Freescale Semiconductor
Application Note

Document Number: AN4979

Linux Application Debug using CodeWarrior for QorIQ LS series - ARM V7 ISA and AppTRK

Contents

1. Introduction

This document describes the steps required for Linux application debugging using CodeWarrior Development Studio for QorIQ LS series - ARM V7 ISA and AppTRK.

This document includes the following sections:

- Preliminary background
- Creating an ARMv7 Linux application project
- Debugging Linux application

2. Preliminary background

This section describes the steps required to compile the Linux image that includes the AppTRK agent for the LS1 boards.

2.1. Download SDK

To debug a Linux application using AppTRK and CodeWarrior, download the latest QorIQ SDK from www.freescale.com or OpenWrt SDK from https://dev.openwrt.org/.

2.2. Install QorIQ SDK

To install QorIQ SDK with Yocto, see the instructions provided on Freescale Infocenter.

2.3. Install OpenWrt SDK

To install OpenWrt SDK, see OpenWrt SDK documentation on https://dev.openwrt.org/.

2.4. Build packages

AppTRK is the target resident application for debugging Linux applications using CodeWarrior. The first two subsections below describe the two ways available to make AppTRK run on the target board. The third subsection explains how to start AppTRK.

2.4.1. Deploy AppTRK into rootfs image

Perform these steps to deploy AppTRK into the rootfs image:

1. To add AppTRK to <image-target>, open the following file for editing:

   <yocto_install_path>/meta-fsl-networking/images/<image-target>.bb
   and add
   apptrk to *IMAGE_INSTALL*

2. Build the Linux image using the *bitbake* command:

```
$ cd <yocto_install_path>/
$ source ./build_<machine>_release/SOURCE_THIS
$ bitbake <image-target>
```

2.4.2. Deploy AppTRK directly into running Linux

Perform these steps to deploy AppTRK directly into the running Linux:

1. SDK includes an AppTRK version. Build the AppTRK using the *bitbake* command:

```
$ cd <yocto_install_path>/
$ source ./build_<machine>_release/SOURCE_THIS
$ bitbake apptrk
```

2. Check *<yocto_install_path>/build_<machine>_release/tmp/work/cortexa7hf-vfp-neon-fsl-linux-gnueabi/apptrk/git-r0/git* for the ELF file.

3. Secure Copy (SCP) the AppTRK executable to the target board.

NOTE	CodeWarrior for ARMv7 includes the source archive and ELF for AppTRK. Ensure that AppTRK was built with the same toolchain and the same floating point application binary interface (ABI) as the libraries from the ramdisk and also as the Linux kernel image.

NOTE	For some LS1 boards, the AppTRK is not available in SDK; therefore, ensure that the AppTRK and associated libraries are deployed on the target Linux. OpenWrt does not support the *libelf* package, which is needed for AppTRK. The *libelf* package can be found on https://dev.openwrt.org and it needs to be enabled using the *make menuconfig* command.

2.4.3. Start AppTRK

After the AppTRK executable was deployed into the root file system, start it using the following command:

```
root@ls1021aqds:~# apptrk :12345 &
[1] 1026
```

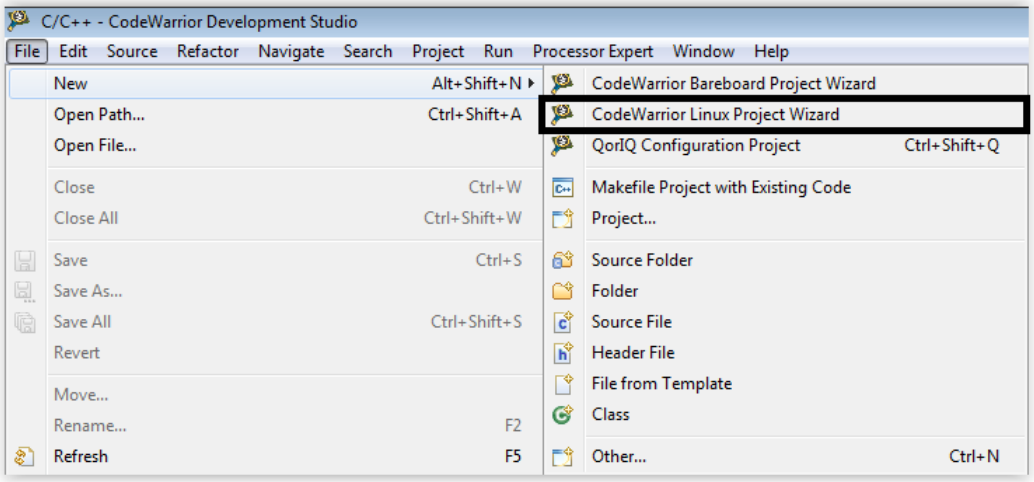
NOTE	Run apptrk in the background to be able to input commands in the same console where AppTRK was started, and use 12345 as the communication port to align with the CodeWarrior wizard.

3. Creating an ARMv7 Linux application project

To create an ARMv7 project for a Linux application, follow these steps:

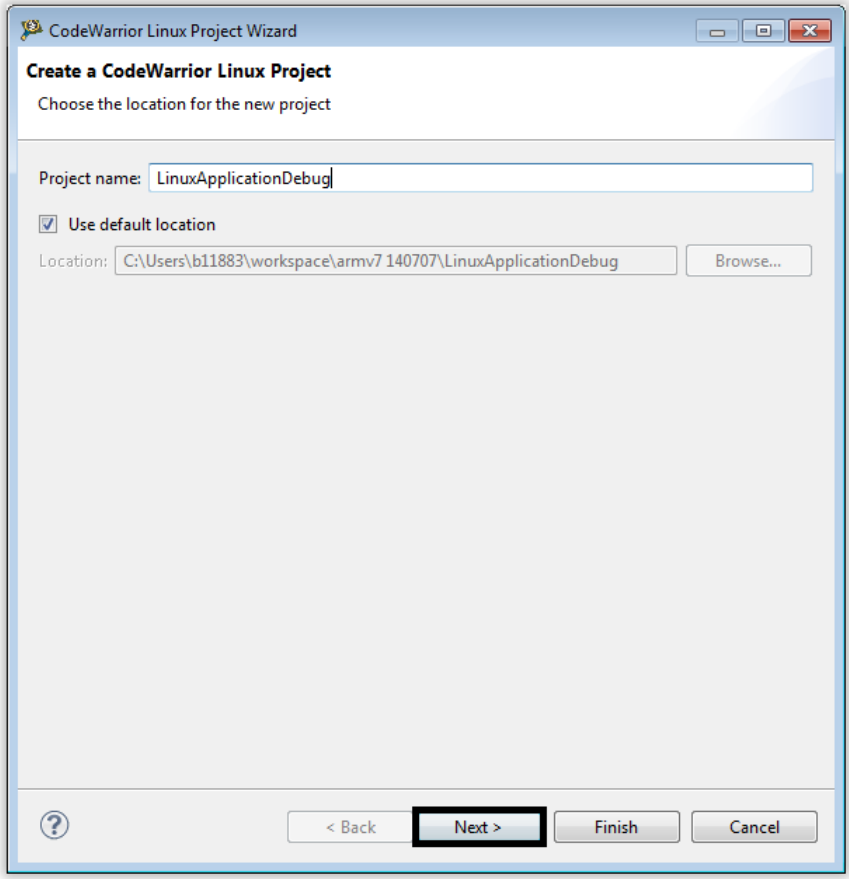
1. Start CodeWarrior for QorIQ LS series - ARM V7 ISA.
2. Choose **File** > **New > CodeWarrior Linux Project Wizard** to create a new Linux project.

Figure 1. CodeWarrior File menu option



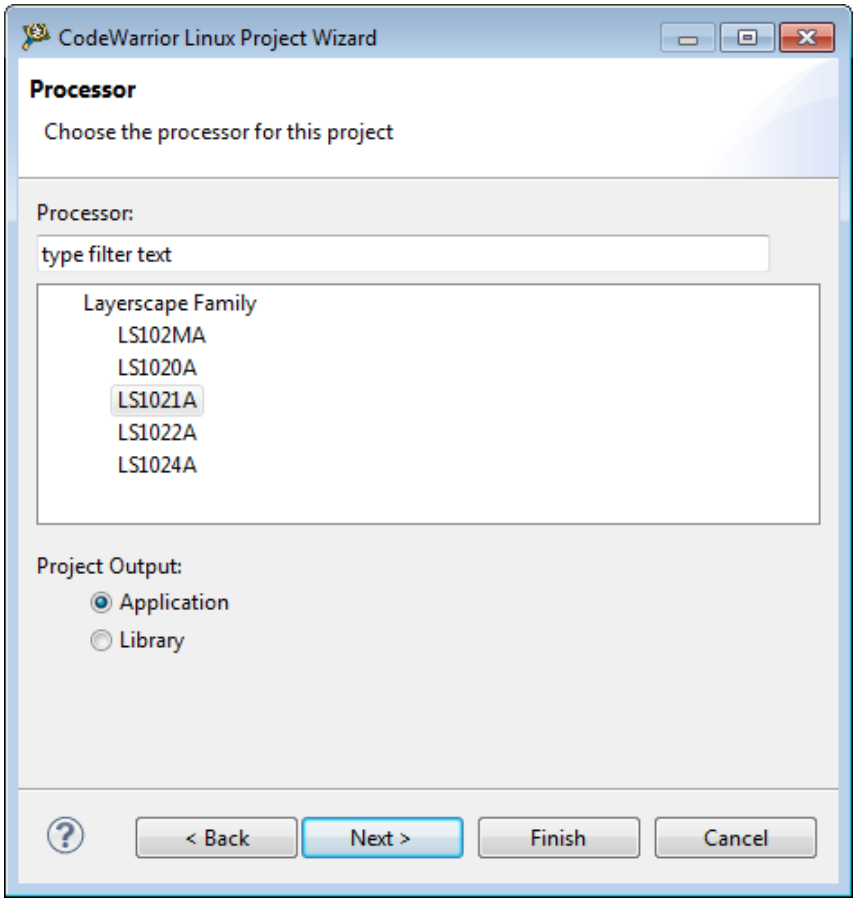
3. Specify project name and location, or use the default location, and click **Next**.

Figure 2. CodeWarrior Linux Project Wizard



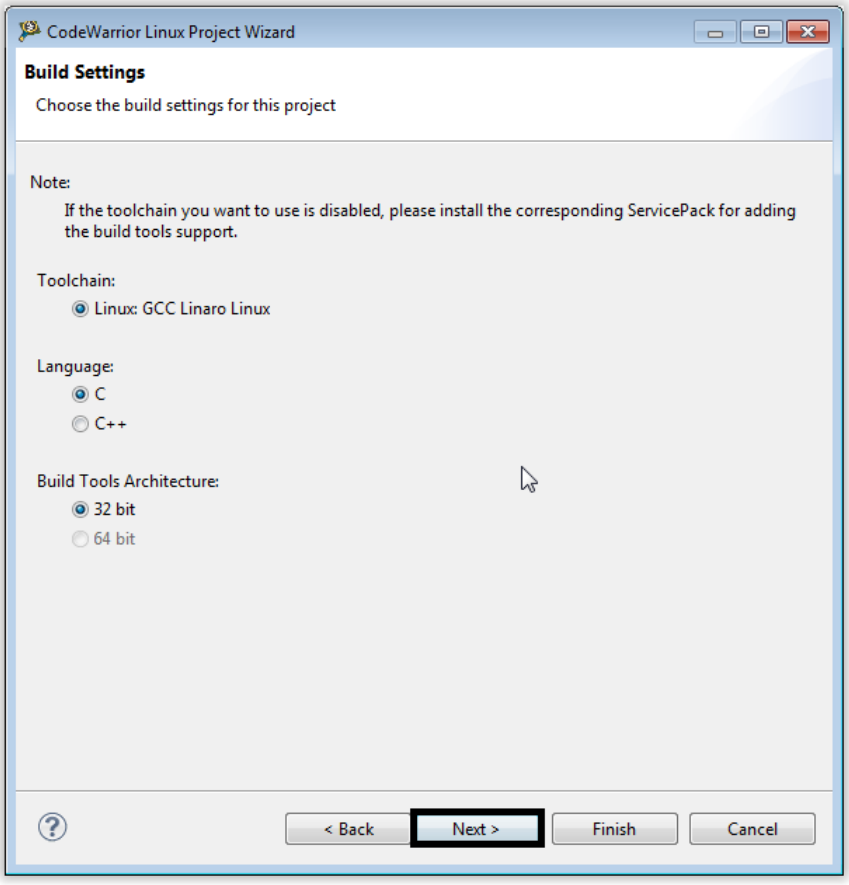
4. Select processor type and project output, and click **Next**.

Figure 3. Processor page



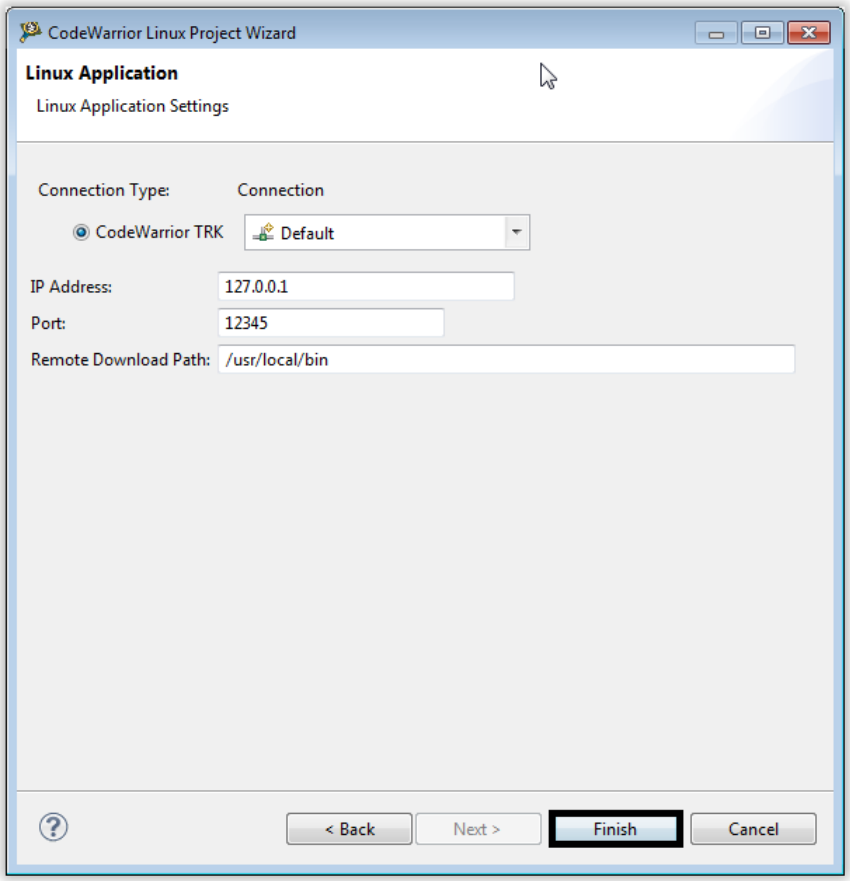
5. Configure build settings, and click **Next**.

Figure 4. Build Settings page



6. Configure connection details, and click **Finish** to create the Linux application project.

Figure 5. Linux Application page



NOTE	Ensure that the path you specify in **Remote Download Path** exists on target Linux.

4. Debugging Linux application

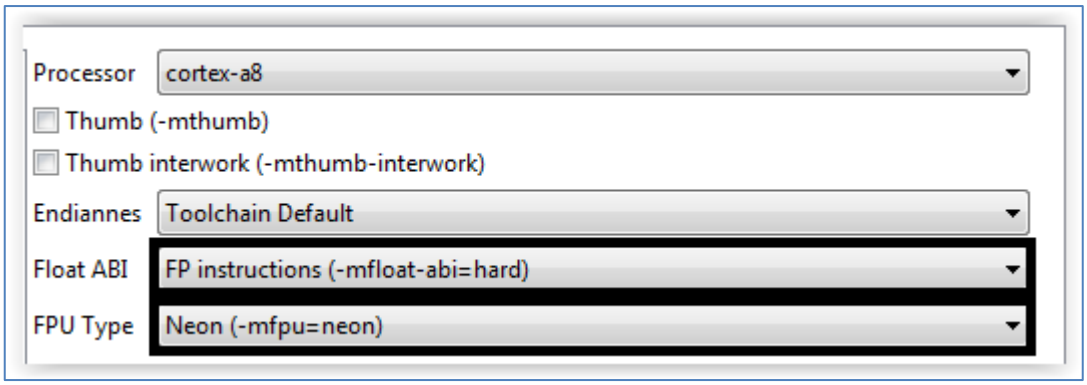
Before starting a debug session, it is mandatory to ensure that the Linux application project is built with the same toolchain version and floating point ABI as the rootfs and AppTRK were built.

Toolchain version used by SDK is gcc-linaro 4.8 (Aarch32) 4.8-2013.12 and by default, FP is set to *`-mfloat-abi=hard -mfpu=neon`*.

To use the same settings for your Linux application project, follow these steps:

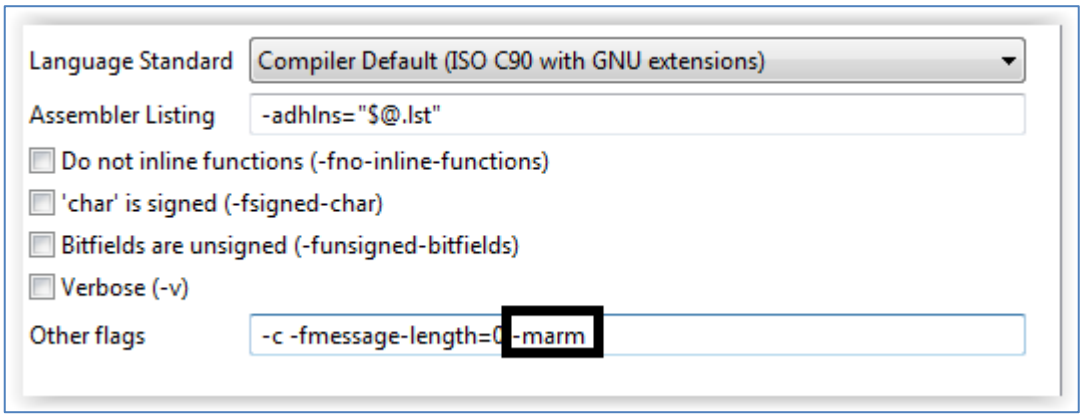
1. Go to **Project Properties > C/C++ Build > Settings > Target Processor**, and select the options as shown below.

Figure 6. Target processor options



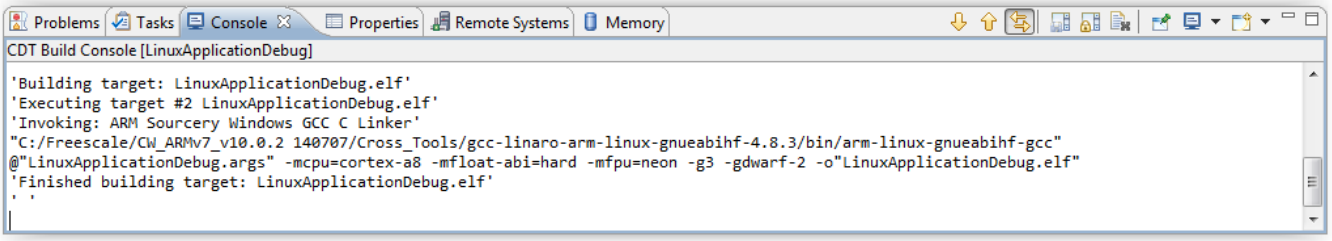
2. Go to **Project Properties > C/C++ Build > Settings > ARM Sourcery Windows GCC C Compiler**, and add the *–marm* option, as shown below.

Figure 7. ARM Sourcery Windows GCC C compiler options



3. Build the project. The **Console** view will display the finish building target message when the build is complete.

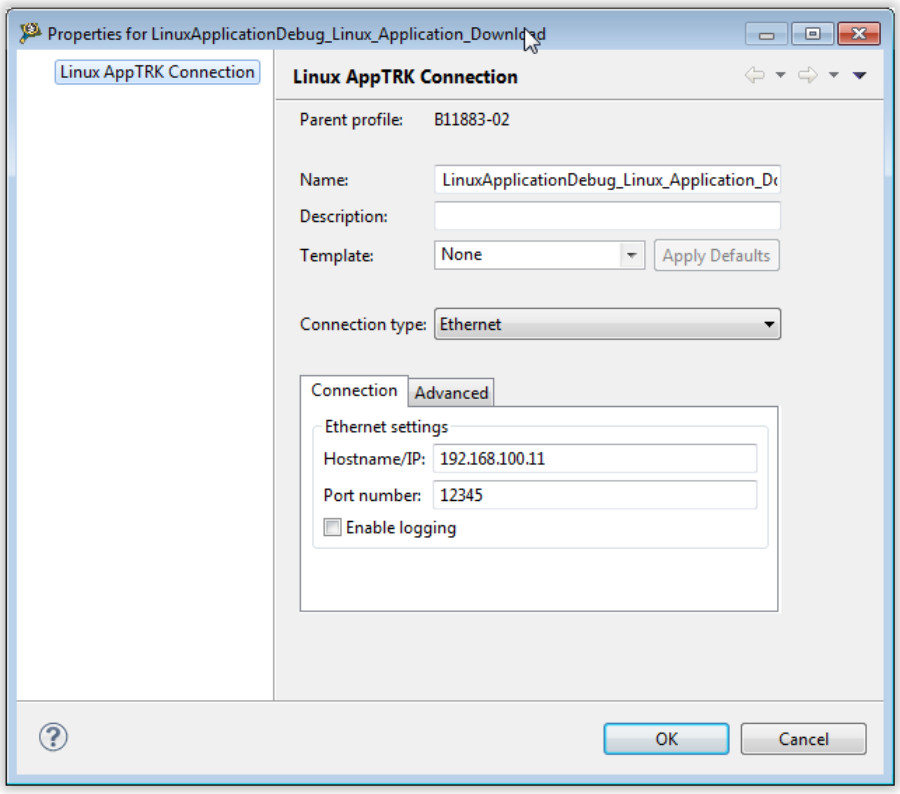
Figure 8. Build output in Console view



NOTE	Linux application for LS102MA must be built with the compiler from OpenWrt SDK.

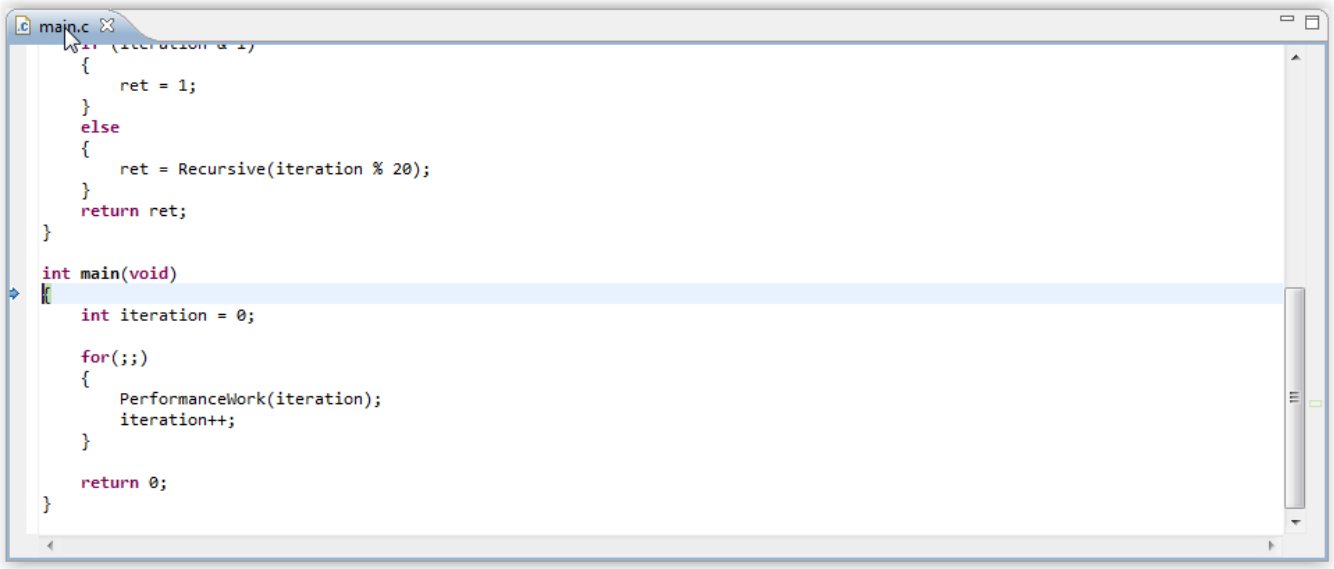
4. Check the debug configuration, **Hostname/IP** and **Port number**, and then start debugging.

Figure 9. Connection details



5. Using **Download Launch**, the Linux application will be downloaded in the folder specified in the **Remote download path** from the **Debug configurations** window, under **Debugger > Remote** tab. The program will stop at *main* function.

Figure 10. Linux application stopped at main function



6. Having the target application stopped at `main`, run control, setting/removal of breakpoints, reading/writing memory/registers can be performed.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

Document Number: AN4979

8 July 2015