恩智浦半导体。

文档编号：AN12270

应用笔记

第 0 版，2019 年 1 月

使用 CCS 脚本和 DRAM 启动实现 Layerscape 器件启动

1. 简介

打开电源后，Layerscape (LS)器件将开始启动过程。它从由配置引脚 `cfg_rcw_src` 指定的启动源中获取 RCW（复位配置字）。如果存储启动映像的闪存为空，则 RCW 无效。LS 器件的启动逻辑首先检查 RCW 的格式；当它查找不到有效标头时，启动过程将停止。

如果启动源为可移动的（例如 SD 卡），则可以使用 PC 或其他板对 SD 卡进行烧写。

对于启动闪存（例如通常焊接在板上的 SPI 闪存/NOR 闪存/NAND 闪存/eMMC），烧写启动映像的两种常用方法如下：

1. 如果可能，请按照参考手册中的说明修改 `cfg_rcw_src` 引脚短接设置，来选择有效的硬编码 RCW 选项。这将使器件处于正常工作状态。然后，使用由用户选择的闪存烧写工具对闪存进行烧写。
2. 使用恩智浦 CodeWarrior 闪存烧写工具。CodeWarrior 工具实现方法在闪存为空时也能够覆盖 RCW。CW 无需依赖开关，在器件中强制执行硬编码 RCW，并对闪存器件进行烧写。

第三种不常用的闪存烧写方法如下：

3. 使用 JTAG/BSDL 闪存烧写工具，例如 Asset。

本应用笔记介绍了启动 Layerscape 板的一种全新方法。

目录

2. 方法说明

注：用户必须确保 RESET_REQ 信号不会产生 HRESET_B 或 PORESET_B 信号，这种方法才会行之有效。

CodeWarrior 配备了一个低级脚本工具 CCS（CodeWarrior 通信服务器）。CCS 使用命令以及 TCL 脚本中的一系列命令来控制板并与之通信。

为使启动过程简化和自动执行，开发了一个名为 lsbp.tcl 的 TCL 脚本。附录 A 提供了一个完整的脚本，以供阅读和参照。只需对该脚本进行一些自定义，即可启动新板的启动过程。

此方法需要通过 JTAG 将 U-Boot 映像加载到 DDR 中，因此需要正常工作的 DDR。

lsbp.tcl 脚本调用 DDR 初始化脚本，该脚本用于配置和初始化 DDR 控制器寄存器。

注：用户应利用 QCVS DDR 工具来获得板正确的 DDR 控制器寄存器值。lsbp.tcl 中使用的特定硬代码 RCW 具有固定的 DDR 时钟比率，该比率不能被覆盖/更改。请确保在 QCVS 工具中使用正确的 DDR 速率来生成正常工作的 DDR 寄存器设置。

如果 DDR 正常工作并且启动映像能够可靠地加载到 DDR，则通常可以在控制台端口上查看一些输出。用户应继续调试 U-Boot，直到它成功启动，出现命令提示符。

以下是启动新板的步骤：

1) 自定义 lsbp.tcl。

   有关更多信息，请参见第 3 节“CCS 脚本的说明”。

2) 自定义 DDR init 文件。

   在附录 A中，针对每个支持器件提供了一个可用作模板的示例 DDR init 文件。用户需要根据板的 DDR 设计修改 DDR 寄存器设置。

3) 安装 CodeWarrior。

   从恩智浦网站下载*用于 QorIQ LS 系列 Armv8 ISA 的 CodeWarrior*。

   查看 CCS 可执行文件，可访问

   {installation directory\Common\CCS\bin}.

   例如，

   C:\Freescale\CW4NET_v2018.01\Common\CCS\bin

4) 将所有 TCL 脚本和 u-boot.bin 复制到 CCS 目录下。

5) 启动 CCS。

   双击 ccs.exe。

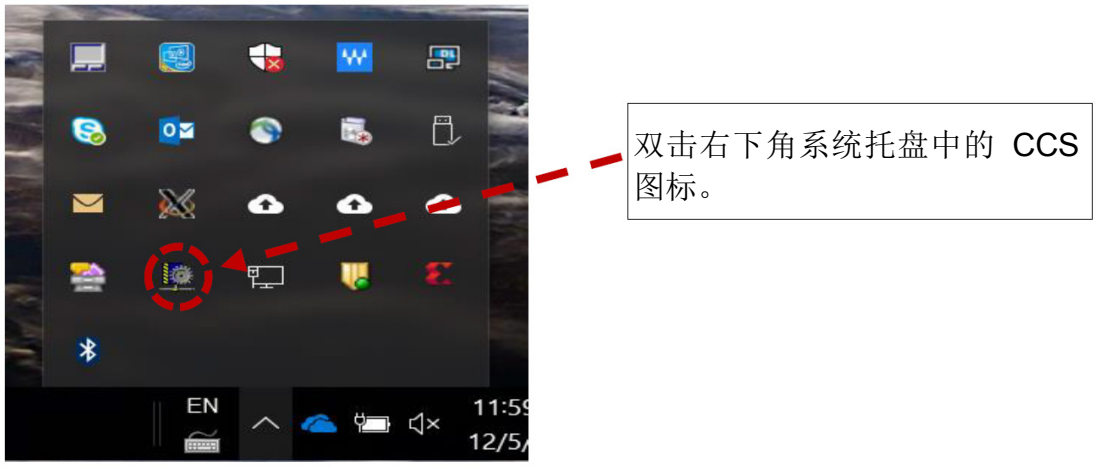

图 1：系统托盘中的 CCS 图标

6) 在 CCS 窗口中，运行 `lspb.tcl` 脚本。

```
(bin) 64 % source lsbp.tcl
(bin) 64 % lsbp::lsbp
Chain Position 0: LS1043A
Chain Position 1: CoreSight ATB Funnel
Chain Position 2: CoreSight ATB Funnel
Chain Position 3: CoreSight TMC
Chain Position 4: CoreSight TMC
Chain Position 5: CoreSight ATB Funnel
Chain Position 6: CoreSight STM
Chain Position 7: CoreSight TMC
Chain Position 8: CoreSight ATB Funnel
Chain Position 9: CoreSight ATB Funnel
Chain Position 10: CoreSight TMC
Chain Position 11: CoreSight TMC
Chain Position 12: CoreSight TMC
Chain Position 13: CoreSight CTI
Chain Position 14: CoreSight CTI
Chain Position 15: CoreSight CTI
Chain Position 16: Cortex-A53
Chain Position 17: CoreSight CTI
Chain Position 18: Cortex-A53 PMU
Chain Position 19: Cortex-A53 ETM
Chain Position 20: Cortex-A53
Chain Position 21: CoreSight CTI
Chain Position 22: Cortex-A53 PMU
Chain Position 23: Cortex-A53 ETM
Chain Position 24: Cortex-A53
Chain Position 25: CoreSight CTI
Chain Position 26: Cortex-A53 PMU
Chain Position 27: Cortex-A53 ETM
Chain Position 28: Cortex-A53
Chain Position 29: CoreSight CTI
Chain Position 30: Cortex-A53 PMU
Chain Position 31: Cortex-A53 ETM
Chain Position 32: DAP
Chain Position 33: SAP2
(bin) 65 %
```

图 2：运行 lsbp.tcl 脚本后的输出

一旦从 DRAM 成功启动 U-Boot，就可以使用 U-Boot 命令将闪存映像烧写至闪存。这样，板就可以从闪存启动，无需 JTAG 的参与。在板启动开始阶段，网络接口可能尚未正常工作。因此，不能使用 TFTP 将 U-boot 闪存映像下载到板子的 U-boot 下。在这种情况下，可以使

用以下命令通过控制台端口下载映像：

```
loadb 80000000
TeraTerm  File -> Transfer -> Kermit -> Send
```

然后，使用浏览选择映像文件。

下载映像后，使用以下标准 U-Boot 命令将 U-boot 烧写到闪存中：

NOR：

```
prot off all
erase 60100000 +100000
cp.b 80000000 60100000 100000
```

loadb 命令会输出映像文件的实际大小。使用实际文件大小或 100000。

100000 比映像文件的大小和可使用的安全值略大一些。

这只会烧写 U-Boot 映像。用户还需要使用 QCVS 工具创建 rcw.bin 并使用以下命令将其烧写到位置 0x60000000：

```
cp.b 80000000 60000000 100
```

RCW 的大小为 100。除非预启动初始化(PBI)命令很多，否则对于 RCW 来说，0x100 是一个合适的大小。

NAND：

```
nand erase 0 100000
nand write 80000000 0 100000
```

NAND 映像随 RCW 一起烧写。

SD/eMMC：

```
mmc write 80000000 8 800
```

QSPI：

```
sf probe
sf erase 100000 +100000
sf write 80000000 100000 100000
```

第一个 100000 是 U-Boot 闪存中的偏移。第二个 100000 是大小。与 NOR 闪存类似，QSPI 闪存也需要使用 QCVS 工具创建 rcw.bin 并使用以下命令将其烧写到偏移 0：

```
sf erase 0 +100
sf write 80000000 0 100
```

从闪存成功启动 U-Boot 后，通过 CCS 脚本启动仍然具备优势。相比每次更改软件后对闪存进行映像擦除/烧写，通过 JTAG 测试软件更改可能更快且更方便。

3. CCS 脚本的说明

lsbp.tcl 是自动执行以下步骤的主脚本：

- RCW 优先
- JTAG 解析以查找 Boot Core/SAP2/DAP 链位置
- PBI 初始化所需最少量的寄存器
- 配置 DDR
- 将启动加载程序(U-boot)映像加载到 DDR 中
- 释放 ARM 内核以进行启动。

lsbp.tcl 脚本使用以下两个文件：

- ddr-init-lsxxxxxxx.tcl 初始化 DDR 控制器
- u-boot.bin 是 U-boot DRAM 映像

用户可以使用其他名称，只需脚本中的文件名与实际文件名匹配即可。以下是脚本的主体部分：

lsbp.tcl

```
#1: Customize the ddr init script file name

source ddr-init-ls10xxxxx.tcl

#2: Set device, currently supports ls1021a/ls1043a/ls1046a
set device ls1043a

switch -exact $device {
    "ls1021a" { set dut ls1020a }
    "ls1043a" { set dut ls1043a }
    "ls1046a" { set dut ls1043a }
    default {echo This device is not supported; return 0}
}

#3: Set hardcode_rcw
# For LS1043A and LS1046A
#   0x9E: DDRCLK is the reference clock for DDR
#   0x9F: DIFF_SYSCLK is the reference clock for DDR
# For LS1021A
#   0x9B: First try value. See RM for more options
set hardcode_rcw 0x9E




#4: Customize the CWTAP setting, see the lsbp.tcl script in Appendix for
more info
```

```
delete all
config cc cwtap:fsl023b3a
ccs::config_chain "$dut dap sap2"

# Call rcw-override procedure
lsbp::rcw-override $dut $hardcode_rcw

# Parse the JTAG to find out the chain position of DAP, SAP2, Boot Core
set tap [lsbp::tap-parse]
set dap [lindex $tap 0]
set sap2 [lindex $tap 1]
set boot_core [lindex $tap 2]

# Call the pbi procedure to write minimal registers
lsbp::pbi $dut $dap

# Call init-ddrc procedure to initialize DDR controller
lsbp::init-ddrc $dap

# Call loadb procedure to load u-boot.bin to DDR address 0x82000000
lsbp::loadb $sap2

# Release the boot core, ARM core 0
ccs::run_core $boot_core
}
```

lsbp.tcl 中需要自定义的部分如上述突出显示。关于第 3 项和第 4 项的其他几点注释：

- **#3 设置 hardcode_rcw**
  - 对于 LS1043A 和 LS1046A，该选择取决于 DDR 参考时钟是 DDRCLK 还是 DIFF_SYSCLK。
  - 对于 LS1021A，则有许多可用选项。请参见 RM。

- **#4 自定义 CWTAP 设置：**
  - 若使用 USB 连接，请使用：

```
config cc cwtap
```

  - 若使用以太网连接，请使用：

```
config cc cwtap:ipaddr
```

或

```
config cc cwtap:fslxxxxxx
```

其中数字 xxxxxx 是 MAC 地址的最后六位。在 CWTAP 的标签上可找到该值。

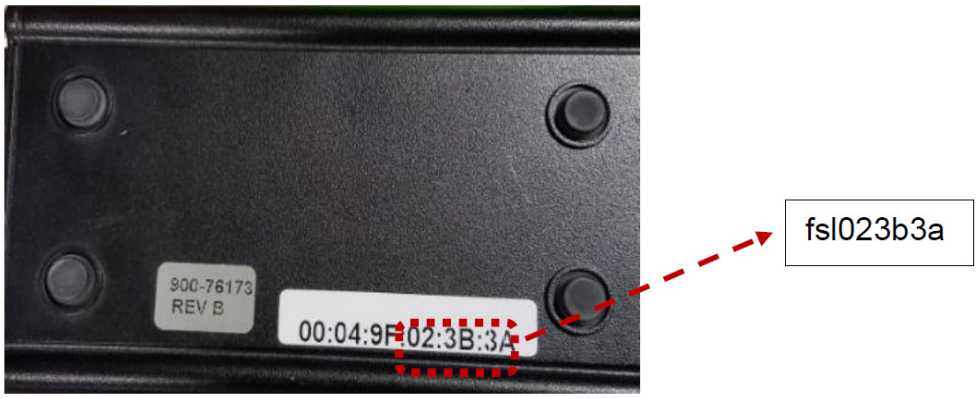

图 3：CWTAP 标签上 MAC 地址的最后六位

4. 验证摘要

所有验证均已使用 LSDK 18.06 通过 U-Boot 执行完毕。目前，本脚本支持 LS1021A、LS1043A 和 LS1046A。将来还会支持更多器件。

注：通常，用于 DRAM 启动的 U-boot 编译目标不同于从闪存启动的目标。

为了从 NOR/QSPI 闪存启动，NOR/QSPI 需映射到存储器并支持本地执行。启动流程为：内核从闪存开始执行，启动代码配置 DDR 控制器，然后将 U-Boot 映像从闪存复制到 DRAM，最后跳转到 DRAM 中的代码，完成 U-Boot。

对于 SD/eMMC/NAND 启动，这些接口未进行存储器映射，并且不支持本地执行。可以利用名为 PBI（预启动加载程序初始化工具）的功能将小程序 SPL（二级程序加载程序）复制到 OCRAM。启动从驻留在 OCRAM 中的 SPL 开始。SPL 配置 DDR 控制器，然后跳转到 DRAM 中的代码，完成 U-Boot。对于 SD/eMMC/NAND，编译器会生成两个二进制映像，一个是 `u-boot.bin`，这是 U-Boot 的 DRAM 版本，它不会尝试初始化 DDR。另一个是 `u-boot-with-spl-pbl.bin`，此映像将 u-boot、PBL 和 SPL 组合到单个映像中。

对于 DRAM 启动，始终使用来自编译目标 SD/eMMC 或 NAND 的 `u-boot.bin`，即使是 NOR/QSPI 启动亦是如此。

例如，对于 LS1043A NOR，DRAM 启动目标是 `ls1043ardb_sdcard_defconfig`，而 NOR 闪存启动目标是 `ls1043ardb_defconfig`。

DRAM 启动的映像名称始终为 `u-boot.bin`。对于闪存启动，名称可能有所不同，为 `u-boot.bin` 或 `u-boot-with-spl-pbl.bin`。

表 1：恩智浦电路板及其测试摘要

器件	电路板	启动源	用于 DRAM 启动的 U-Boot 编译目标映像	用于闪存启动的 U-Boot 编译目标映像	备注
LS1021A	LS1021A Tower 板	NOR	ls1021 atwr_sdcard_ifc_defconfig u-boot.bin	ls1021 atwr_nor_defconfig u-boot.bin	DRAM 启动已经过测试 闪存启动已经过测试
		NAND	在 Tower 板上不支持 NAND。甚至无法将 cfg_rcw_src 设置为 NAND。	在 Tower 板上不支持 NAND	在 Tower 板上不支持。 ls1021aqds_nand_defconfig 可用作 DRAM 启动和闪存启动的参考目标。 u-boot.bin 和 u-boot-with-spl-pbl.bin 文件是 RAM 启动和 NAND 启动的映像
		QSPI	ls1021 atwr_sdcard_qspi_defconfig u-boot.bin	ls1021 atwr_qspi_defconfig u-boot.bin	DRAM 启动已经过测试 闪存启动已经过测试
		SD	ls1021 atwr_sdcard_ifc_defconfig u-boot.bin	ls1021 atwr_sdcard_ifc_defconfig u-boot-with-spl-pbl.bin	DRAM 启动已经过测试 闪存启动已经过测试
LS1043A	LS1043 AR DB	NOR	ls1043ardb_sdcard_defconfig u-boot.bin	ls1043ardb_defconfig u-boot.bin	DRAM 启动已经过测试 闪存启动已经过测试
		NAND	ls1043 ardb_nand_defconfig u-boot.bin	ls1043 ardb_nand_defconfig u-boot-with-spl-pbl.bin	DRAM 启动已经过测试 闪存启动已经过测试
		SD	ls1043ardb_sdcard_defconfig u-boot.bin	ls1043 ardb_sdcard_defconfig u-boot-with-spl-pbl.bin	DRAM 启动已经过测试 闪存启动已经过测试
		QSPI	ls1043ardb_sdcard_defconfig u-boot.bin	LS1043ARDB 和 LSDK 18.06 不支持 QSPI 闪存。 用户可以使用 ls1043aqds_qspi_defconfig 作为参考目标。应将 u-boot.bin 烧写到 QSPI 闪存。	DRAM 启动已经过测试
LS1046A	LS1046 AR DB	SD	ls1046ardb_sdcard_defconfig u-boot.bin	ls1046ardb_sdcard_defconfig u-boot-with-spl-pbl.bin	DRAM 启动已经过测试 闪存启动已经过测试
		QSPI	ls1046ardb_sdcard_defconfig u-boot.bin	ls1046ardb_qspi_defconfig u-boot.bin	DRAM 启动已经过测试 闪存启动已经过测试
		NOR	ls1046ardb_sdcard_defconfig u-boot.bin	LS1046ARDB 和 LSDK 18.06 不支持 NOR 闪存。 用户可以使用	DRAM 启动已经过测试

器件	电路板	启动源	用于 DRAM 启动的 U-Boot 编译目标映像	用于闪存启动的 U-Boot 编译目标映像	备注
				ls1046aqds defconfig 作为参考目标。应将 u-boot.bin 烧写到 NOR 闪存。	

5. 实用信息

1. 目前，在 rcw-override 过程中，仅 rcw11 使用以下 CCS 命令覆盖：

```
ccs::write_reg 0 rcw11 0x00030000
```

如果需要，也可以通过在此过程中添加新的覆盖命令来覆盖其他 RCW 字。请注意，TCL 脚本中的 rcw 索引从 0 到 15。

> **注：**用户必须注意不要覆盖/改变 PLL 比率。它始终使用 RM 中记录的比率。如果在此处添加了 PLL 覆盖，即使更改了 DCFG 的 RCWSR1-16，在 PLL 中覆盖也不会生效。

2. 对于所有可能的硬编码 RCW 选项：
   - LS1043A DDR 的运行比率始终为 13:1。
   - LS1046A DDR 的运行比率始终为 16:1。
   - LS1021A 具有更多时钟选择。

   使用 QCVS 获得的 DDR 控制器寄存器的值应考虑如上所示的 DDR 时钟速率，并且不能通过 RCW 覆盖来更改。

3. 为了测试 JTAG 接口是否正常工作，请在 CCS 窗口中使用以下命令：

```
source IDcode.tcl
```

如果 JTAG 接口工作正常，则将显示 Layerscape 器件 ID。

4. 在 rcw_override 的最后部分，将显示 JTAG 链信息。如果仅显示：

```
Chain Position 0: LSxxxxA
Chain Position 1: DAP
Chain Position 2: SAP2
```

这意味着覆盖失败。如果覆盖成功，则应显示更多的链位置，包括所有 ARM 内核。如果覆盖成功，最后一个链如下所示。

LS1021A：`Chain Position 18:SAP2`

LS1043A：`Chain Position 33: SAP2`

LS1046A：`Chain Position 33: SAP2`

5. 为了测试 DDR 是否正常工作，您可以在 `lsbp.tcl` 中将以下两行注释掉：

```
#lsbp::loadb $sap2
#ccs::run_core $boot_core
```

这样，`lsbp.tcl` 初始化 DDR 控制器时不会将 u-boot 映像加载到 DDR，并会释放 ARM 内核。

使用以下命令读取对存储器的写入。用于 LS1021A 的 SAP2 为 18。对于 LS1043A/LS1046A，则将 18 替换为 33。

```
(bin) 80 % ccs::write_mem 18 0 0x82000000 4 0 {0x1 0x2 0x3 0x4}
(bin) 81 % disp ccs::read_mem 18 0 0x82000000 4 0 8
                              +0       +4       +8       +C
[0x0000000082000000] 00000001 00000002 00000003 00000004
[0x0000000082000010] E59FF014 E59FF014 E59FF014 E59FF014
```

```
(bin) 82 % disp ccs::read_mem 18(or 33) 0 0x1080000 4 0 0x400  #display all
DDR registers.
```

6. 其他一些 CCS 调试命令：

   a) 以下两行可读取 PC。对于 LS1021A，chain_position_for_first_core 为 9。对于 LS1043A 和 LS1046A 则为 16。在读取前必须先将停止内核。

```
ccs::stop_core chain_position_for_first_core
disp ccs::read_reg chain_position_for_first_core pc 1
```

   b) 下一行执行一条指令。要运行＃条多指令，可使用更大的数字取代 1。

```
ccs::step_core chain_position_for_first_core 1
```