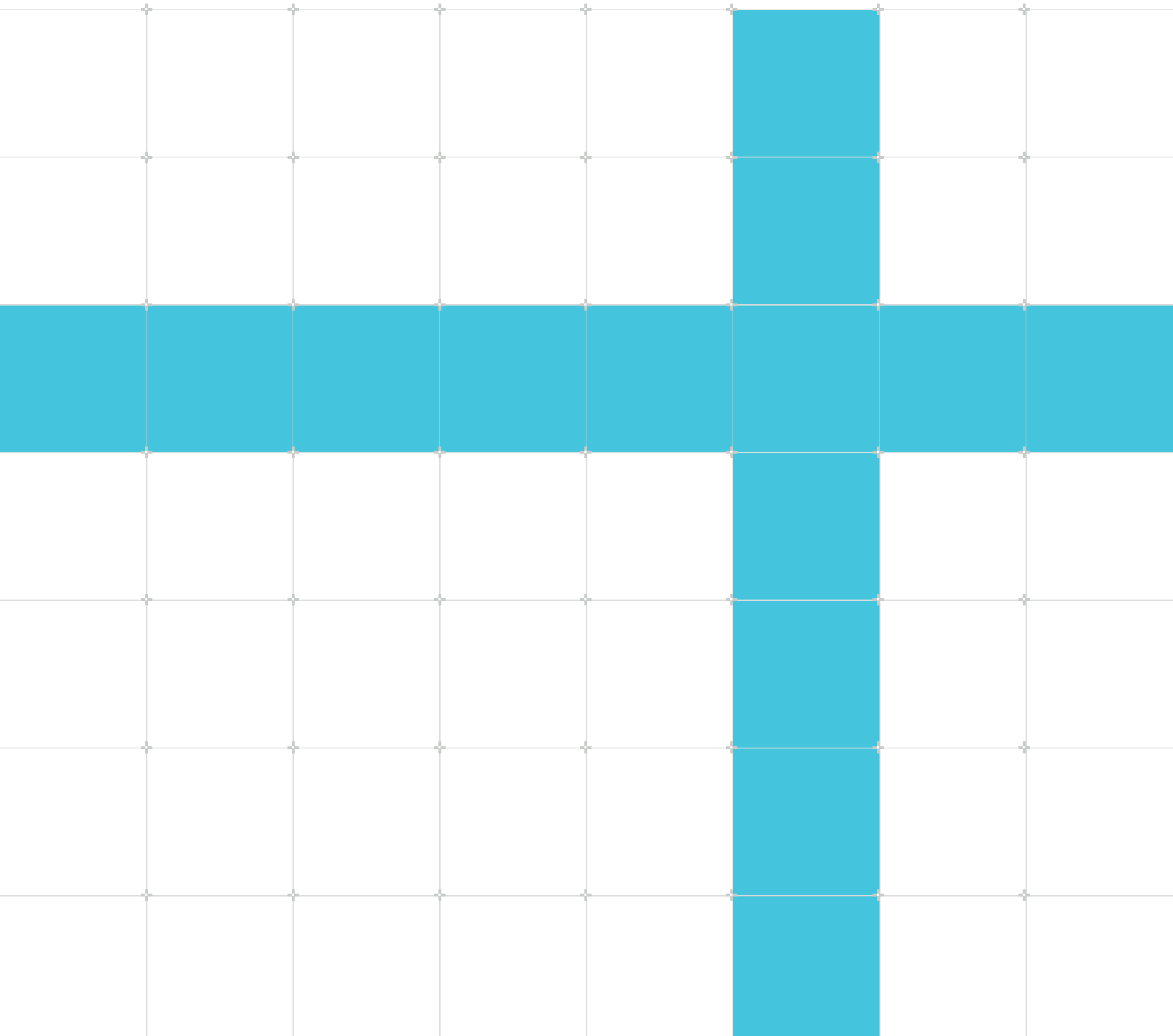
arm

Linux Application Debug Tutorial

Version 1.0

Issue 01
102625_0100_01_en



Linux Application Debug Tutorial

Release information

Document history

Issue	Date	Confidentiality	Change
0100-01	1 January 2020	Non-Confidential	First release

Proprietary Notice

or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349|version 21.0)

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Feedback

Arm® welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Contents

1. Prerequisites

The following list items are prerequisites for debugging Linux applications:

- Downloaded DS-5 CE as either a complete IDE or an Eclipse plugin, and installed it on your host machine (Windows, Linux or, for plugin-only installation, Mac).
- Prepared your Arm-based Linux target platform using the instructions on how to build a kernel and setup a file system for your own board (see manufacturers' website).
- To allow Linux application debug, you need to ensure gdbserver is present on the target (apt-get install gdbserver on Ubuntu systems).
- To make use of the Remote System Explorer (RSE) that is provided within DS-5, a ssh daemon (sshd) must be running on the target.
- For file transfer, RSE makes use of sftp-server, so that must be present in your embedded Linux system.
- To run a graphics-intensive application like Xaos, you may need some graphics libraries, such as, gtk, cairo, pango, glib (available through apt-get).
- Have built your application. The Xaos application is provided within the examples supplied with DS-5 or can alternatively be downloaded from SourceForge.

2. Context

This tutorial shows how to debug the open source fractal rendering application Xaos running on Linux on an Arm Cortex-A8 processor-based development board using Arm Development Studio 5 (DS-5) Community Edition. You can also use DS-5 CE to develop Linux or Android applications on other Arm-based platforms in a similar way.

For the purposes of this tutorial, Xaos is built as a multi-threaded application.

3. Procedure

Linux application debug procedure:

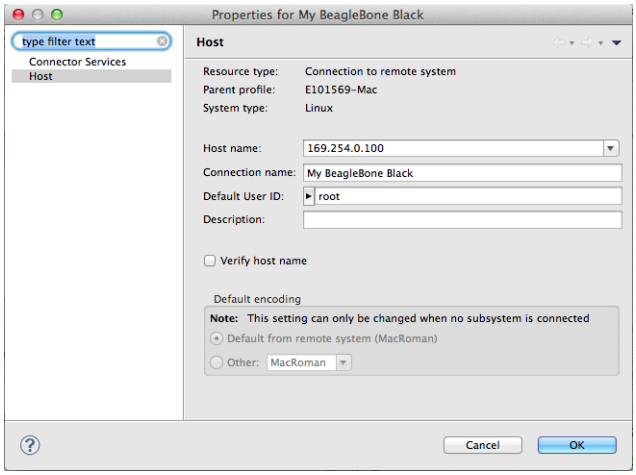
1. Set up a SSH connection using the RSE, and create a debug launch configuration that uses that connection.

   In the **Remote Systems** tab, create a **New connection**. Select **Linux**, then click **Next**. Enter the IP Address, and name it (for example `My BeagleBone Black`).

   Click **Next**. Select **ssh.files**, then click **Finish**.

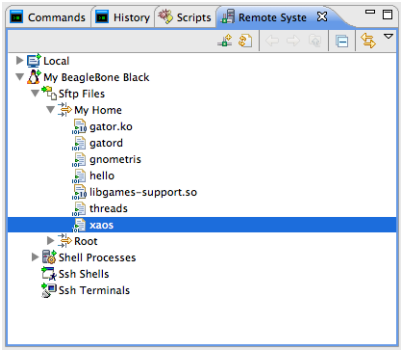
   When you first try to connect to the target, RSE prompts you to enter the User ID and Password. The resulting properties for the connection look similar to:

   Figure 3-1: User ID and password prompt in the RSE window.



   Following a successful RSE connection, you can drill down into the remote target's file system in Eclipse, and drag and drop files from host to target:

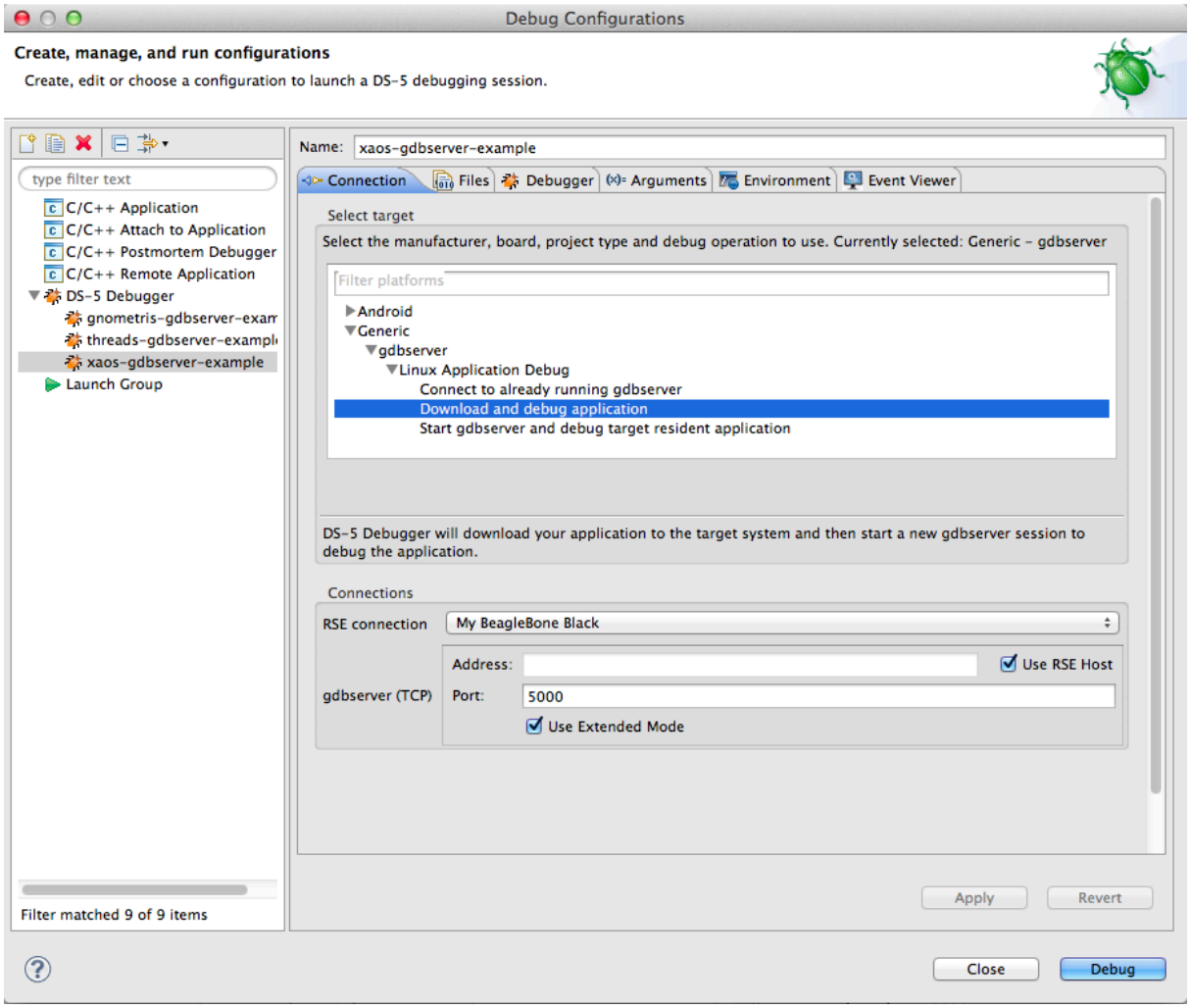
   Figure 3-2: RSE file system



2. To create a debug launch configuration that uses that RSE connection, select **Run** > **Debug Configurations...**. Create a **New launch configuration** and give it a name, for example, `xaos-gdbserver-example`.
3. In the target tree view of the **Connection** tab, select **Generic** > **Linux Application Debug** > **Download and debug application**.

   The RSE connection made earlier is automatically used by default:

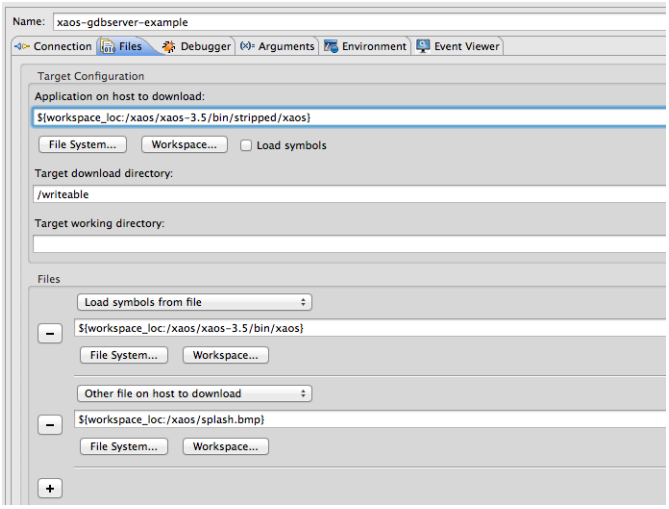
Figure 3-3: Tree view of Connection tab



4. In the **Files** tab, enter the stripped executable to download, and the location of the unstripped executable containing the debug information to allow debug at source level.

   The resulting dialog looks similar to:

Figure 3-4: Files tab dialog

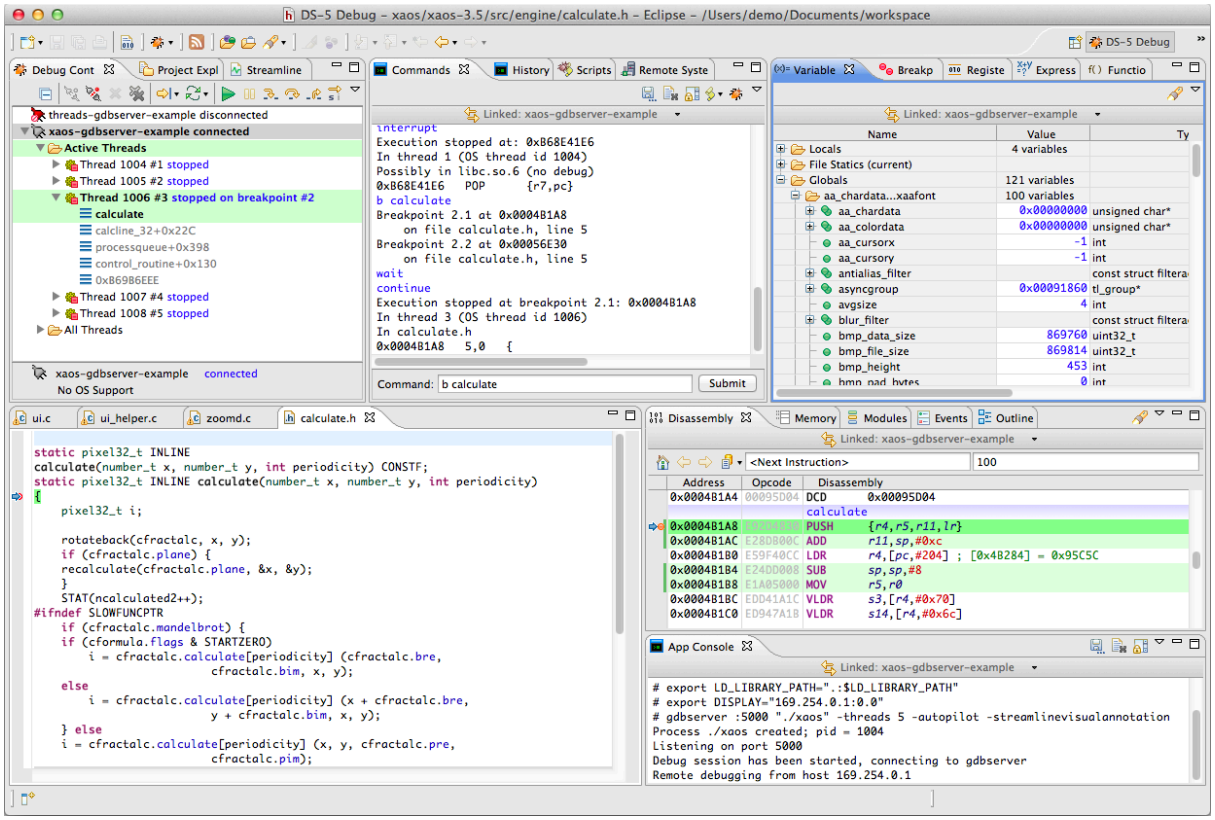


5. Click **Debug**.

   DS-5 Debugger downloads the stripped executable onto the target via RSE, starts gdbserver on the target, reads the debug information from the unstripped executable, and runs to a breakpoint at main().

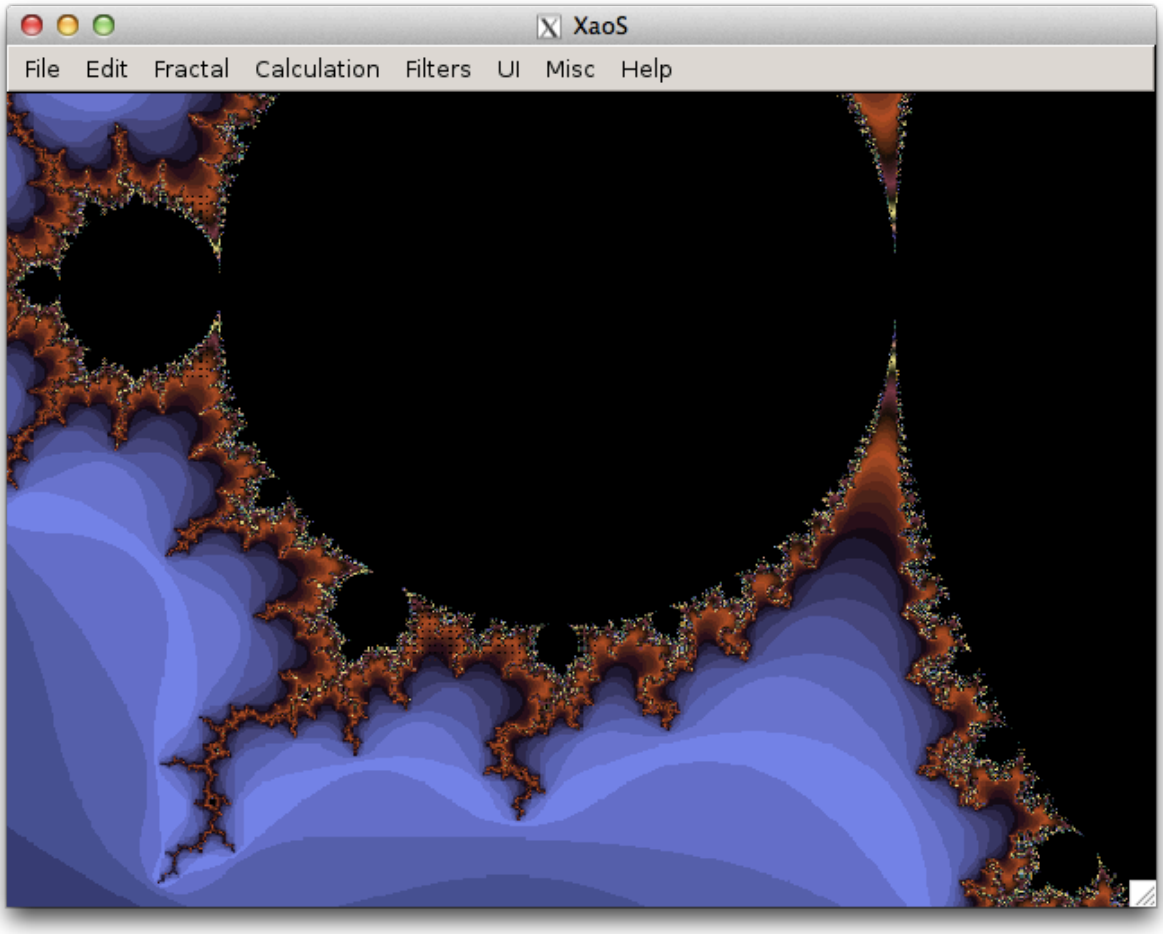
   You can view the source, run, stop, single-step, set breakpoints, set watchpoints, view registers, view variables, view disassembly, view memory, view symbols, view the stack backtrace for each thread, and so on.

Figure 3-5: Debug view



6. To view the graphical output from Xaos either:
   - Connect an LCD screen to the board
   - Redirect the graphical screen output to the XQuartz X server by setting, for example, `export DISPLAY=[host IP address]:0.0` in the **Environment** tab of the **Debug Configurations** dialog.

Figure 3-6: Example graphical output from Xaos



This display uses the XQuartz X server to view the graphical output from Xaos.

4. Results

That is it. You are debugging an Arm Linux application.

5. Related information

Here are some resources related to material in this guide:

- Resources
- Tutorials
- DS-5 documentation