NXP Semiconductors
Application Note

Document Number: AN5346
Rev. 0, 10/2016

Adapting CodeWarrior ARMv8 Stationery to Download and Run from OCRAM

Contents

1 Introduction

CodeWarrior projects provide an excellent test bed for developing and evaluating embedded target code. Standard CodeWarrior projects typically initialize and use target DDR to host the downloaded project code. However, there are some cases in which target evaluation or testing must be performed, but target DDR is not available.

This document explains:

- Method for downloading a CodeWarrior ARMv8 project into the target On-Chip RAM
- Running the target code in On-Chip RAM, without needing any other target memory

2 Preliminary background

OCRAM on QorIQ LS processors is 128 KB that is large enough to hold some embedded C or C++ code while still allowing for standard uses of RAM, such as variables, stack, and heap. But, care must be taken to ensure that test code intended to be hosted by and run from only OCRAM, not only fits within this space during downloads, but also has enough available space to manage the target code's run-time requirements such as variables, stack, and heap. Recursive functions are especially vulnerable to stack overflow in this limited RAM space.

The limited capacity of OCRAM on QorIQ LS processors prevents the use of large support libraries, such as *stdio.h*, which prevents use of convenient function as `printf()`. The user can still monitor data and variable values in the debugger Registers, Variables, and Memory views.

Debugging should be limited to use only one core to the stack and heap space needed for each core.

Trace data can be collected while debugging in OCRAM. Be sure to select the On-Chip buffer in the CodeWarrior ARMv8 Trace Commander view.

3 Download and run ARMv8 stationery project only from OCRAM

You need to perform following changes to a CodeWarrior ARMv8 C stationery project to download and run the project only from OCRAM:

- Source code changes
- Linker Control File changes
- Enable the use of OCRAM to execute code

3.1 Source code changes

`#include <stdio.h>` is included in three source files of every "`Hello World C`" project:

- `src\exceptions\exception.c`
- `src\gic\gic.c`
- `src\main.c`

Delete `#include <stdio.h>` and all references to printf(), from each of these three files and save the changes.

It is recommended (though not required) to edit `src\start.S` to delete the use of the Linker Control File variable `___DDR_ADDRESS`. This variable is used during the initialization of the target MMU. Leaving the variable and this portion of initialization code in place, will not affect the operation of the target code, but doing so makes code bad. If the reference to `___DDR_ADDRESS` is retained in `start.S`, make sure it is defined in the Linker Control file. If it is deleted, then also delete the block of code that uses it, in the function `_init_tables`.

3.2 Linker Control File changes

The Linker Control File (LCF) will also need editing. This section describes what linker definitions are required.

In the CodeWarrior project, expand the `Linker_Files` folder and edit the file `aarch64elf.x`. OCRAM on LS20xxA processors starts at address `0x1800_0000`, so for an LS20xxA target only, change the definition of:

```
PROVIDE (___OCRAM_ADDRESS = 0x10000000);
```

to

```
PROVIDE (___OCRAM_ADDRESS = 0x18000000);
```

NOTE

`PROVIDE (___OCRAM_ADDRESS = 0x10000000);` is valid for LS1043A (and derivatives) and LS1012A (and derivatives).

Next, change:

```
PROVIDE (___START_RAM_ADDRESS = ___DDR_ADDRESS + ___CORE_NUMBER * ___MEMORY_SIZE);
```

to

```
PROVIDE (___START_RAM_ADDRESS = ___OCRAM_ADDRESS + ___CORE_NUMBER * ___OCRAM_SIZE);
```

Next, change:

```
PROVIDE (___STACK_AND_HEAP_SIZE = 0x80000);
```

to

```
PROVIDE (___STACK_AND_HEAP_SIZE = 0x2000);
```

NOTE

The Linker variable `___STACK_SIZE_PER_CORE` is not used, so does not need to be reduced in defined size.

Finally, change:

```
PROVIDE(___END_RAM_ADDRESS_EXPECTED = ___START_RAM_ADDRESS + ___MEMORY_SIZE);
```

to

```
PROVIDE(___END_RAM_ADDRESS_EXPECTED = ___START_RAM_ADDRESS + ___OCRAM_SIZE);
```

The CodeWarrior build tools will warn with a “Not enough memory” error displayed in the Console view when building the CodeWarrior project, when code size is too large to fit in the available OCRAM space. If this happens, the solution is one or both of the following:

- Reduce code size
- Reduce `___STACK_AND_HEAP_SIZE` value

A convenient way to determine how much total RAM your code requires is to examine the contents of either the `.map` or `.lst` file after building your project. These files are located in project’s Debug folder, but they may not be generated if errors are generated. If none of these files are generated, edit the LCF and temporarily make `___OCRAM_SIZE` large enough to build without errors.

3.3 Enable the use of OCRAM to execute code

For LS20xxA projects, edit the *Target Initialization File* in the project’s **Target Connection Configuration**, to add these lines at the end of the file:

```
# Enable "honor_ewa_en" bit in "SA Auxiliary Control register" of the CCN-504.
CCSR_LE_M(0x04080500, 0x000008d7)
```

4 Debugging

Debugging the OCRAM-hosted application proceeds is similar to debugging DDR-hosted code. Configure the **Target Connection** definition as described in Section 5.1 “Target Connection configurator overview” of the QorIQ LS series - ARM V8 ISA, Targeting Manual, then start the debugger according to Section 2.5.1 “Debugging Bareboard project” in the Targeting manual.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number AN5346
Revision 0, 10/2016