Chapter 6: Using the Debugger

Time 1 Hour

At the end of this chapter you should understand how to use different debugging techniques.

6.1 Debugger Introduction 2

6.2 Make Target 2

6.3 Running the Debugger 2

### 

# Debugging Introduction

# BTSpy

# Make Target

In order to use the debugger, create a new make target for an existing project so that *–debug* is added after the platform name (with no space) and remove run from the end of the target. That is, the target should look like:

*<folder1>.[<folder2>…].<project>-<platform>-debug download*

For example, the make target for the 02\_blinkled project from the previous chapter would be:

*ww101.02.02\_blinkled-CYW943907AEVAL1F\_WW101-debug download*

# Running the Debugger

First, execute the make target created above to download the program to the board. Once the project is downloaded, click the down arrow next to the green bug icon and select “43xxx\_Wi-Fi\_Debug\_Windows”. If you get a message asking if you want to open the debug perspective, click “Yes”. You can click the check box to tell the tool to switch automatically in the future.



Note: If you get an error when trying to launch the debugger you may need to terminate an existing debug process. Open the Windows Task Manager, select the Process tab, click on “Image Name” to sort by the process name and terminate all “arm-none-eabi-gdb” processes.

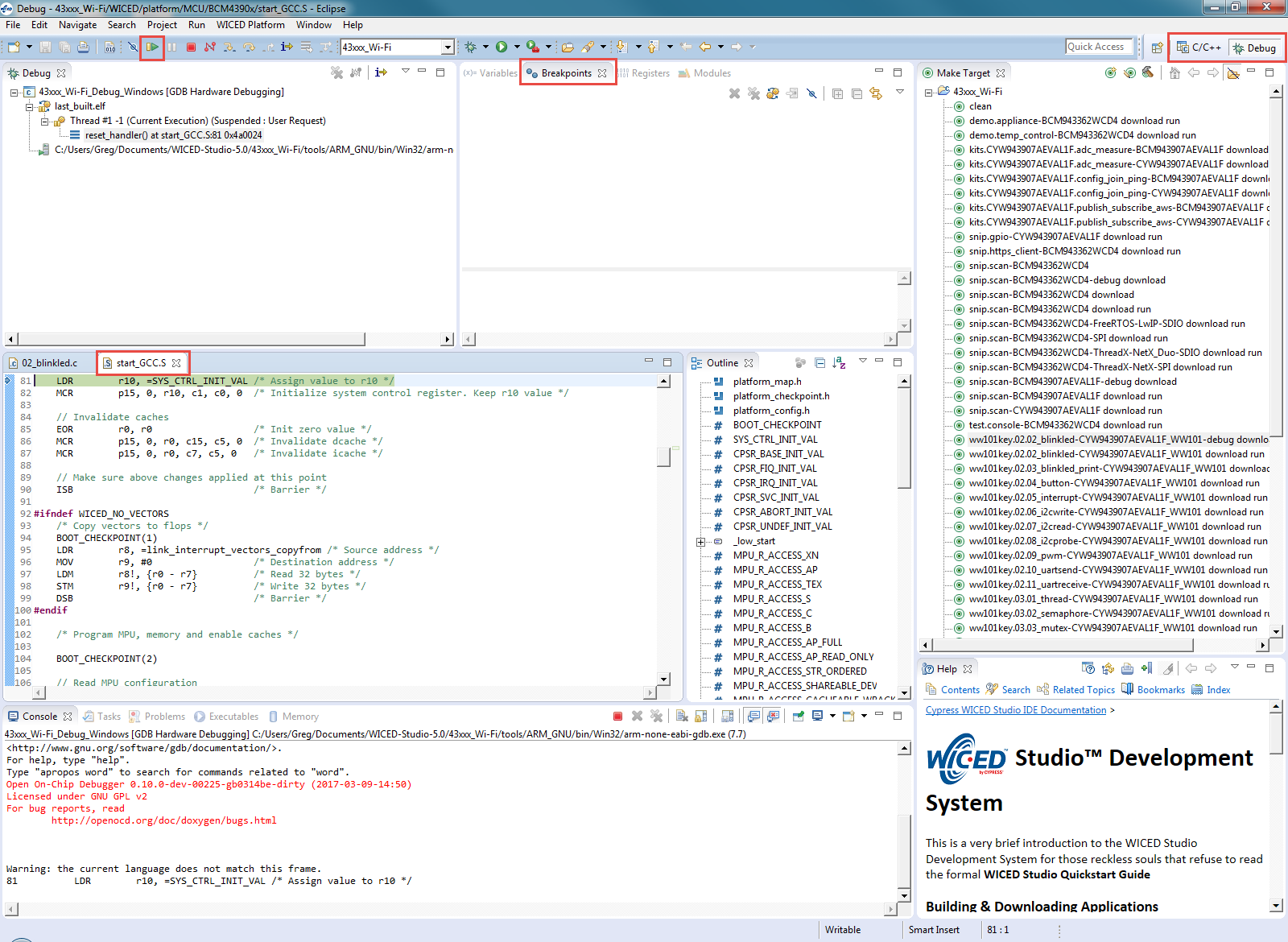


You may also need to set some of the debug options. These can be found in the community article at:

<https://community.cypress.com/community/wiced-wifi/wiced-wifi-forums/blog/2014/05/09/creating-andor-editing-debug-configurations>

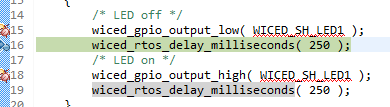
If you still get an error, execute the “clean” make target and then re-execute the make target for the project that you want to debug.

When the debugger starts, you will be in the “Debug Perspective”. The session will halt in the start\_GCC.s file. The top of the window will look something like this:

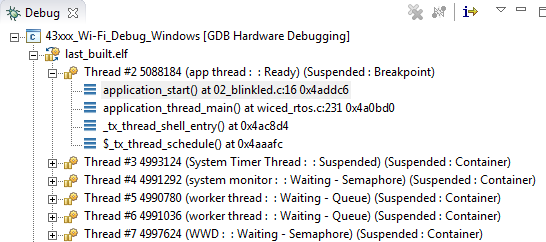


In order to add a breakpoint, open the source file (such as 02\_blinkled.c), click on the line where you want a breakpoint and press Ctrl-Shift-B or from the menu select “Run > Toggle Breakpoint”. If you need to see the project explorer window in order to open the source file, click on “C/C++” in the upper right corner in order to switch to the C/C++ Perspective. Once you have opened the file, switch back to the Debug Perspective.

Click the “Resume” button (shown in the figure above) to resume execution. The program will halt once it reaches the breakpoint.

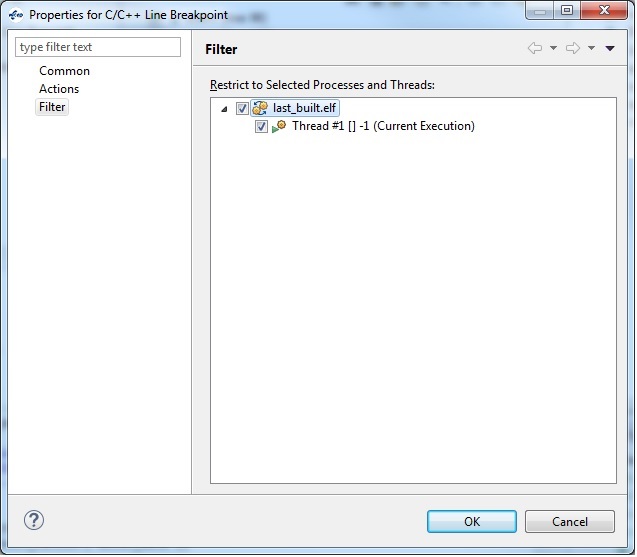


Once a thread suspends due to a breakpoint you will see that line of code highlighted in green as shown above and you will see that the thread is suspended due to the breakpoint in the debug window as shown below.

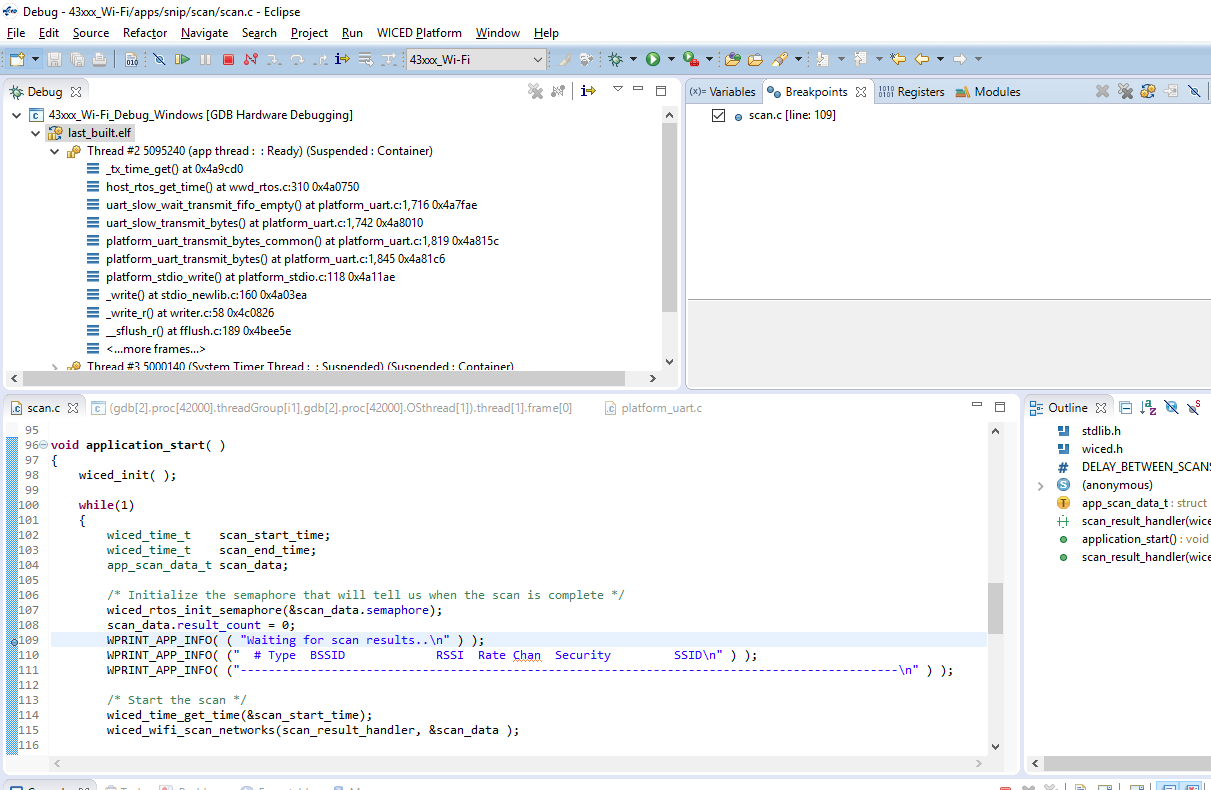


You can enable or disable breakpoints by double clicking on the green circle next to the line in the source code or from the “Breakpoints” window. If you don’t see the Breakpoints tab, use the menu item “Window > Show View > Breakpoints”.

If breakpoints are created prior to starting the current debug session, they will not be associated with the current thread and will be indicated with a blue circle without a check mark. To enable the breakpoints in the current thread, right-click the desired breakpoint and select “Breakpoint Properties…” Click on “Filter” and then select the “last\_built.elf” check box as shown below.



Note: If you do not see any breakpoints in the Breakpoints window, click the “Show Breakpoints Supported by Selected Target” button as shown below.



Click the red “Terminate” button to stop debugging. Once you terminate the debugger, you will want to switch back to the C/C++ Perspective by clicking on the button at the top right corner.

