Document Number: AN5277

Freescale Semiconductor Application Note

Using an External GCC Toolchain with CodeWarrior for Power Architecture

1. Introduction

This document explains how to use an external GNU compiler collection (GCC) toolchain with CodeWarrior for Power Architecture. This process is only applicable to the Linux version of CodeWarrior.

This document provides steps to:

- Build the toolchain supplied with Freescale Linux SDK
- Customize a Linux project to work with an SDK standalone toolchain
- Build the project using an external toolchain
- Import existing code as a makefile project

Contents

- 1. Introduction.....1
- 2. Preliminary background......2
- 3. Installing SDK standalone toolchain......2
- 4. Working with a Power Architecture Linux application project......2
- 5. Importing existing code as a Makefile project9



© 2016 Freescale Semiconductor, Inc. All rights reserved.

Preliminary background

2. Preliminary background

CodeWarrior for Power Architecture may or may not include the Freescale PPC or GCC binary toolchain. If you are developing a Linux user space application with CodeWarrior, then you should use the toolchain supplied with the Freescale Linux SDK.

3. Installing SDK standalone toolchain

You can use the standalone toolchain provided in SDK to build a Linux user space application with CodeWarrior. To build and install the standalone toolchain with Yocto, perform these steps:

```
$ cd <sdk_install_path>
$ source ./fsl-setup-env -m <machine>
$ cd build_<machine>_release
$ bitbake fsl-toolchain
$ cd build_<machine>_release/tmp/deploy/sdk
$ ./fsl-networking-eglibc-<host-system>-<core>-toolchain-
<release>.sh
```

NOTE	The default installation path for the standalone toolchain is: /opt/fsl-
	networking/. You need to specify this path while installing the standalone
	toolchain.
	For additional information about building and installing the standalone toolchain
	with Yocto, see SDK Knowledge Center.

See Change toolchain for using SDK standalone toolchain as the default build tool in CodeWarrior.

4. Working with a Power Architecture Linux application project

This section contains the following subsections:

- Create a stationary project for Linux application
- Change toolchain
- Verify build settings
- Build project using an external toolchain

4.1. Create a stationary project for Linux application

To create a Power Architecture stationary project for Linux application, follow these steps:

1. Start CodeWarrior for Power Architecture.

Using an External GCC Toolchain with CodeWarrior for Power Architecture Application Note

- 2. Choose File > New > CodeWarrior Linux Project Wizard from the CodeWarrior IDE menu bar. CodeWarrior Linux Project Wizard starts.
- 3. Specify the project name and location.
- 4. Select the processor and project output.
- 5. Configure the build settings.
- 6. Configure the connection details and click **Finish** to create the Linux application project.

NOTE To create Linux application projects, you need to install Linux build tool.

CodeWarrior does not provide Linux project wizard for the MPC8323 processor. If you need to create a new Linux project using the 8323 processor, then choose any processor at step 4 and follow the steps from next section.

Importing an example project can also be a starting point for creating a Linux application.

4.2. Change toolchain

By default, the stationary project for Linux application includes the Freescale PPC or GCC binary toolchain. In case, CodeWarrior does not provide the required toolchain, then you can add a default toolchain and an external build tool, by following these steps:

- 1. Choose **Project > Properties** from the CodeWarrior IDE menu bar. The **Properties** dialog appears.
- 2. Select C/C++ Build > Discovery Options in the left pane and specify Freescale Linux SDK standalone toolchain path in the Compiler invocation command field in the right pane.

Figure 1. Specify Freescale Linux SDK standalone toolchain path

😣 Properties for Linu	іх_Арр		
type filter text 🛛 🗷	Discovery Options - Linux_App		, , , , , , , , , , , , , , , , , , ,
 Resource Builders C/C++ Build Build Variables 	Configuration: Linux_Application	[Active]	Manage Configurations
Discovery Options Environment	Per Language		\$
Logging Settings	Tools:	Automated discovery of paths and symbols	
Tool Chain Editor C/C++ General	Power Linux Compiler	 Automate discovery of paths and symbols Report path detection problems 	
Run/Debug Settings		Discovery profile:	Power Linux C discov 💲
		Clear discovered entries now:	Clear
		Discovery profile options	
		Compiler invocation command	
		/opt/fsl-networking/QorIQ-SDK-V1.7-MPC83	2XMDS/sysro

NOTE Compiler invocation command should be something like this: /opt/fslnetworking/QorIQ-SDK-<release>/sysroot/<target_architecture>-fsl-linux.

- 3. Click **Apply** to apply the new settings.
- 4. Select C/C++ Build > Tool Chain Editor in the left pane and:
 - For an active configuration, choose **Power Architecture Linux C Debug Tool Chain** from the **Current toolchain** menu in the right pane
 - For a release configuration, choose **Power Architecture Linux C Release Tool Chain** from the **Current toolchain** menu in the right pane

Figure 2. Specify toolchain type

8 Properties for Linux	Арр		
type filter text 🛛 🗷	Tool Chain Editor - Linux_App		⟨⇒▼ ⊂⟩▼ ▼
 Resource Builders C/C++ Build 	Configuration: Linux_Application [Active]	* *	Manage Configurations
Discovery Options Environment Logging	✓ Display compatible toolchains only Current toolchain: Power Architecture Linux C Debug Tool Chain		:
Tool Chain Editor ► C/C++ General Pup/Debug Settings	Current builder: Power Linux Debug Builder		i k
Kair Debug Settings	Used tools Power Linux Linker Power Linux Compiler Power Linux Assembler Power Linux Preprocessor Power Linux Disassembler		Select Tools

NOTE A dialog appears, asking you whether you want to copy tool settings or not. You are recommended to copy compatible tool settings into the current toolchain.

- 5. Click **Apply** to apply the new toolchain.
- 6. Select C/C++ Build > Settings in the left pane and click the Build Tool Versions tab in the right pane, as shown in the figure below.

Figure 3. Add new toolchain

😣 Properties for Linux	х_Арр		
type filter text 🛛 🗷	Settings - Linu	к_Арр	⇔ ▼ ⇒
 Resource Builders C/C++ Build Build Variabler 	Configuration:	Linux_Application [Active]	Manage Configurations
Discovery Options	Tool Settings	➢Build Steps ●Build Artifact Banary Parsers ② Error Parsers Build	Tool Versions
Environment	Available versio	ons for current build tools. Default version is in bold letters.	Add
Logging	Version	Path	
Tool Chain Editor ► C/C++ General Run/Debug Settings	4.4.1	\${CROSS_TOOLS_HOME}/powerpc-linux/bin	Edit Delete Set As Default

- 7. Click Add and browse for the new toolchain location. The default installation path for Freescale Linux SDK standalone toolchain is: /opt/fsl-networking/QorIQ-SDK-<release>/sysroot/<host-system>/usr/bin/powerpc-fsl-linux/
- 8. Click Apply to apply the new toolchain.
- 9. Select the new toolchain and click **Set As Default**, as shown in the figure below. This will make the new toolchain as the default toolchain for the project.

Figure 4. Change default toolchain

No.	Tool Settings	Build Steps	Build Artifact	Binary Parsers	😣 Error Parsers	Build Tool Versi	ons	
	Available versions for current build tools. Default version is in bold letters.							
	Version		m d in					
	sdk_toolchain /opt/fsl-networking/QorIQ-SDK-V1.7-MPC832XMDS/sysroots/x86_64-fslsd							
	sdk_toolchain	/opt/fsl-net	working/QorIQ-	SDK-V1.7-MPC832	XMDS/sysroots/>	86_64-fslsd	Edit	
	sdk_toolchain 4.4.1	/opt/fsl-net \${CROSS_TC	working/QorIQ- OLS_HOME}/pow	SDK-V1.7-MPC832 /erpc-linux/bin	XMDS/sysroots/>	x86_64-fslsd	Delete	

4.3. Verify build settings

After setting the external toolchain as the default toolchain and before building your project, you should verify the build settings of the project. To verify build settings, follow these steps:

1. Select C/C++ Build > Settings in the left pane of the Properties dialog and click the Tool Settings tab in the right pane, as shown in the figure below.

Figure 5. Verify build settings

😣 Properties for Linu	х_Арр						
type filter text 🛛 🕱	Settings - Linux_	Арр				¢	∘∗⇔∗ ▼
 Resource Builders C/C++ Build Build Variables 	Configuration:	inux_Applicatio	Don [Active]	P p'		Manage Configu	rations
Discovery Options Environment Logging Settings Tool Chain Editor ► C/C++ General Run/Debug Settings	 Tool Settings Architecture Second Settings Architecture General General General Libraries Miscellane Shared Lib Power Linux Power Linux Preproces Symbols Includes Optimizat Optimizat Optimizat Optimizat Optimizat Miscellane Secenal Secenal Power Linux General Secenal Power Linux Preproces Power Linux Power Linux Power Linux Power Linux Power Linux Power Linux 	Build Steps	Build Artifact	ini Binary Parsers	C Error Parsers	Build Tool Versions	

2. Choose the correct architecture type from the Architecture menu on the **Architecture** page of the **Tool Settings** page. If the desired architecture type is not available for selection, then choose

Using an External GCC Toolchain with CodeWarrior for Power Architecture Application Note

none and specify the architecture type as compiler options on the **Miscellaneous** page, as shown in the figure below.





NOTE The example shown in the figure above is for the MPC8323 processor.

3. For PowerPC Linux assembler, compiler, linker, preprocessor, and disassembler, verify if the command is same as in the external toolchain (see the figure below).

Figure 7. Verify linker command



4. Click **OK** to save the project settings.

NOTE	Some build options may not be valid for a specific toolchain. For example, -
	fdebug-unwind-tables is not a valid option for the e300c2 toolchain. You can
	remove this option using the Other flags field on the C/C++ Build > Settings >
	Tool Settings > Power Linux Compiler > Miscellaneous page of the Properties
	dialog.

The SDK toolchain is a sysrooted toolchain. It means that toolchain will start to look for target fragments and libraries starting from the path specified by the sysroot option. To have a working build configuration, follow these steps:

- Go to the C/C++ Build > Settings > Tool Settings > Power Linux Compiler > Miscellaneous
 page of the Properties dialog and specify --sysroot="<path_to_target_sysroot>"
 in the Other flags field.
- 2. Go to the C/C++ Build > Settings > Tool Settings > Power Linux Linker > Miscellaneous page and specify --sysroot="vath_to_target_sysroot>" in the Linker flags field.

4.4. Build project using an external toolchain

To build the project, choose **Project > Build Project** from the CodeWarrior IDE menu bar. The project starts building and the **Console** view displays the progress of building the project, as shown in the figure

Using an External GCC Toolchain with CodeWarrior for Power Architecture Application Note

below.

Figure 8. Build project

```
Properties B Remote Systems
🕄 Problems 🧔 Tasks 🖳 Console 🖾
CDT Build Console [Linux App]
                                                           Ð
                                                                                    🗳 🕈 🔁 🕈
                                                             ER.
**** Build of configuration Linux Application for project Linux App ****
/sdk/Freescale/CodeWarrior PA 10.5.1/gnu/bin/make -j8 all
Building file: ../Sources/main.c
Executing target #1 ../Sources/main.c
Invoking: Power Linux Compiler
"/opt/fsl-networking/QorIQ-SDK-V1.7-MPC832XMDS/sysroots/x86 64-fslsdk-linux/usr/bin/
powerpc-fsl-linux/powerpc-fsl-linux-gcc" @"Sources/main.args" -MMD -MP -MF"Sources/main.d"
-o"Sources/main.o" "../Sources/main.c"
Finished building: ../Sources/main.c
Building target: Linux App.elf
Executing target #2 Linux App.elf
Invoking: Power Linux Linker
"/opt/fsl-networking/QorIQ-SDK-V1.7-MPC832XMDS/sysroots/x86 64-fslsdk-linux/usr/bin/
powerpc-fsl-linux/powerpc-fsl-linux-gcc" -o"Linux App.elf" @"Linux App.args"
Finished building target: Linux App.elf
```

5. Importing existing code as a Makefile project

This section contains the following subsections:

- Import Makefile project
- Build imported Makefile project using an external toolchain

5.1. Import Makefile project

To import a Makefile Project into CodeWarrior, follow these steps:

- 1. Choose **File > Import** from the CodeWarrior IDE menu bar. The **Import** wizard starts, displaying the **Select** page.
- 2. Choose C/C++ > Existing Code as Makefile Project and click Next. The wizard name and page name changes to Import Existing Code.
- 3. Specify project name, existing code location, language, and toolchain, as shown in the figure below.

Importing existing code as a Makefile project

Figure 9. Specify toolchain

8 Import Existing Code	
Import Existing Code	
Create a new Makefile project from existing code in that same directory	
Designation	
Import_Makefile	
Existing Code Location	
/sdk/test Brows	se
Languages	
✓ C □ C++	
Toolchain for Indexer Settings	
<none></none>	A
Linux GCC	J
Power Architecture Linux C Debug Tool Chain	
Power Architecture Linux C Release Tool Chain	
Power Architecture Linux C Static Library Debug Tool Chain	
Power Architecture Linux C Static Library Release Tool Chain	
Power Architecture Linux C++ Debug Tool Chain	
Power Architecture Linux C++ Release Tool Chain	
Power Architecture Linux C++ Static Library Debug Tool Chain	
Power Architecture Linux C++ Static Library Release Tool Chain	
Show only available toolchains that support this platform	
(?) < Back Next > Cancel Finish	

4. Click Finish. The Makefile Locations dialog appears asking for Makefile.

Importing existing code as a Makefile project

😣 Mal	cefile Locations
?	Makefile(s) are found in the project, please select which makefile to use as the default Build Location. The Build Location setting can be modified later as well in Properties dialog under C/C++ Build panel.
Proje	ct location (default/no makefile present)
/Impo	ort_Makefile/Makefile
	ОК

- 5. Click **OK**. The project is created.
- 6. Follow the steps from Change toolchain, to change the compiler invocation command and add the external build tool.

5.2. Build imported Makefile project using an external toolchain

To build the project, choose **Project > Build Project** from the CodeWarrior IDE menu bar. The project starts building and the **Console** view displays the progress of building the project, as shown in the figure below.

Figure 10. Build project

🕄 Problems 🖉 Tasks 📮 Console 🛿 🔲 Properties	📲 Remote Syster	ms					- 8
CDT Build Console [Import_Makefile]	Ŷ	৫ দ্বি		6		.	[] ▼
<pre>**** Build of configuration Power Linux C Debu /sdk/Freescale/CodeWarrior_PA_10.5.1/gnu/bin/m /opt/fsl-networking/QorIQ-SDK-V1.7-MPC832XMDS/ fsl-linux/powerpc-fsl-linux-gcc -m32 -msoft-fl networking/QorIQ-SDK-V1.7-MPC832XMDS/sysroots/ /opt/fsl-networking/QorIQ-SDK-V1.7-MPC832XMDS/ fsl-linux/powerpc-fsl-linux-gcc -m32 -msoft-fl networking/QorIQ-SDK-V1.7-MPC832XMDS/sysroots/</pre>	g for project In ake -j8 all sysroots/x86_64 oat -mcpu=e300c2 ppce300c2-fsl-li sysroots/x86_64 oat -mcpu=e300c2 ppce300c2-fsl-li	nport_M -fslsdk 2sys inux -c -fslsdk 2sys inux -c	Makefi (-linu (-linu (-linu (root=) test	ile * /opt 1.0 - IX/US /opt mai	r/bin/ /fsl- c main r/bin/ /fsl- .n.o	powerț .c powerț)C -

How to Reach Us:

Home Page: nxp.com

E-mail: nxp.com/support Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: <u>nxp.com/SalesTermsandConditions</u>.

Freescale, the Freescale logo, CodeWarrior, QorIQ, QorIQ Qonverge, and StarCore are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. All other product or service names are the property of their respective owners. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2016 Freescale Semiconductor, Inc.

Document Number: AN5277 10 May 2016

