

# Building Linux Kernel in CodeWarrior ARMv8

## 1 Introduction

This application note defines guidelines for configuring CodeWarrior for ARMv8 for Linux Kernel development.

This document explains:

- Installing standalone toolchain supplied with NXP Linux SDK
- Configuring CodeWarrior for ARMv8 for building Linux Kernel
- Building Linux Kernel with CodeWarrior for ARMv8

## 2 Requirements

For building Kernel using CodeWarrior for ARMv8, you need a host computer with Linux OS and CodeWarrior for ARMv8 Linux version installed.

### Contents

1	Introduction.....	1
2	Requirements.....	1
3	Installing SDK standalone toolchain.....	2
4	Configuring CodeWarrior for ARMv8 for building Linux Kernel.....	2
5	Building Linux Kernel using CodeWarrior for ARMv8.....	9



### 3 Installing SDK standalone toolchain

Linux SDK provides a standalone toolchain that can be used for building different application outside Yocto. In our case, we can use the standalone toolchain for building U-Boot using CodeWarrior for ARMv8.

To build and install the standalone toolchain with Yocto, perform these steps:

```
$ cd build_<machine>_release
$ bitbake fsl-toolchain
$ cd build_<machine>_release/tmp/deploy/sdk
$ ./fsl-qoriq-glibc-<host-system>-<core>-toolchain-<release>.sh
```

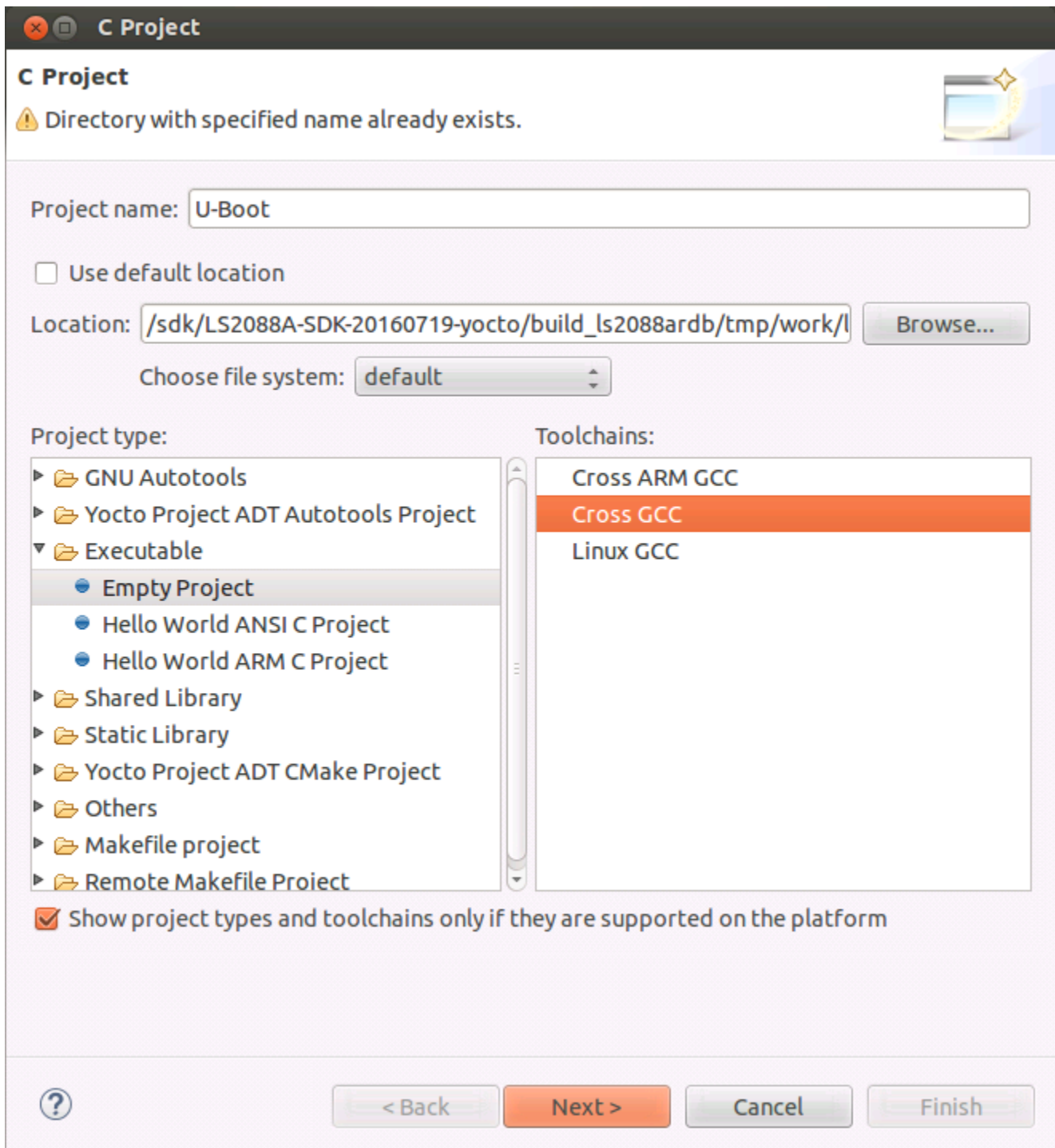
#### NOTE

The default installation path for the standalone toolchain is: /opt/fsl-qoriq/. You need to specify this path while installing the standalone toolchain. For additional information about building and installing the standalone toolchain with Yocto, see [https://freescale.sdlproducts.com/LiveContent/web/ui.xql?action=html&resource=publist\\_home.html](https://freescale.sdlproducts.com/LiveContent/web/ui.xql?action=html&resource=publist_home.html)

### 4 Configuring CodeWarrior for ARMv8 for building Linux Kernel

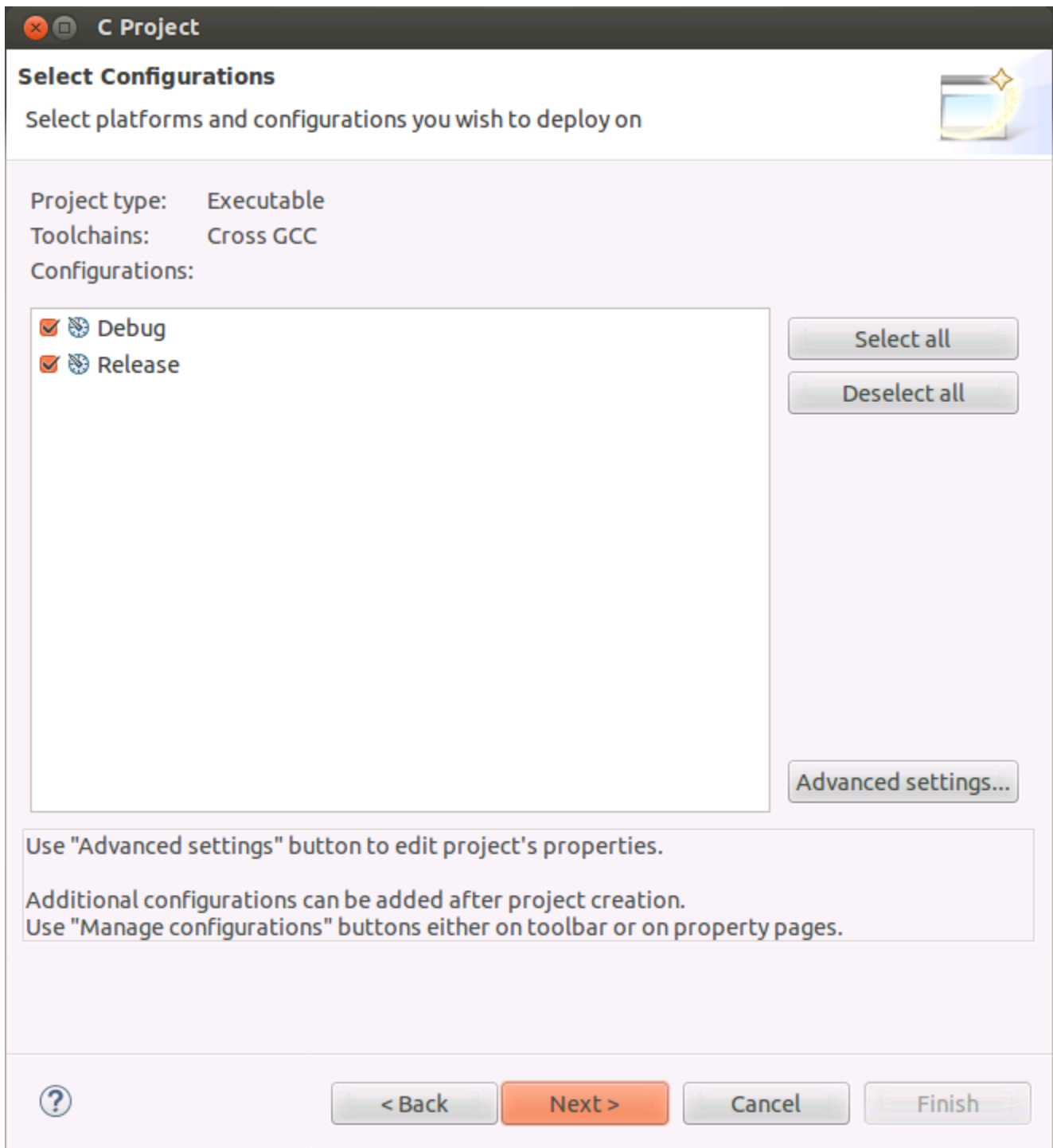
To create a project for building U-Boot inside CodeWarrior for ARMv8, perform these steps:

1. Select **File > New > C Project**.
2. Specify the project name and select **Empty Project** as Project type.
3. Uncheck the **Use default location** and click the **Browse** button to find the location for Linux Kernel source.
4. Select **Cross GCC** as **Toolchain**.
5. Click **Next**.



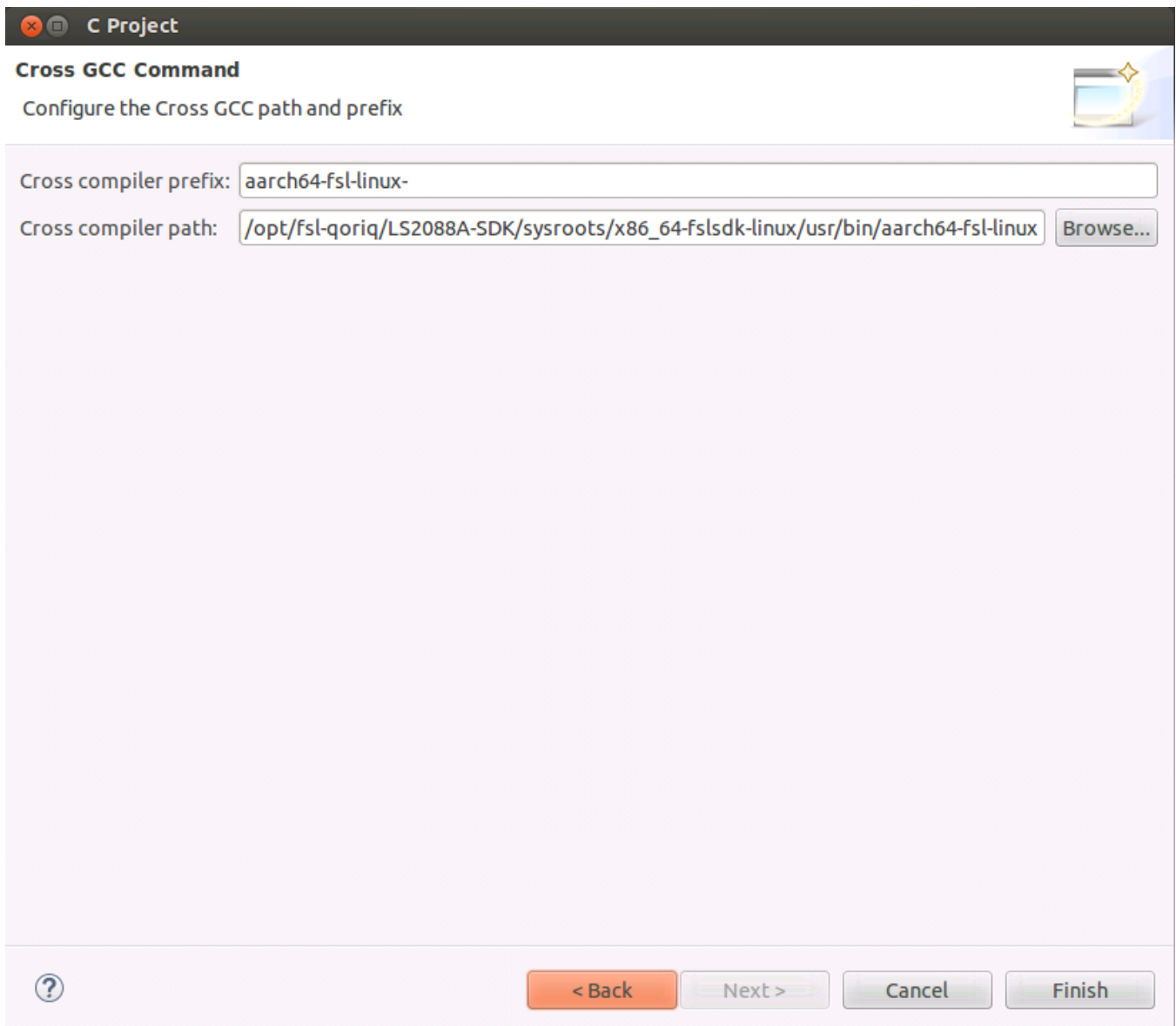
**Figure 1. C Project**

6. Select both **Debug** and **Release** configurations and click **Next**.



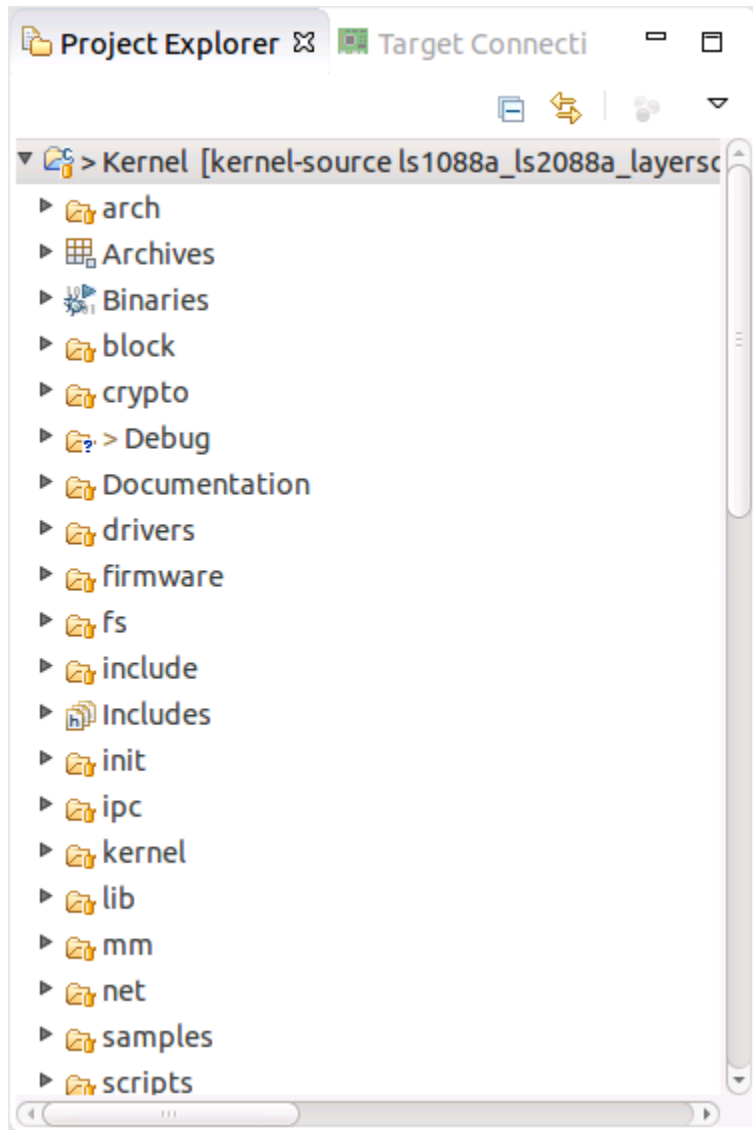
**Figure 2. Select configurations**

7. Specify the **Cross compiler prefix**, **Cross compiler path** and click **Finish**.



**Figure 3. Cross GCC Command**

8. Project is created and will appear in the **Project Explorer** view.



**Figure 4. Project Explorer**

9. Go to **Project > Properties > C/C++ build**, select **Builder settings** and uncheck **Generate Makefiles automatically**.

Builder Settings Behavior Refresh Policy

Builder

Builder type: External builder

Use default build command

Build command: make Variables...

Makefile generation

Generate Makefiles automatically  Expand Env. Variable Refs in Makefiles

Build location

Build directory: /sdk/LS2088A-SDK-20160719-yocto/build\_ls2088ardb/tmp/work/ls2088ardb-fsl-linux/lir

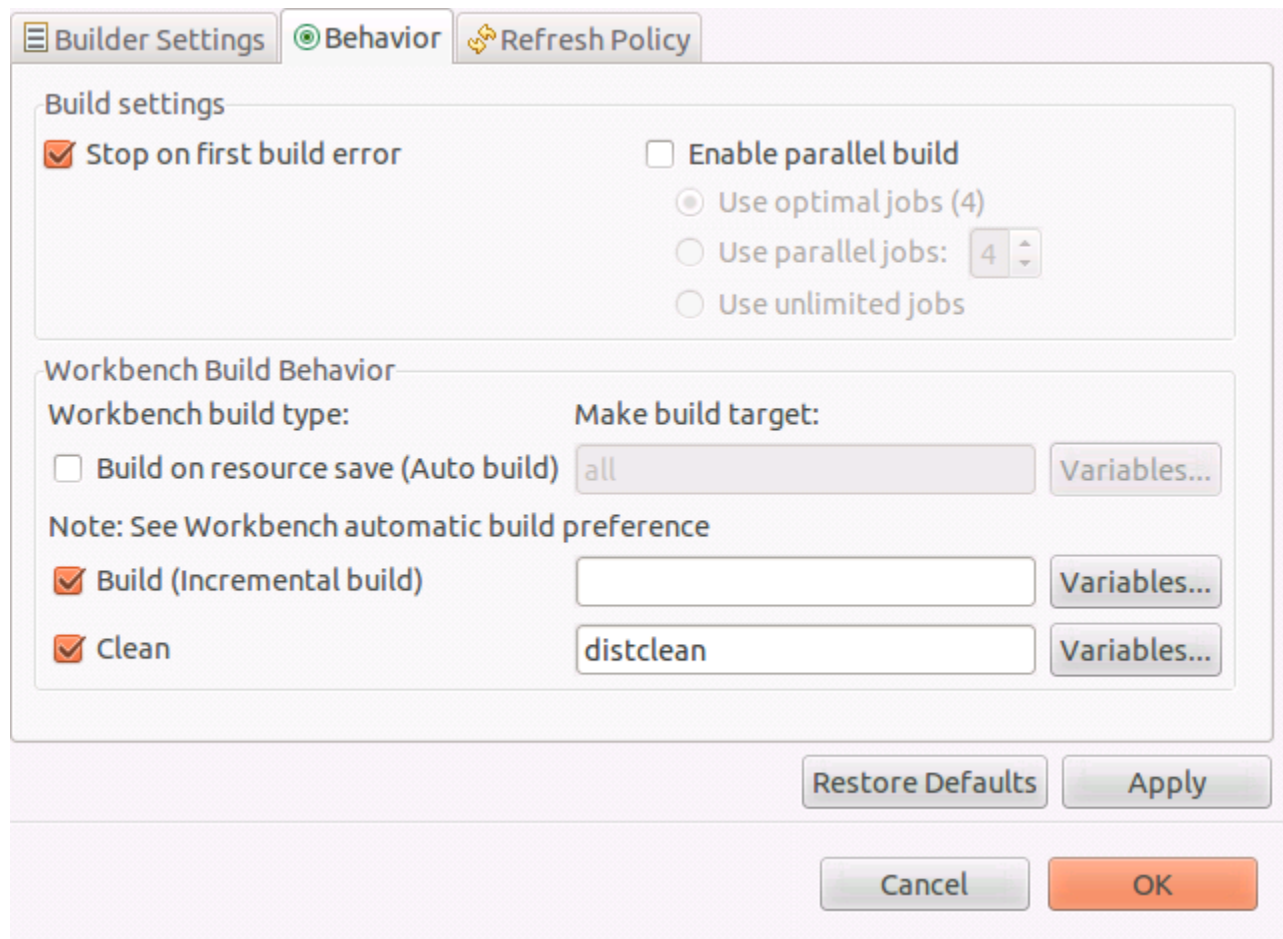
Workspace... File system... Variables...

Restore Defaults Apply

Cancel OK

**Figure 5. Builder settings**

10. Update the **Build directory** with Linux Kernel source code path.
11. Select **Behavior**, empty the **Build (incremental build)** field and change clean to distclean in **Clean** field.



**Figure 6. Behavior**

- Go to **Project > Properties > C/C++ build > Environment** and add environmental variables for:

Name: **CROSS\_COMPILE**

Value: **aarch64-fsl-linux-**

Click **Add to all configuration**

Name: **ARCH**

Value: **arm64**

Click **Add to all configuration**

Name: **PATH**

Value: **/opt/fsl-qorIQ/LS2088A-SDK/sysroots/x86\_64-fslsdk-linux/usr/bin:/opt/fsl-qorIQ/LS2088A-SDK/sysroots/x86\_64-fslsdk-linux/usr/bin/aarch64-fsl-linux:/usr/sbin:/usr/bin:/bin**

Click **Add to all configuration**

**NOTE**

When SDK standalone toolchain is built in other location than default, it is possible that other environmental variables must be set. Check the error from **Console** view and add the necessary variables.

- Go to **Project > Properties > C/C++ build > Settings** and uncheck **Elf Parser** and select **GNU Elf Parser**.



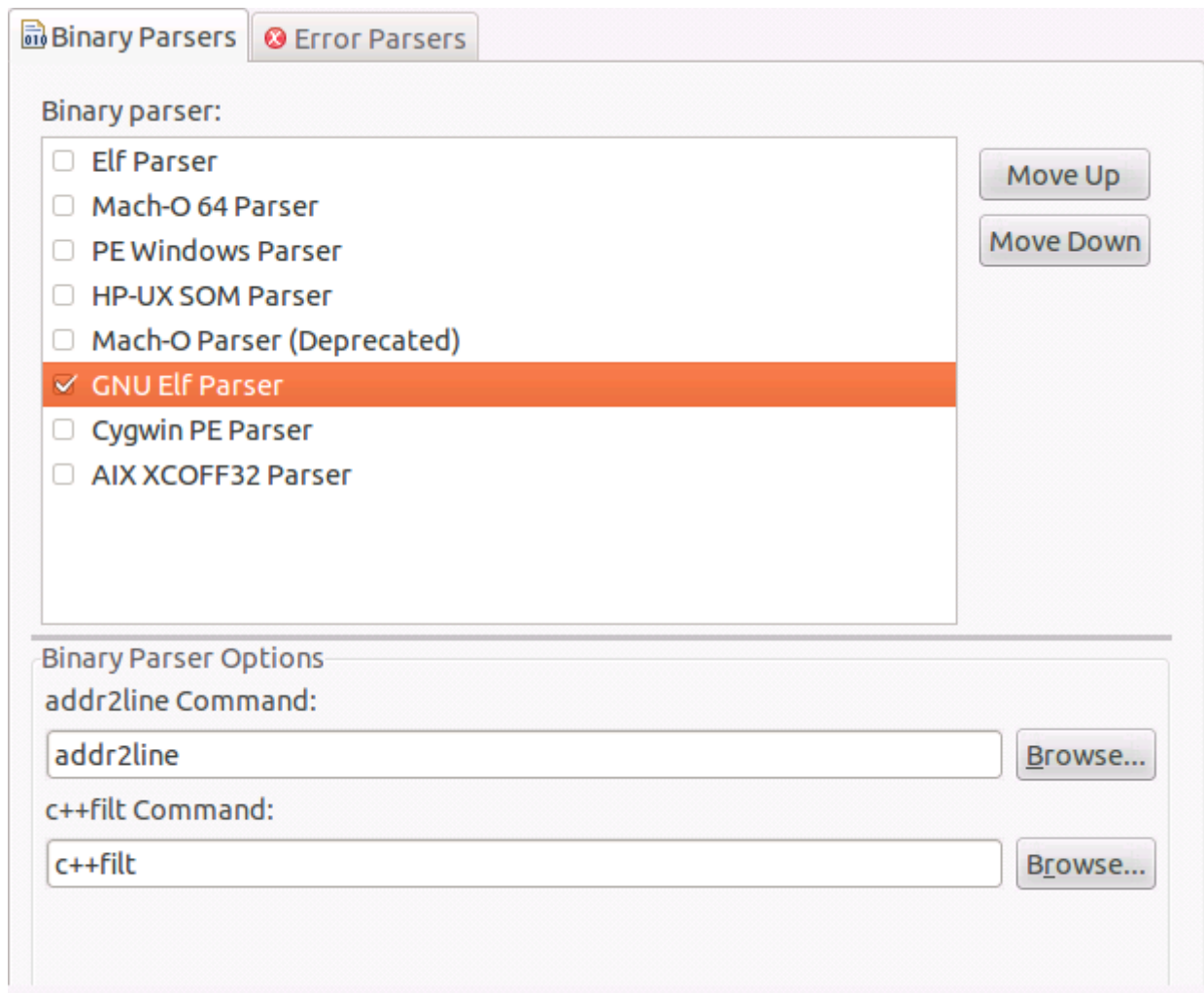
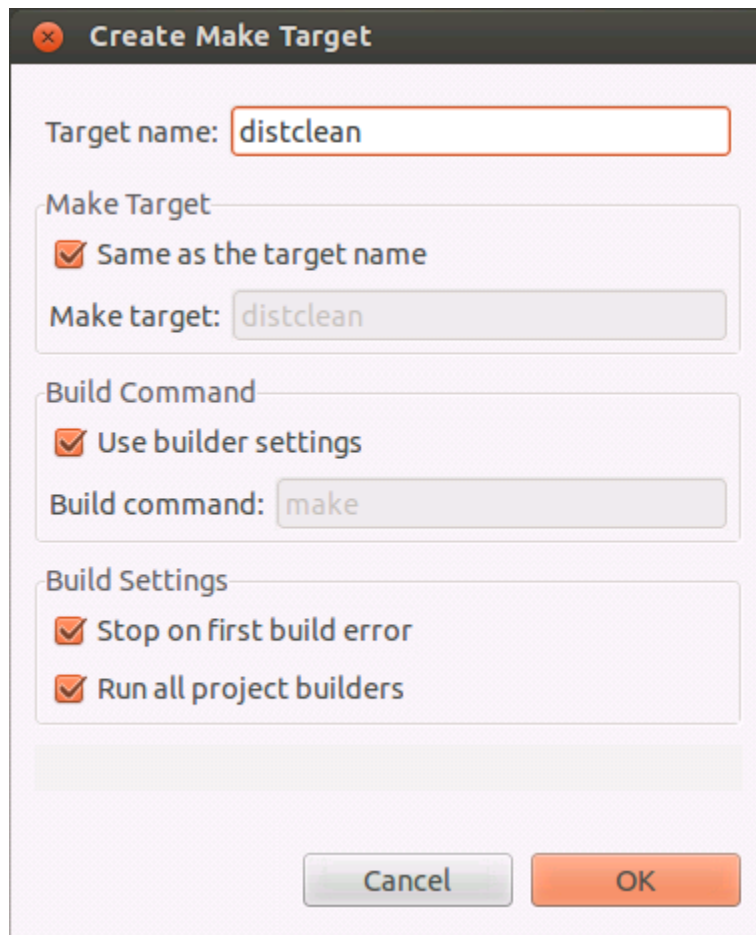
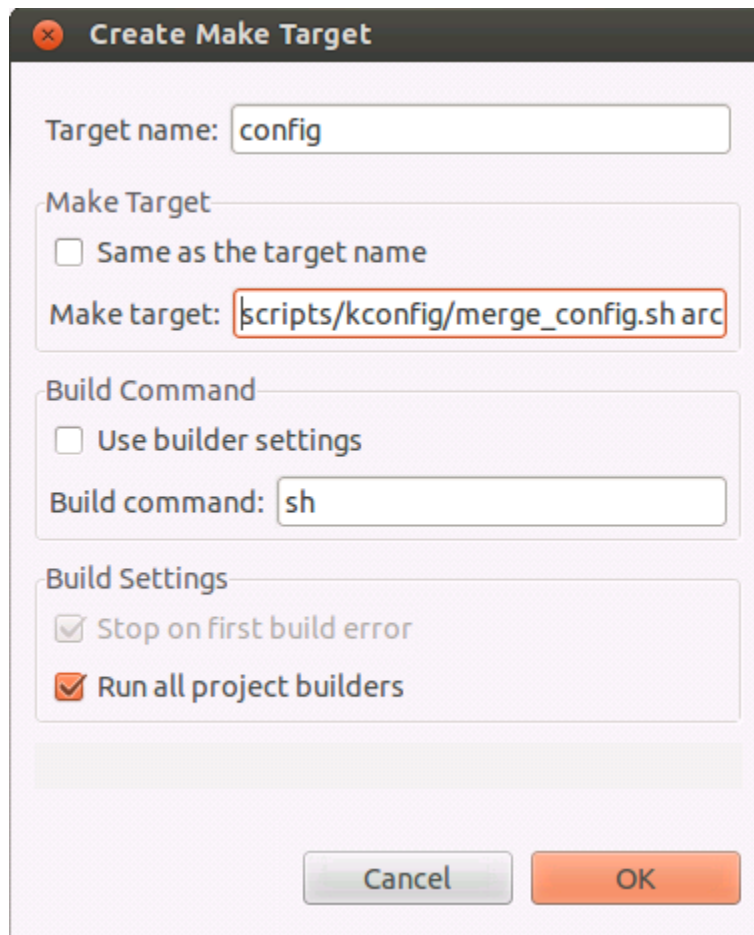


Figure 7.

## 5 Building Linux Kernel using CodeWarrior for ARMv8

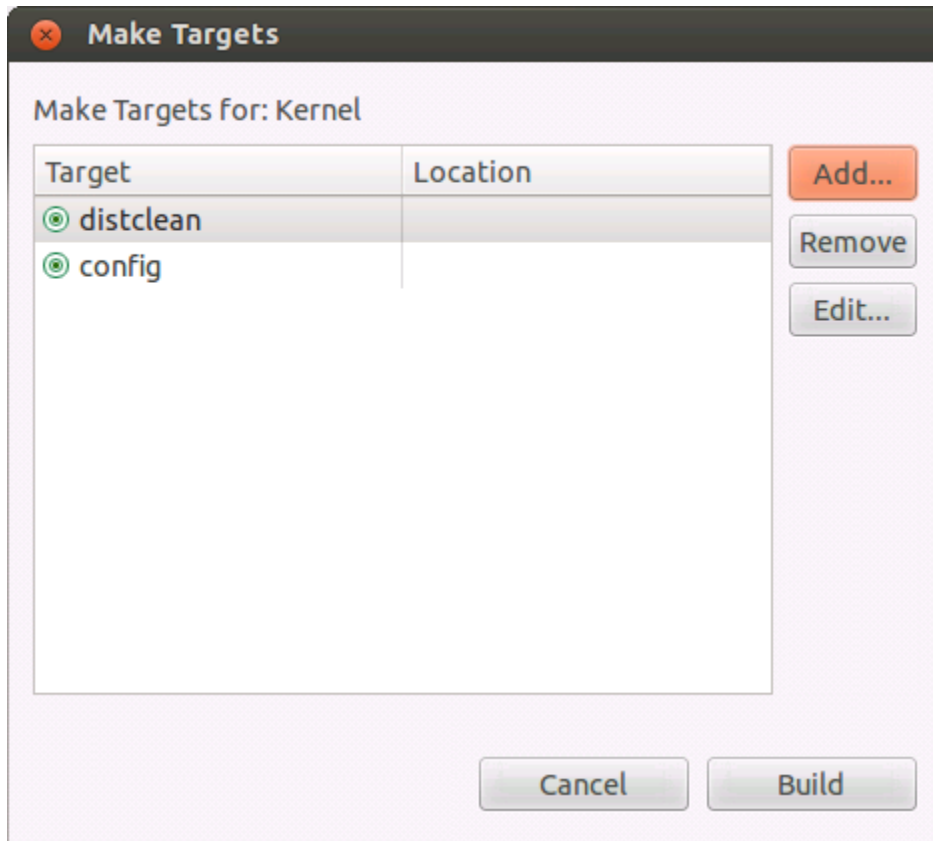
In order to build Linux Kernel using CodeWarrior for ARMv8, two build activities must be created under **Project > Make Target > Build** from the menu bar.





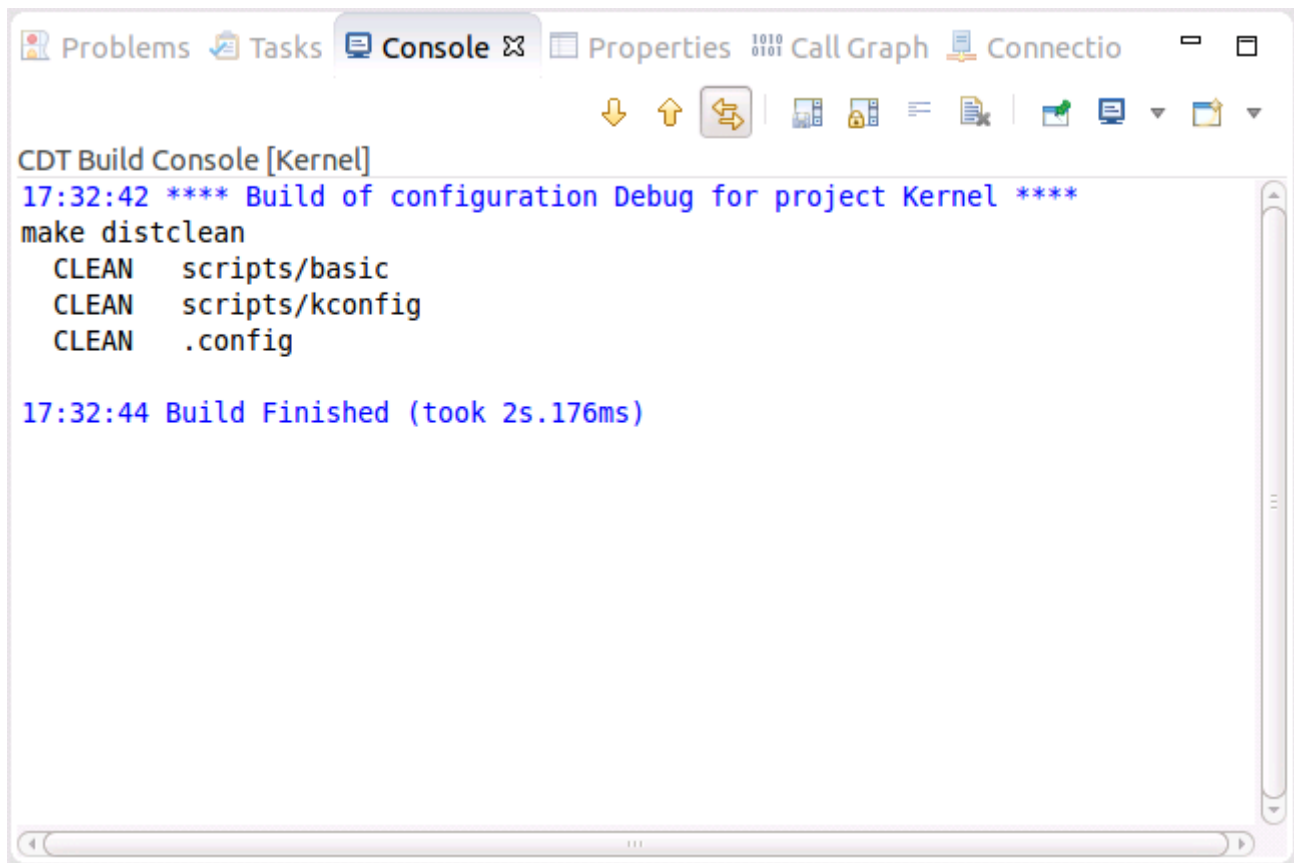
**Figure 8. Create Make Target**

Once configured we have two build targets.



**Figure 9. Make Targets**

Go to **Project > Make Target > Build**, select `distclean` and click **Build**. A “make distclean” command will run removing all the object and temporary files. Below message will be displayed when build is complete in the **Console** view.



**Figure 10. Console view**

Go again to **Project > Make Target > Build**, select config and click **Build**. A `sh scripts/kconfig/merge_config.sh arch/arm64/configs/defconfig arch/arm64/configs/freescale.config` command will run and configure the Linux Kernel to be built for LS2088ARDB board in this case.

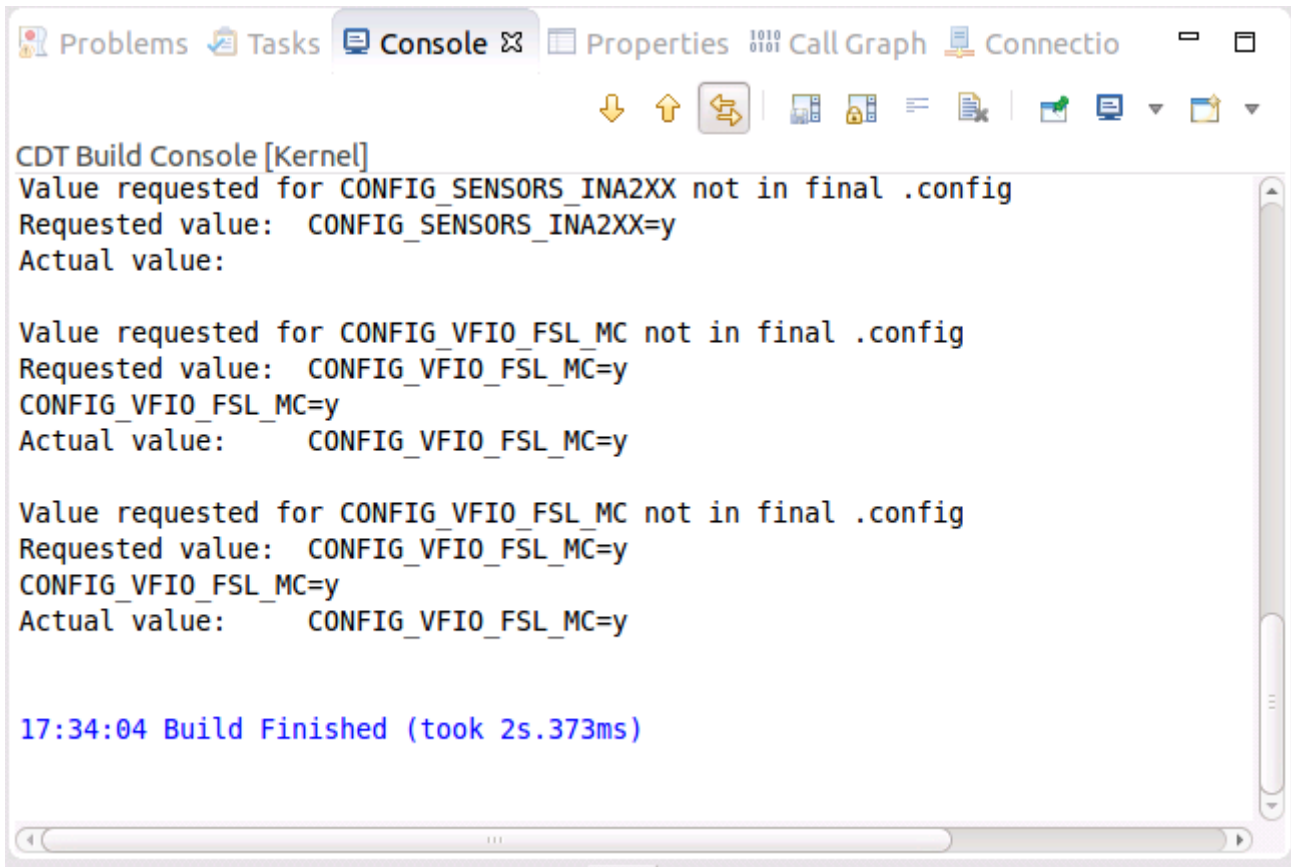
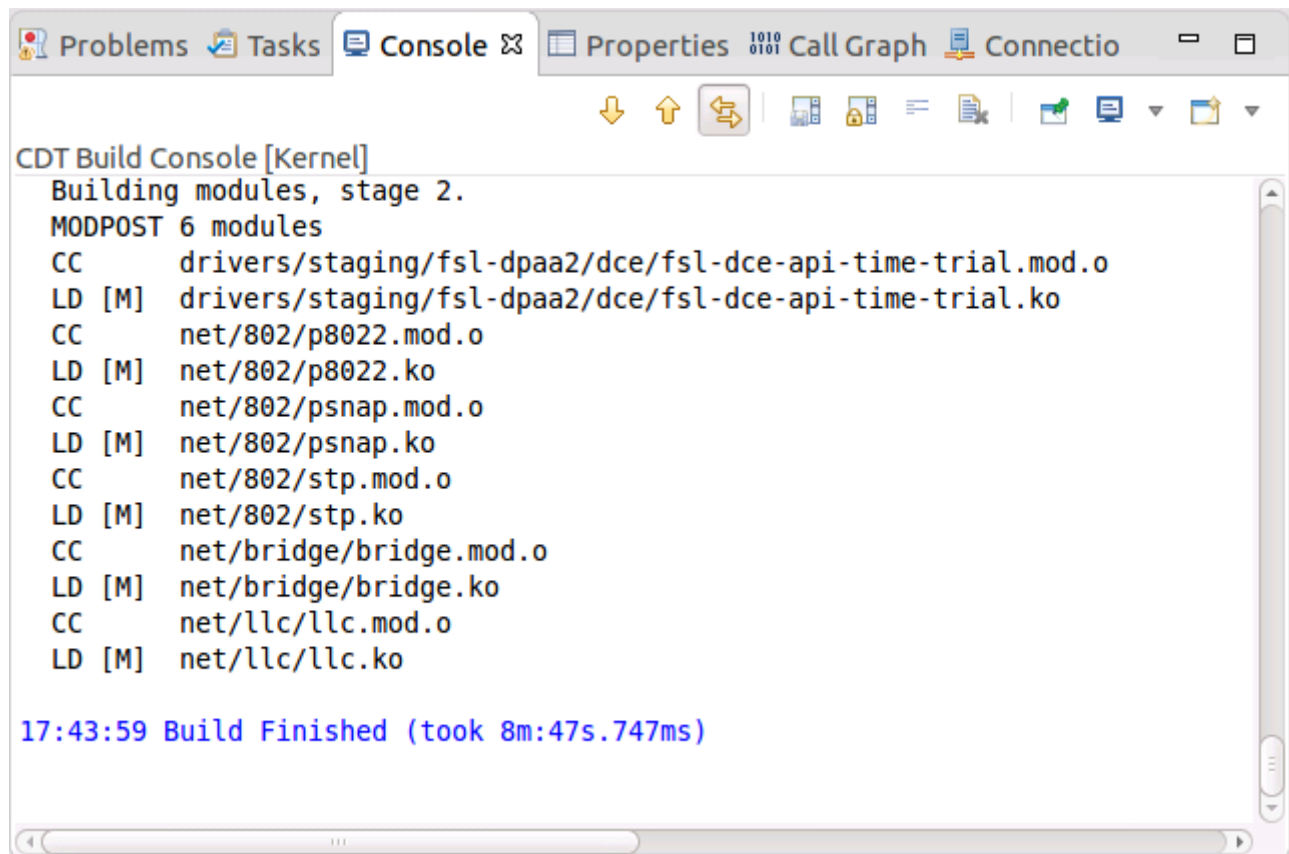


Figure 11. Console view

To build Linux Kernel, go to **Project > Build Project** from the menu bar. Below message will be displayed when build is complete in the **Console** view.



```
CDT Build Console [Kernel]
Building modules, stage 2.
MODPOST 6 modules
CC      drivers/staging/fsl-dpaa2/dce/fsl-dce-api-time-trial.mod.o
LD [M]  drivers/staging/fsl-dpaa2/dce/fsl-dce-api-time-trial.ko
CC      net/802/p8022.mod.o
LD [M]  net/802/p8022.ko
CC      net/802/psnap.mod.o
LD [M]  net/802/psnap.ko
CC      net/802/stp.mod.o
LD [M]  net/802/stp.ko
CC      net/bridge/bridge.mod.o
LD [M]  net/bridge/bridge.ko
CC      net/llc/llc.mod.o
LD [M]  net/llc/llc.ko

17:43:59 Build Finished (took 8m:47s.747ms)
```

Figure 12. Console view

**How to Reach Us:**

**Home Page:**

[nxp.com](http://nxp.com)

**Web Support:**

[nxp.com/support](http://nxp.com/support)

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any 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: [nxp.com/SalesTermsandConditions](http://nxp.com/SalesTermsandConditions).

Freescale, the Freescale logo, CodeWarrior, and QorIQ are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. ARM, Cortex, Cortex-A53, Cortex-A57, and TrustZone are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.

© 2016, Freescale Semiconductor, Inc.

Document Number AN5351  
Revision 0, 10/2016

