Resource R2 Embedded computer and development environment subsystem

The development system is a powerful and convenient tool for embedded computing applications. As shown below, the development system consists of a personal computer, connected via a USB cable to target computer.

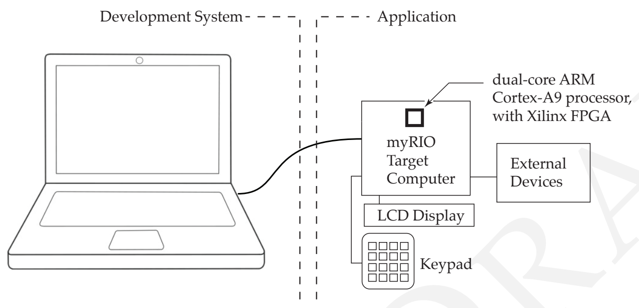

Figure 00.9:

During the development of an embedded computing application, the development system communicates with the real-time Linux operating system of the myRIO target computer.

The development environment includes an integrated set of hardware and software tools that help to debug a microcomputer design by allowing you to watch your program execute, as well as to stop it and inspect system variables. As you will see, it allows you to monitor and control the target computer, without interfering with its timing.

Once hardware and software development is completed, the development system is disconnected from the target system. In the final application, the target program resides in ROM on the target computer.

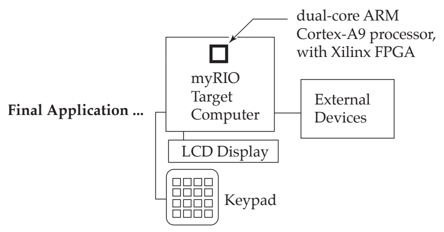

Figure 00.10:

Resource R2.4 Getting Started with CDT

Eclipse is an integrated development environment (IDE). We will use Eclipse through its C Development Tool (CDT) to create, edit, build, deploy, and debug C language projects for the myRIO target computer. Within Eclipse all of your projects are organized into a single workspace on your computer. Each project, along with all of its necessary resources, are stored in a named project folder.

The outline below describes the basic functions of the IDE in preparing a C program for subsequent loading and execution on the myRIO remote system. Additional features are described in the Help menu.

Box 00.1 is CDT set up?

If the development PC has not yet been set up on the development computer, follow the procedure of Resource 7 to do so, before continuing.

Begin by starting the Eclipse IDE application.

Resource R2.5 C/C++ Perspective

Enter the C/C++ Perspective by selecting that button in the upper right.

The Project The ME 477 C Support for myRIO archive that you imported into your Eclipse workspace when you set up the CDT contains a template project for each of the nine laboratory exercises this quarter. They are listed in the right pane of the **C/C++** perspective. Open a project by double clicking on its folder.
Each C program consists of a collection of functions, one of which must be called `main{}`, and is executed first. For large projects, additional functions are often in separate files. However, the organization of the assignments in this class is such that all of the functions for a single assignment can be conveniently stored in `main.c` along with `main{}`.

Run and Debug Configurations Among other things, Run and Debug Configurations specify how the project will be stored on the remote target. Configurations for all ME 477 laboratory exercises were loaded into your workspace in steps 4 and 5 of Part 1 of the C Development Tool Setup documentation—see Resource 7.

Building the Project Building the project consists of compiling your C source code into object modules, and linking them with other

resources. Many coding errors can be found during the building process. Since building does not require that the development system be connected to the target, time spent in the lab is minimized.
Before building, save any edit changes in the source code (`ctrl-s`). Either right click the project and use Build Project, or select and use Build Project from the Project pull down menu. Errors and warnings are displayed in the console menu in the bottom pane.
During each build, the CDT automatically re-compiles any file that has been edited (and saved). The build operation creates an output file in project's Debug folder.

Running the Project The project must build without errors before it can be run. The first time a project is run, pull down the Run menu and select Run Configurations.... In the Run Configurations window, select the Run Configuration of your project. Then click Run.
The first run after a connection, you may be asked to login. Use User ID: `admin` and Password: `me477`.
Recently run projects may be conveniently run from the pull down menu under the run icon.
A project will not run if a project is already running.
Barring execution problems, the project runs until `main{}` terminates.

Resource R2.6 Debug Perspective

Enter the Debug Perspective by selecting that button in the upper right. The Debug perspective lets you manage the debugging or running of a program. You can control the execution of your program by setting breakpoints, suspending launched programs, stepping through your code, and examining the contents of variables.

Debugging the Project The project must build without errors before it can be debugged. The first time a project is debugged, pull down the Run menu and select Debug Configurations.... In the Debug Configurations window, select the configuration of your project. Then click Debug.
After the first debug, the project may be conveniently selected for debugging by pulling down menu under the debug icon.
A project may not be debugged if a project is already running.

Breakpoints A breakpoint suspends the execution of a program at the location where the breakpoint is set. To set a line breakpoint, right-

click in the marker bar area on the left side of an editor beside the line where you want the program to be suspended, then choose Toggle Breakpoint. You can also double-click on the marker bar next to the source code line.
A new breakpoint marker appears on the marker bar, directly to the left of the line where you added the breakpoint. Also, the new breakpoint appears in the Breakpoints view list.
Once set, a breakpoint can be enabled and disabled by right-clicking on its icon or by right-clicking on its description in the Breakpoints view.

- When a breakpoint is enabled, it causes the program to suspend whenever it is hit. Enabled breakpoints are indicated with a blue enabled breakpoint circle.
- Enabled breakpoints that are successfully installed are indicated with a checkmark overlay.
- When a breakpoint is disabled, it will not affect the execution of the program. Disabled breakpoints are indicated with a white disabled breakpoint circle.

Resource R2.7 Debug view toolbar commands

The Debug perspective also drives the C/C++ Editor. As you step through your program, the C/C++ Editor highlights the location of the execution pointer.

Resume Select the Resume command to resume execution of the currently suspended debug target.

Suspend Select the Suspend command to halt execution of the currently selected thread in a debug target.

Terminate Ends the selected debug session and/or process. The impact of this action depends on the type of the item selected in the Debug view.

Step Over Select to execute the current line, including any routines, and proceed to the next statement.

Step Into Select to execute the current line, following execution inside a routine.

Step Return Select to continue execution to the end of the current routine, then follow execution to the routine's caller.

Resource R2.8 Debug information

Variables You can view information about the variables in a selected stack frame in the Variables view. When execution stops, the changed values are by default highlighted in red. Like the other debug-related views, the Variables view does not refresh as you run your executable. A refresh occurs when execution stops.

Expressions An expression is a snippet of code that can be evaluated to produce a result. The context for an expression depends on the particular debug model. Some expressions may need to be evaluated at a specific location in the program so that the variables can be referenced. You can view information about expressions in the Expressions view.

Registers You can view information about the registers in a selected stack frame. Values that have changed are highlighted in the Registers view when the program stops.

Memory You can inspect and change memory.

Disassembly You can view disassembled code mixed with source information.

Resource R2.9 The system on a chip

The NI myRIO is centered around a *Xilinx Z-7010 system on a chip* (SoC): a dual-core *Coretex A-9* CPU, memory, I/O inerfaces, and an *Artix-7 fully programmable gate array* (FPGA). The Z-7010 datasheet Xilinx (2017) is available here.

Xilinx Z-7010 SoC
Coretex A-9
Artix-7 FPGA

These are powerful SoCs. The Coretex A-9 CPUs have 667 MHz clocks, have single- and double-precision vector float point units, and include NEON extensions (Xilinx, 2017). These processors use the *ARMv7-A instruction set architecture* (ISA) (ARM, 2014, 2012). The Coretex-A9 *Reference Manual* and *Programmer's Guide* are available here.

ARMv7-A ISA

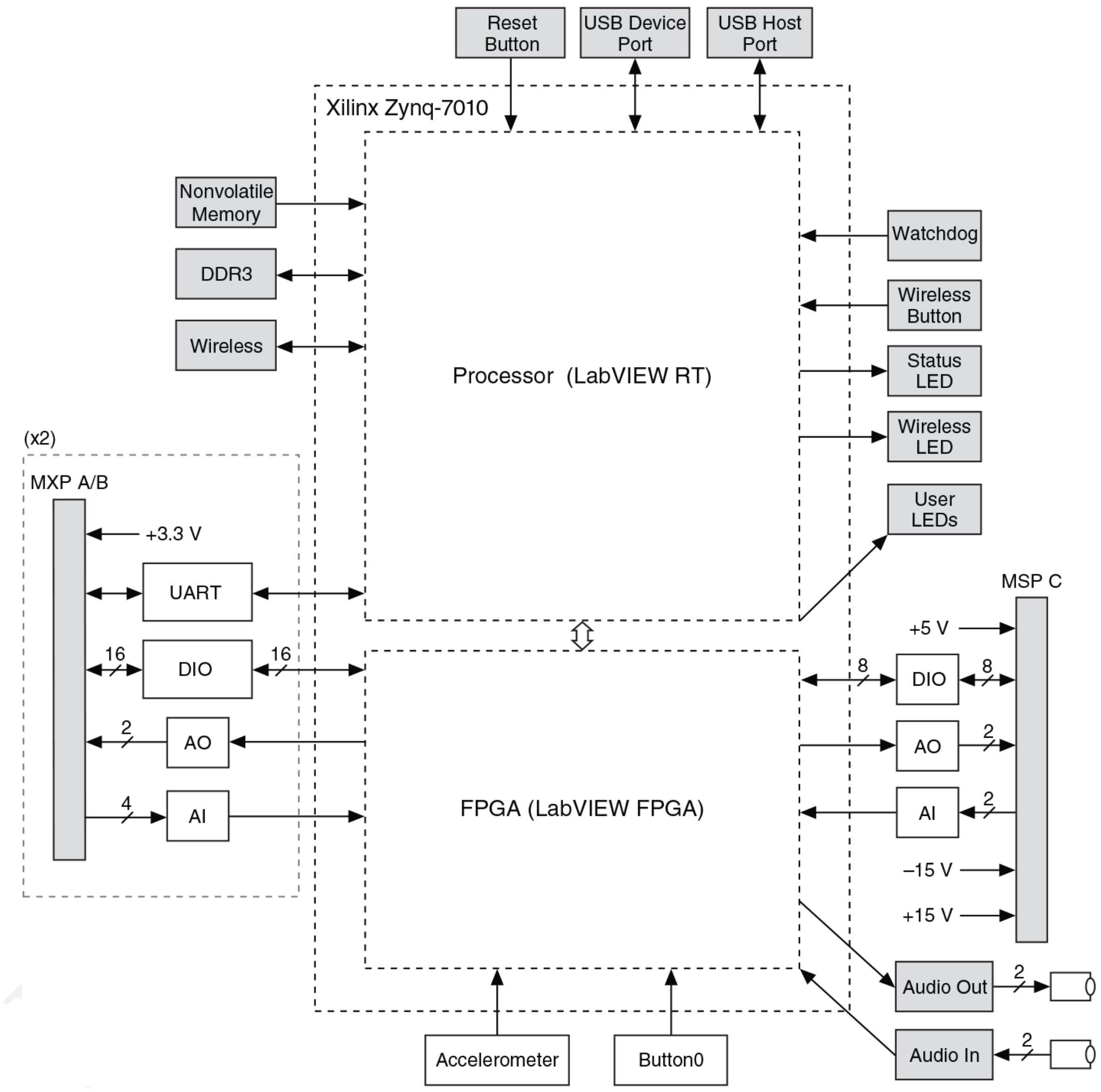

Figure 00.11: myRIO-1900 Hardware Block Diagram (source: Instruments (2013))