

Building U-Boot in CodeWarrior ARMv8

1. Introduction

This application note defines guidelines for configuring CodeWarrior for ARMv8 for U-Boot development.

This document explains:

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

Contents

1. Introduction.....	1
2. Preliminary background.....	2
3. Changes in CodeWarrior ARMv8 stationery project.....	2
4. Debugging.....	6

2. Requirements

For building U-Boot using CodeWarrior for ARMv8, is necessary a host computer with Linux OS and CodeWarrior for ARMv8 Linux version installed.



3. Installing SDK standalone toolchain

Linux SDK provides a standalone toolchain which 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:

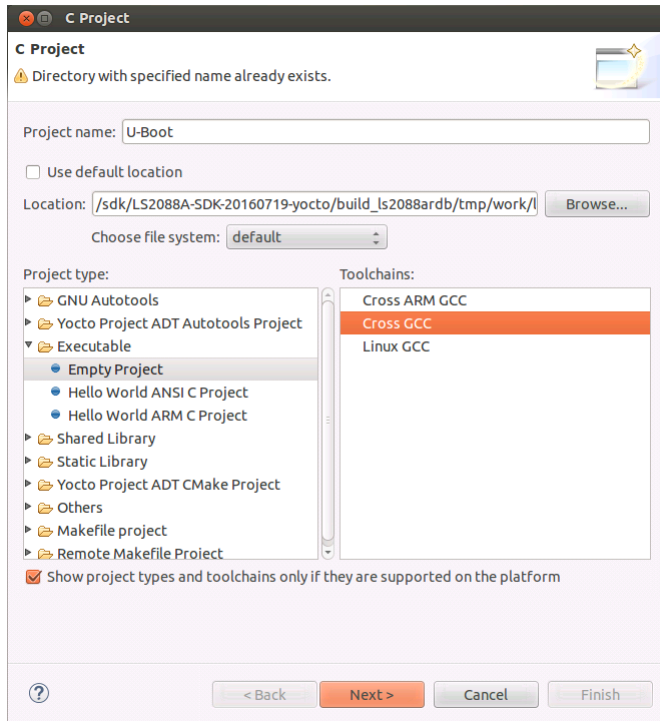
```
$ cd build_<machine>_release
$ bitbake fsl-toolchain
$ cd build_<machine>_release/tmp/deplo/sdk
$ ./fsl-qoriq-glibc-<host-system>-<core>-toolchain-<release>.sh
```

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 [SDK Knowledge Center](#).

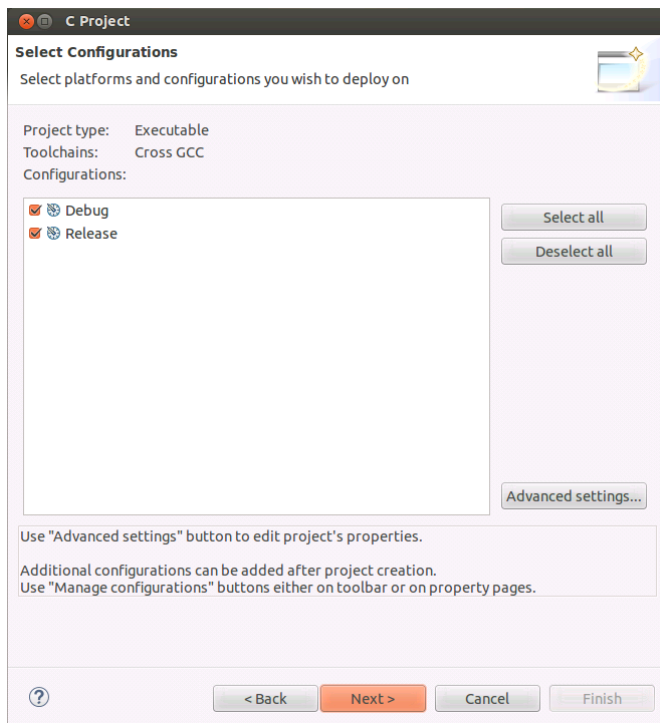
4. Configuring CodeWarrior for ARMv8 for building U-Boot

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

1. Choose **File > New > C Project**
2. Specify the project name and select Empty Project as Project type
3. Uncheck the **Use default location** and use the Browse button to find the location for U-Boot source
4. Chose Cross GCC as Toolchain
5. Click Next

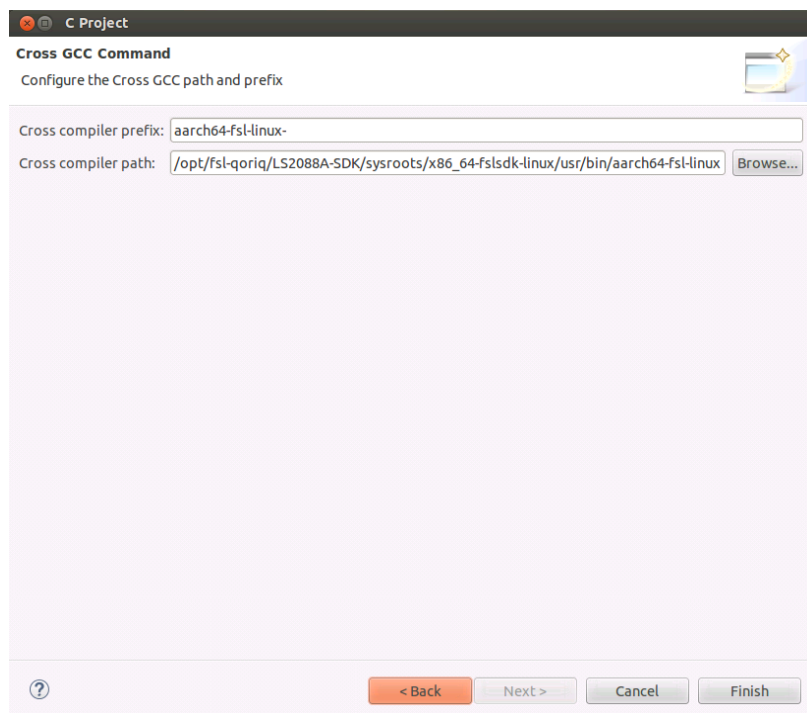


6. Choose both Debug and Release configurations and click Next

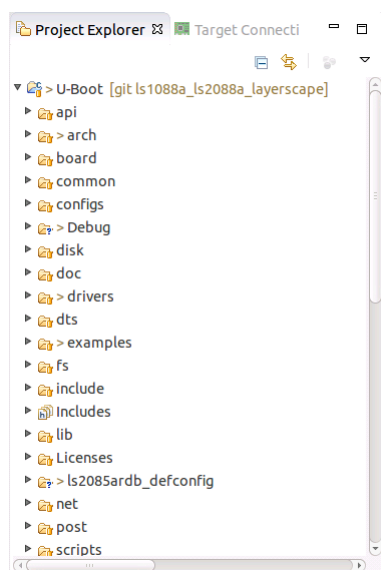


7. Specify the Cross compiler prefix, Cross compiler path and click Finish

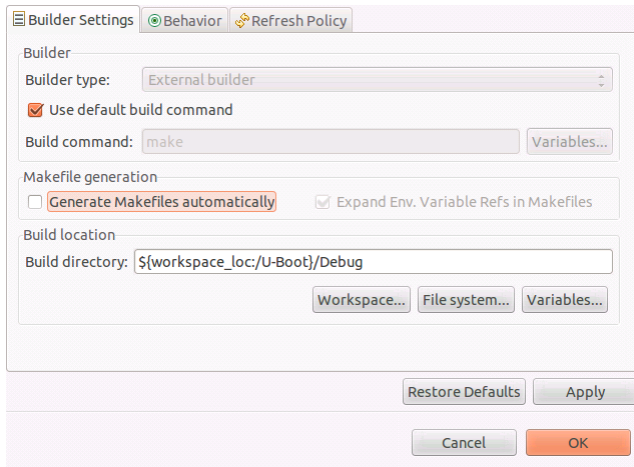
Configuring CodeWarrior for ARMv8 for building U-Boot



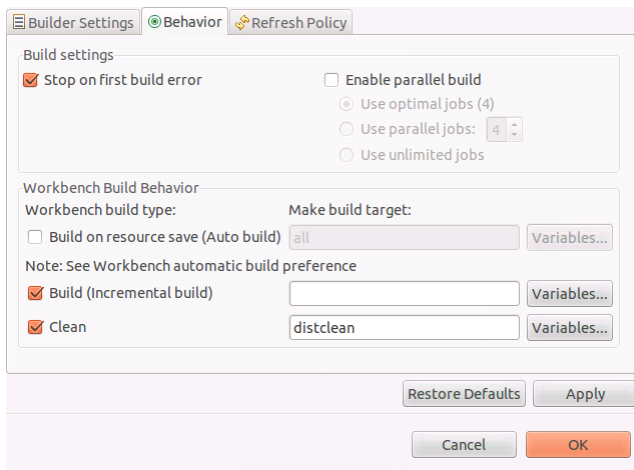
8. Project is created and will appear in Project Explorer view



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



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



12. 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: **SDKTARGETSYSROOT**
 Value: **/opt/fsl-qoriq/LS2088A-SDK/sysroots/aarch64-fsl-linux**
 Click **Add to all configuration**

NOTE SDK toolchain is a sysrooted toolchain. This means that GCC will start to look for target fragments and libraries starting from the path specified by the sysroot option.

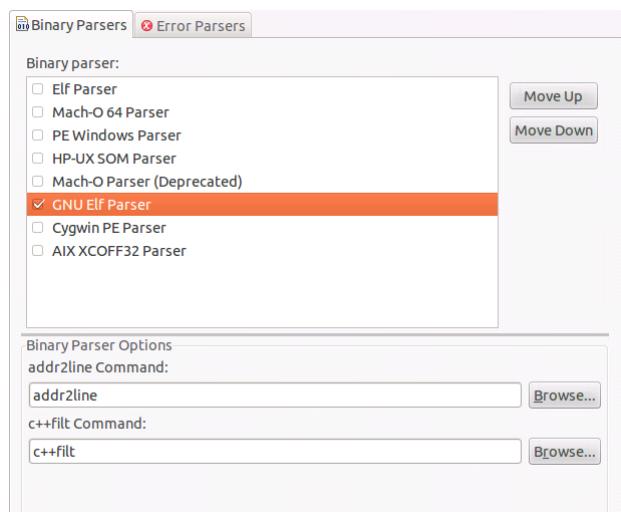
Name: **PATH**

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:/bin**
Click **Add to all configuration**

Name: **KCFLAGS**

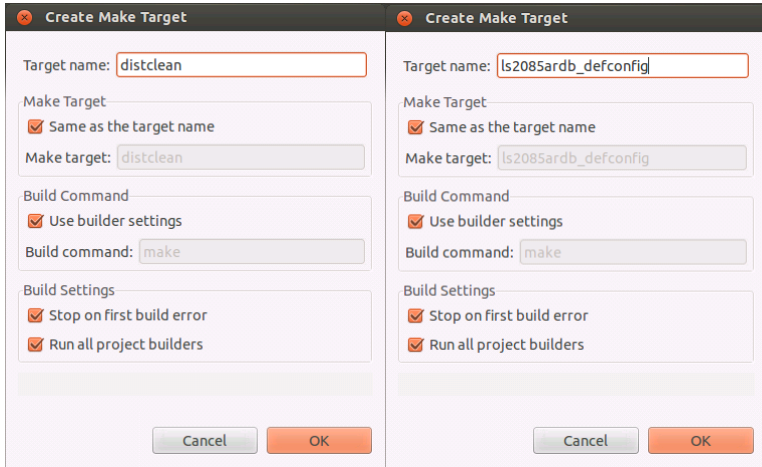
Value: **“--sysroot=\${ SDKTARGETSYSROOT }”**
Click **Add to all configuration**

13. Go to **Project > Properties > C/C++ build > Settings** and uncheck **Elf Parser** and check on **GNU Elf Parser**

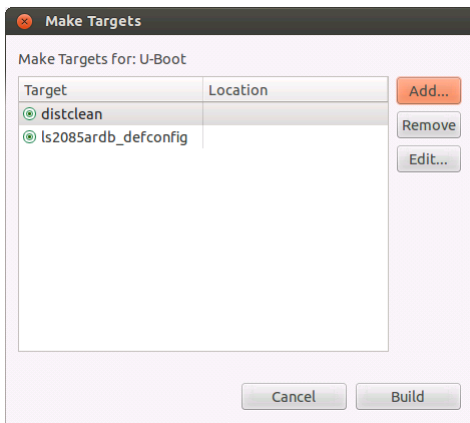


5. Building U-boot using CodeWarrior for ARMv8

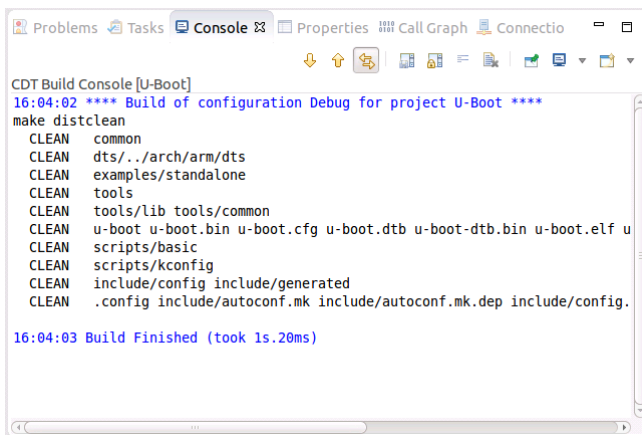
In order to build U-Boot using CodeWarrior for ARMv8, two build activities must be created under **Project > Make Target > Build** from the menu bar.



Once configured we have two build targets.

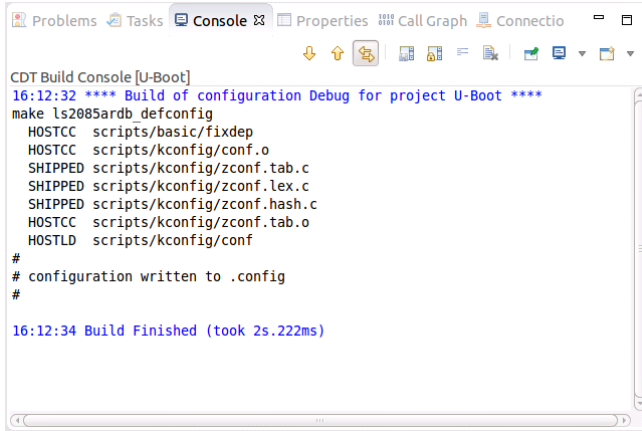


Go to **Project > Make Target > Build**, select **distclean** and click **Build**. A “make distclean” command will run removing all the object and temporary files. Below message will be displayed when build is complete in **Console** view.



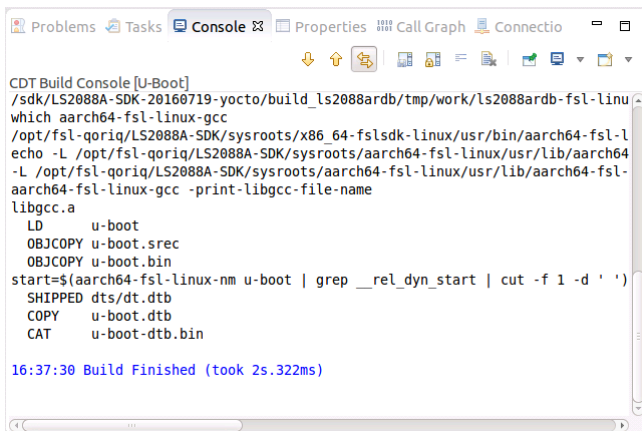
Building U-boot using CodeWarrior for ARMv8

Go again to **Project > Make Target > Build**, select **ls2085ardb_defconfig** and click **Build**. A "make ls2085ardb_defconfig" command will run and configure the U-Boot to be built for LS2088ARDB board in this case.



```
CDT Build Console [U-Boot]
16:12:32 **** Build of configuration Debug for project U-Boot ****
make ls2085ardb_defconfig
HOSTCC scripts/basic/fixdep
HOSTCC scripts/kconfig/conf.o
SHIPPED scripts/kconfig/zconf.tab.c
SHIPPED scripts/kconfig/zconf.lex.c
SHIPPED scripts/kconfig/zconf.hash.c
HOSTCC scripts/kconfig/zconf.tab.o
HOSTLD scripts/kconfig/conf
#
# configuration written to .config
#
16:12:34 Build Finished (took 2s.222ms)
```

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



```
CDT Build Console [U-Boot]
/sdk/LS2088A-SDK-20160719-yocto/build_ls2088ardb/tmp/work/ls2088ardb-fsl-linu
which aarch64-fsl-linux-gcc
/opt/fsl-qorIQ/LS2088A-SDK/sysroots/x86_64-fslsdk-linux/usr/bin/aarch64-fsl-l
echo -L /opt/fsl-qorIQ/LS2088A-SDK/sysroots/aarch64-fsl-linux/usr/lib/aarch64
-L /opt/fsl-qorIQ/LS2088A-SDK/sysroots/aarch64-fsl-linux/usr/lib/aarch64-fsl
aarch64-fsl-linux-gcc -print-libgcc-file-name
libgcc.a
LD u-boot
OBJCOPY u-boot.srec
OBJCOPY u-boot.bin
start=$(aarch64-fsl-linux-nm u-boot | grep __rel_dyn_start | cut -f 1 -d ' ')
SHIPPED dts/dt.dtb
COPY u-boot.dtb
CAT u-boot-dtb.bin
16:17:30 Build Finished (took 2s.322ms)
```


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 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.

Freescale, the Freescale logo, CodeWarrior, QorIQ, and Processor Expert 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.

© 2016 Freescale Semiconductor, Inc.

