Secon	orks6x_10.1.16.2
 Refresh Close Project Open Workbench Development She Run As Debug As Team Compare With Restore from Local History Search 	F5 # } Ctr/+H	Advanced Options:	Apply Revert
Properties	Alt+Enter		Debug Close

FRC C++-Introduction-23



Close

Attaching to Threads

- In order to debug your iterative robot, you need to make sure you automatically attach to spawned tasks
 - Otherwise you won't be able to debug the autonomous/teleop code
- In the debug launcher configuration, set the auto attach feature in the debug configuration:

Nam	e: FRC_UserProgram_StartupLibraryInit - SimpleTemplate.out - VxWorks6x_10.1.16.2
-	Main 📓 Downloads 🚼 Projects to Build 🟇 Debug Options 🏼 🦆 Source 🔲 Common
	Break on Entry
	Automatically attach spawned Kernel Tasks



Debugging Code #2

Once debugging as started, Workbench will automatically switch to the debug perspective:

Device Debug - SimpleTemplate/	MyRobot.cpp - Wind River Workbench					
File Edit Refactor Navigate Search Project Analyze Target FIRST Run Window Help						
$ \stackrel{\circ}{\boxtimes} \stackrel{\circ}{\boxtimes} \cdot \underbrace{\mathbb{R}} \stackrel{\circ}{\boxtimes} \underbrace{\mathbb{R}} \stackrel{\circ}{\boxtimes} \underbrace{\mathbb{R}} \stackrel{\circ}{\boxtimes} \cdot \underbrace{\mathbb{R}} \stackrel{\circ}{\to} \cdot \underbrace{\mathbb{R}} \stackrel{\circ}{\to} \cdot \underbrace{\mathbb{R}} \stackrel{\circ}{\to} \stackrel{\circ}{\to} \underbrace{\mathbb{R}} \stackrel{\bullet}{\to} \underbrace{\mathbb{R}} \stackrel{\bullet}{\to} \underbrace{\mathbb{R}} \stackrel{\bullet}{\to} \underbrace{\mathbb{R}} \bullet$						
🎦 Project 🛛 🙋 Debug S 🖵 🗖	🖸 MyRobot.cpp 🛛 📄 sample.txt 📃 🗖	🏇 Debug 🛛 🖓 🗖				
E SimpleTemplate (Wind River VxWo	*/ class RobotDemo : public SimpleRobot { RobotDrive myRobot; // robot drive system Joustick stick; // conty ioustick	Image: StartupLibraryInit - SimpleTemplate.out				
	<pre>public: RobotDemo(void):</pre>					
	<pre>GetWatchdog().SetExpiration(0.1);) /** * Drive left & right motors for 2 seconds then stop */ void Autonomous(void)</pre>	● Breakpoints ☆ ♥ ♥ ♥ ■ ■ ♥ ♥ SimpleTemplate/MyRobot.cpp:16				
Remote Systems 27	<pre>{ GetWatchdog().SetEnabled(false); myRobot.Drive(0.5, 0.0); // drive forwards half speed Wait(2.0); // for 2 seconds myRobot.Drive(0.0, 0.0); // stop robot V </pre>					
	🔕 Tasks 🔎 Terminal 🔝 Problems 🔲 Properties 🖳 Build Console 🛛 📮 Console 🖓 🖓					
Local Local Files Local Shells Wind River Registries Wind River Registries Wind River Registries Wind River Registries Wind River Registries Wind River	Build Started in Project 'SimpleTemplate': 2010-11-19 19:02:50 Generation of makefiles started. Generation of makefiles finished (Elapsed Time: 00:00). Platform: Wind River YxWorks 6.3 Command: makeno-print-directory BUILD_SPEC=PPC603gnu DEBUG_MODE=1 TRACE=1 Working Directory: C:/windriver/workspace/SimpleTemplate/PPC603gnu make: built targets of C:/windriver/workspace/SimpleTemplate/PPC603gnu Build finished in Project 'SimpleTemplate': 2010-11-19 19:02:51 (Elapsed Time: 00:01)	M= Variabl X 188 Regist 1 2 Expres 1 Memor □ □				



Debugging Code #3

- By default, the code will stop at the public constructor entry-point
- You can right-click in the code view "gutter" to toggle a breakpoint
 - Then tell the system to step or continue
- This is using the GDB debugger under the covers
 - ▶ If you understand GDB, you're in familiar territory
 - If not, we'll show you some examples during the demo

Deploying Code on the Robot

Once you code is working, you can deploy the code to the robot

Only one program can be resident at a time

pleTemplate/MyRobot.cpp - Wind River Workbench									
Project	Analyze	Target	FIRST	Run	Wind	low	Help		
i ⇒i	* •	0 - 9	Dow Unde	nload eploy	-	6	A	 T	Π
23	- D)(MyRob	ot.cpp Σ	X 🖊	📄 sa	ample	.txt		
** 🧭	• ~	*/							

Downloading to robot
Downloading to robot
Run in Background Cancel Details >>

 Once the code is downloaded, you can reboot the robot and your code will run!
 You can also do this with Filezilla (it's just FTP [©])

Some Hints

- The environment for C++ can be a bit tricky to deal with at first
 - Especially if you want to use CAN bus
- For CAN bus, you need to use a Black Jaguar and disable the standard console output on the cRIO (or use the 2CAN Ethernet bridge)
 - For the Black Jaguar, you need exclusive access to the serial port
 - Look on Chief Delphi for write-ups on CAN use



Some Hints #2

- To make the serial console available while still being able to monitor the cRIO console, load the network console application to the target and run the network console application on your host
 - All of the console I/O of the cRIO comes across the network now into a new window
- We'll give you a link to the network console application in the links



The Network Console

Useful as an option to using the RS-232 port on the cRIO

🗵 NetConsole.vi	
Machine Address 10.1.16.2 Auto	ScreenBufferSize 32768
-> * Loading StartupDlls: debug Entering debug.o StartupLibraryInit Debugging is up, target server mounted at /tsfs	
UxWorks	
Copyright 1984-2006 Wind River Systems, Inc.	
CPU: MPC5200 Wind River Lite5200 BSP. Runtime Name: UxWorks Runtime Version: 6.3 BSP version: 2.0/10 Created: Jul 29 2009, 13:41:47 ED&R Policy Mode: Deployed WDB Comm Type: WDB_COMM_END WDB: Ready. Leaving debug.o StartupLibraryInit	
<pre>* Loading StartupDlls: NiRioRpc * Loading StartupDlls: niorbs</pre>	
★ Loading StartupDlls: NiViSrvr	<u>~</u>
EXIT Clear Screen	



Resources

- Chief Delphi

 http://www.chiefdelphi.com

 FIRST forums
 - http://forums.usfirst.org
- NI Community Forums
 http://ni.com/FIRST
- WPI / FIRST NSF Community site (ThinkTank)
- These sites are monitored members of:
 - ► WPI
 - ► NI
 - ► FIRST
- All source code available for team-team assistance
- Phone support through NI
 > 866-511-6285 (1PM-7PM CST, M-F) ?



Important Links

WPILib updates and documentation

- http://first.wpi.edu/FRC/frccupdates.html
- http://first.wpi.edu/Images/CMS/First/WPI_Robotics_Library_Users_Guide.pdf
- http://first.wpi.edu/Images/CMS/First/WPILibSource20100107.zip
- http://first.wpi.edu/Images/CMS/First/CProgrammingReference.chm
 - doxygen old

Net console

http://first.wpi.edu/Images/CMS/First/NetConsoleClient_1.0.0.4.zip

FIRST software resource page

http://www.usfirst.org/roboticsprograms/frc/content.aspx?id=1093



Summary

- C/C++ can be very challenging to new developers
 - C/C++ is similar enough to Java that Java developers can adapt to it quickly
 - However, pointers will require some explaining
- The rewards include:
 - Faster operation of the robot
 - Fine-grain control of the robot's behavior
 - ► The ability to leverage native VxWorks[™] facilities
 - Training in techniques that will enable most of the "green" technology of the future