NXP i.MXRT685 DSP User Guide

Introduction

This document provides an overall introduction to the DSP including system architecture, file organization, DSP-related toolchain, and so on. This document helps with the overall understanding of the DSP-related code.

File Organization

The DSP audio framework (XAF) and example applications source codes and build projects are contained inside the MCUXpresso SDK release package. Please see the MCUXpresso SDK Release Notes for an overview of the SDK package contents and locations.

This table provides an overview of the DSP-specific package contents:

<install_dir>/devices/MIMXRT685S/ - Unified device code, including peripheral driver source that can be compiled for ARM or DSP core.

The following drivers have been tested for execution on the DSP:

- I2C (fsl_i2c.c, fsl_i2c_dma.c)
- I2S (fsl_i2s.c, fsl_i2s_dma.c)
- DMIC (fsl_dmic.c, fsl_dmic_dma.c)
- DMA (fsl_dma.c)
- USART (fsl_usart.c)

<install_dir>/middleware/multicore/rpmsg_lite - Unified RPMsg Lite, with platform porting layer for ARM and DSP core (lib/rpmsg_lite/porting/platform/).

<install_dir>/boards/evkmimxrt685/dsp_examples - DSP example applications:

- hello_world - DSP-only example that prints 'Hello World' on the DSP debugger.
- hello_world_usart - Example that prints 'Hello World' to the UART using both cores.
- xaf_demo - Example that utilizes the XAF DSP audio framework to execute several components demonstrations. This example also demonstrates the DMIC, I2S, and I2C drivers operating on the DSP.

<install_dir>/middleware/audio_framework/ - DSP audio framework.

<install_dir>/middleware/audio_framework/hifi4_xaf/ - Source code and documentation for the core XAF framework.

<install_dir>/middleware/audio_framework/hifi4_app/ - Utilities and unit tests for developing applications on top of the XAF framework.

<install_dir>/middleware/audio_framework/hifi4_app/plugins/ - XAF component wrappers and codec binaries for MP3, AAC, SRC, PCM gain, capturer, and renderer.

<install_dir>/middleware/naturedsp_hifi4/ - NatureDSP library for HiFi4 DSP on i.MXRT685.

Building DSP applications on Linux

Before you compile the DSP-related code, set up the DSP-related toolchains. The DSP framework, DSP codec wrapper, and DSP codec use Xtensa development toolchain.

Toolchain Files

The Xtensa development toolchain consists of three components which are installed separately in the Linux OS, including:

- Configuration-independent Xtensa Tool
- Configuration-specific core files and Xtensa Tool
- Memory map linker files (Xtensa LSP) for i.MXRT685 HiFi4

The configuration-independent Xtensa Tool is released by Cadence. For the current code, the version of this tool is XtensaTools_RG-2017.8_linux.tgz. This toolchain package can be downloaded from the Xplorer IDE directly from Cadence, and it will also be distributed by NXP.

The configuration-specific core files and the Xtensa Tool are released only by NXP. The current version of this tool is nxp_rt600_ext_dual_linux.tgz.

The LSP package for the Xtensa toolchain is release only by NXP. This SRAM version of this is nxp_rt600_sram_lsp.tgz.

Linux Host OS Setup

The Xtensa Tools and additional tools are provided as 32-bit (x86) binaries. They are supported on 32-bit (x86) systems, and also on recent 64-bit (x86-64) systems that have appropriate 32-bit compatibility packages installed. If you use a 64-bit system (for example; Ubuntu 16.04), install the 32-bit compatibility packages first. Use these commands:

```
sudo apt-get install lib32ncurses5 lib32z1
sudo dpkg --add-architecture i386
sudo apt-get install libc6:i386 libstdc++6:i386
```

Toolchain Installation

When you have these three components, you can set up the toolchain as follows:

- Open a shell and execute the following commands:

```
mkdir -p ~/xtensa/tools
mkdir -p ~/xtensa/builds
mkdir -p ~/xtensa/lsp
```

- Set up the configuration-independent Xtensa Tool:

```
cd ~/xtensa
tar zxvf XtensaTools_RG_2017_8_linux.tgz -C ./tools
```

- Set up the configuration-specific core files:

```
cd ~/xtensa
tar zxvf nxp_rt600_ext_dual_linux.tgz -C ./builds
```

- Copy the configurable LSP memory map files to this path:

```
cd ~/xtensa
tar zxvf nxp_rt600_sram_lsp.tgz -C ./lsp
```

- Install the Xtensa development toolchain:

```
cd ~/xtensa
./builds/RG-2017.8-linux/nxp_rt600_ext_dual/install --xtensa-tools ./tools/RG-2017.8-linux/XtensaTools --registry ./tools/RG-2017.8-linux/XtensaTools/config
```

- Set environment variables:

```
export PATH=~/xtensa/tools/RG-2017.8-linux/XtensaTools/bin:$PATH
export XTENSA_SYSTEM=~/xtensa/builds/RG-2017.8-linux/nxp_rt600_ext_dual/config
export XTENSA_LSP=~/xtensa/lsp/nxp_rt600_sram
export XCC_DIR=~/xtensa/tools/RG-2017.8-linux/XtensaTools
```

- Set the LM_LICENSE_FILE environment variable.

The Xtensa development tools use FLEXlm for license management. The FLEXlm licensing is required for tools such as the Xtensa Xplorer, TIE Compiler, and Xtensa C and C++ compiler. If you want to use a floating license, install the FLEXlm license manager and set the LM_LICENSE_FILE environment variable. In case of any problems, you can find useful information in the Xtensa Development Tools Installation Guide User's Guide.doc document provided by Cadence.

After the above steps, the Xtensa development toolchain is set up successfully.

Compile example applications

After installing the the DSP-related toolchains on Linux OS, you can build example applications that utilize the MCUXpresso SDK and XAF audio framework. Linux DSP examples in the MCUXpresso SDK use CMake and GNU make.

Most DSP examples have independent ARM core and DSP core source and makefiles. Navigate to <install_dir>/boards/evkmimxrt685/dsp_examples/ and you will see the following structure:

hello_world_usart/cm33/ - Source and build for Cortex-M33 ARM core
hello_world_usart/hifi4/ - Source and buld for HiFi4 DSP

Under <example>/hifi4/xcc/ is the CMakeLists.txt build file, and convenience shell scripts to build the DSP image. Execute the build_debug.sh script to compile the example application. Upon successfully building, the elf binary will be located in the debug/ directory, and two stripped DSP binaries (one for TCM and one for SRAM memories) will be placed under <example>/cm33/. The elf binary can be debugged directly, and the .bin files can be loaded by the ARM demo to execute on startup.

If you would like to see what commands the cmake scripts are exeucting to build the DSP (or ARM core) binaries, add the following to the top of the CMakeLists.txt file:

```
set(CMAKE_VERBOSE_MAKEFILE ON)
```

Debugging example applications

After generating an elf DSP application binary, this can be debugged using the Xtensa GDB tool under Linux.

In order to use xt-gdb, you will need to run the xt-ocd on-chip debugger. Please refer to the Cadence 'Xtensa Debug Guide' for more information on setting up the OCD daemon and configuring it with your debugger.

Building DSP applications on Windows

The DSP framework can be built also on Windows OS. The Xplorer software can be used to build the DSP framework on Windows. This chapter explains how to use Xplorer to build the DSP framework. It is assumed that you have installed the Xplorer software on your Windows system and you acquired the Xplorer licence from Cadence. The Xplorer 7.0.8 version is used as an example and its default installing folder is C:\usr\xtensa.

Add Configuration package

The nxp_rt600_ext_win32.tgz configuration package is used to build the DSP framework on Windows, so you need to add this configuration package into Xplorer before building the code. You can get this configuration package and the corresponding memory map LSP files from NXP. The required files are as following:

- nxp_rt600_ext_win32.tgz
- nxp_rt600_sram_lsp.tgz

When you have the DSP configuration package and memory map linker files, you can add a new configuration package into Xplorer as follows:

- Download and install Xtensa Tools for Xplorer.

If you do not have the Xtensa Tools, you shall download and install it using Xplorer. Currently, the Xtensa Tool that we use is XtensaTools_RG_2017_8_win32.tgz. You can first open the Xplorer software and click the “RG_2017.8” option in the “XPG View” panel and select the “tools->Xtensa Tools->Xtensa Tools 12.0.8 for Windows” option. After you select it, you can click the download button to start the downloading process.

After the download finishes, right click the “Xtensa Tools 12.0.8 for Windows” option and select the “Install Xtensa Tools…” option in the new dialog. The installing process takes some time. The Xtensa Tool is installed successfully after this step. You can see this folder in the Xplorer’s installing folder if everything is ok:
C:\usr\xtensa\XtDevTools\install\tools\RG-2017.8-win32

- Add the configuration package into Xplorer.

When you have the nxp_rt600_ext_win32.tgz package from NXP, you can add it into Xplorer. The first thing to do is to create a new folder called build in Xplorer’s installing path if the build folder is not created already. The total path after this operation is as follows:

C:\usr\xtensa\XtDevTools\downloads\RG-2017.8\build

- Place the nxp_rt600_ext_win32.tgz package into the new build folder.

C:\usr\xtensa\XtDevTools\downloads\RG-2017.8\build\nxp_rt600_ext_win32.tgz

- After you have performed the above steps, you can click the refresh button in the “XPG View” panel and find the “build” option in this panel.
- Right click the build->nxp_rt600_ext_win32.tgz package and click the “Install Build…” option in the new dialog to start the installing process. This takes some time. You can see the following folder in the Xplorer’s installing folder if everything is OK.

C:\usr\xtensa\XtDevTools\install\builds\RG-2017.8-win32\nxp_rt600_ext_win32

- Add the new LSP files into Xplorer.

After you add the nxp_rt600_ext_win32.tgz configuration package into Xplorer, you can add the new memmap linker files. You can extract the LSP into the C:\usr\xtensa\ folder.

After you complete the above three steps, the new configuration package and the corresponding memory map linker files are successfully added into Xplorer.