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.
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:

```bash
$ 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**.
6. Select both **Debug** and **Release** configurations and click **Next**.
7. Specify the Cross compiler prefix, Cross compiler path and click Finish.
8. Project is created and will appear in the Project Explorer view.
9. Go to Project > Properties > C/C++ build, select Builder settings and uncheck Generate Makefiles automatically.
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.
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: 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.

13. Go to Project > Properties > C/C++ build > Settings and uncheck Elf Parser and select GNU Elf Parser.
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.
Building Linux Kernel using CodeWarrior for ARMv8

Create Make Target

Target name: distclean

Make Target
- Same as the target name
- Make target: distclean

Build Command
- Use builder settings
- Build command: make

Build Settings
- Stop on first build error
- Run all project builders
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 the Console view.

Figure 9. Make Targets
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.
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.

**Figure 11. Console view**
Figure 12. Console view