Application Report
Lit. Number – Feb 2014

通过 IBL 扩展 Keystone I DSP 启动模式

Allen Yin
Multicore Application

摘 要

Keystone I DSP 可以支持多种启动方式如 RapidIO，Ethernet， PCIe boot 等等，但在实际应用中，由于内部 ROM 大小的限制，内部的 RBL(Rom Bootloader)不能满足客户对启动模式的需求，比如，使用 Linux 操作系统时，通常需要芯片支持 NAND FLASH 启动，但在 C667x 系列 DSP 上，RBL 不能支持直接从 NAND FLASH 启动；有些用户可能希望 DSP 能够到指定的远程 TFTP 服务器上下载镜像启动。为了满足这些需求，扩展 DSP 启动模式，在新的 MCSDK（Multi-Core Software Development Kit） 里，引入了中间启动加载器（Intermediate Boot Loader，简称 IBL），本文将向读者介绍 IBL 的使用方法和场景。

目 录

图 例

1 引言

Keystone I DSP 自身可以支持多种启动模式，以常用的 C667x 为例，支持的启动模式包括 RapidIO，Ethernet，PCIe，I2C，SPI 以及 Hyperlink，这些启动模式都是通过 DSP 内部 ROM 的 Boot Loader（RBL）完成。用户通过 DSP 引脚设置不同的启动模式，在上电时序完成以后，DSP 执行 RBL，根据不同的启动设置初始化对应的外设接口，读取或者下载用户的启动镜像进行启动。

用户使用 RBL 启动非常方便，但是因为内部 ROM 大小的限制，RBL 不能满足越来越丰富的启动模式的需求，为了满足这些需求，扩展启动模式，在新的 MCSDK（Multi-Core Software Development Kit） 里，引入了中间启动加载器（Intermediate Boot Loader，简称 IBL），IBL 启动的流程如下图，

图 1. IBL 流程图

DSP 上电以后，通过 RBL 从 I2C 启动，读取 I2C EEPROM 里的 IBL 程序并执行，同时用户可以通过对 DSP 管脚的配置选择 IBL 启动的分支如 NAND FLASH，NOR FLASH 或 TFTP，IBL 会配置对应的接口设备，并从中取得启动镜像后跳转到用户的程序入口处开始执行。

2 IBL 介绍

2.1 启动模式

为了弥补 RBL 的不足，IBL 目前支持三种启动模式，NAND FLASH, NOR FLASH 和 TFTP 启动

2.1.1 NAND FLASH

在 C667x 系列 DSP 上，NAND FLASH 芯片通过 EMIF 接口与 DSP 连接。C667x 系列 DSP 不支持直接从 NAND FLASH 启动，需要通过 I2C 启动 IBL 初始化 NAND FLASH，从 NAND FLASH 读取启动镜像后转到相应的程序入口处执行。IBL 支持 NAND FLASH 中两个不同位置的启动镜像，镜像所处的位置可以通过 GEL 脚本进行配置。

2.1.2 NOR FLASH

IBL 同样支持从 NOR FLASH 启动，在 C667x 系列 DSP 的 EVM 板上，NOR FLASH 芯片通过 SPI 接口与 DSP 连接，IBL 需要初始化 SPI 接口以读取 NOR FLASH 中的启动镜像。IBL 支持 NOR FLASH 中两个不同位置的启动镜像，镜像所处的位置可以通过 GEL 脚本进行配置。

2.1.3 TFTP

C667x 系列 DSP 本身支持 EMAC 启动方式，在这种启动方式下，DSP 自身初始化以太网接口以后向主控方发出 BOOTP 包，主控方在收到 BOOTP 包以后将启动镜像通过以太网包发给 DSP。 IBL 支持的以太网启动与此过程不同，它将主动启动一个 FTP 过程，到指定的 FTP 服务器上读取镜像文件，然后用此镜像进行启动，使用的服务器地址和文件名可以通过 GEL 脚本进行配置。

2.2 文件格式

IBL 目前支持两种文件格式，常用的 ELF 和 BBLOB，TI 也正在继续完善 IBL 以支持更多的文件格式。

2.2.1 ELF

ELF 是由 USL(UNIX System Laboratories)作为 ABI(Application Binary Interface) 的一部分发布的，ELF 为开发者提供了一整套能应用于多个操作系统的二进制接口定义,在软件移植时大大减少了需要重新编码和编译的地方。TI 的 C6000 编译器从 V7.0 开始支持 ELF 文件格式，并把它作为缺省的输出格式。

2.2.2 BBLOB

BBLOB（Big Binary Large Object）文件格式是 16 进制文件格式，在 IBL 中，BBLOB 文件格式是从内存直接映射生成，现在也可以通过 CCS 直接生成 BBLOB 文件格式，通常 BBLOB 文件格式可用于启动 Linux 镜像。

2.3 IBL 操作方法

本节将主要介绍 IBL 的操作方法，包括烧写 I2C EEPROM，NAND FLASH 或 NOR FLASH，以及如何生成 ELF 文件以及 BBLOB 文件格式。

2.3.1 烧写 I2C EEPROM

使用<MCSDK>\tools\writer\eeprom\（因 MCSDK 的版本不同，目录可能略微不同，本文实验均在 MCSDK 2.1.2.6 下实现）里提供的 I2C EEPROM 烧写软件，可在 JTAG 连接的环境下烧写 IBL 数据文件到 I2C EEPROM，IBL 的镜像文件通常可以在<MCSDK>\tools\boot_loader\ibl\src\make\bin 目录下，具体的操作步骤应参考<MCSDK>\tools\writer\eeprom\docs 内的说明文档，使用的文件格式为.dat 文件，用户也可以修改 writer 的源代码以适应自己的硬件环境。烧写完成可以在控制台看到如下反馈信息，

Writing 51768 bytes from DSP memory address 0x0c000000 to EEPROM bus address 0x0051 starting from device address 0x0000 ...

Reading 51768 bytes from EEPROM bus address 0x0051 to DSP memory address 0x0c010000 starting from device address 0x0000 ...

Verifying data read ...

EEPROM programming completed successfully

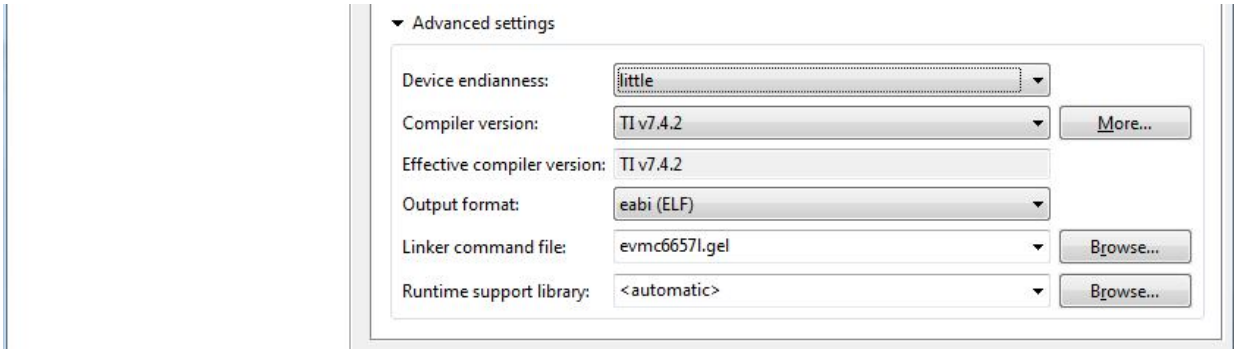
默认情况下，在 EVM 上运行 IBL 时，通常将 POST 程序烧写到 0x50 地址处，将 IBL 烧写到 0x51 地址处，如果需要改变此地址，需要重新配置和编译 IBL，重新编译 IBL 的步骤可参考<MCSDK>\tools\boot_loader\ibl\doc\build_instructions.txt。

2.3.2 生成启动文件

注意 IBL 需要使用 0x00800000~0x0081FFFF 作为 IBL 的运行时内存空间，所以用户启动镜像不应在启动时对此段内存进行写入的操作，这也就意味着用户代码不能将这段内存空间用于代码段，常数段等初始化段。

- ELF

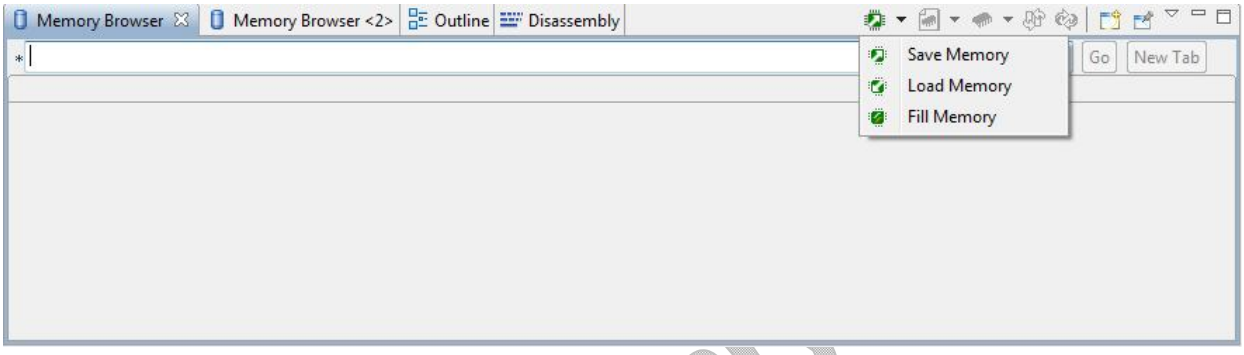
ELF 格式可以通过编译器直接生成，用户可以使用 CCS 配置输出文件的格式；



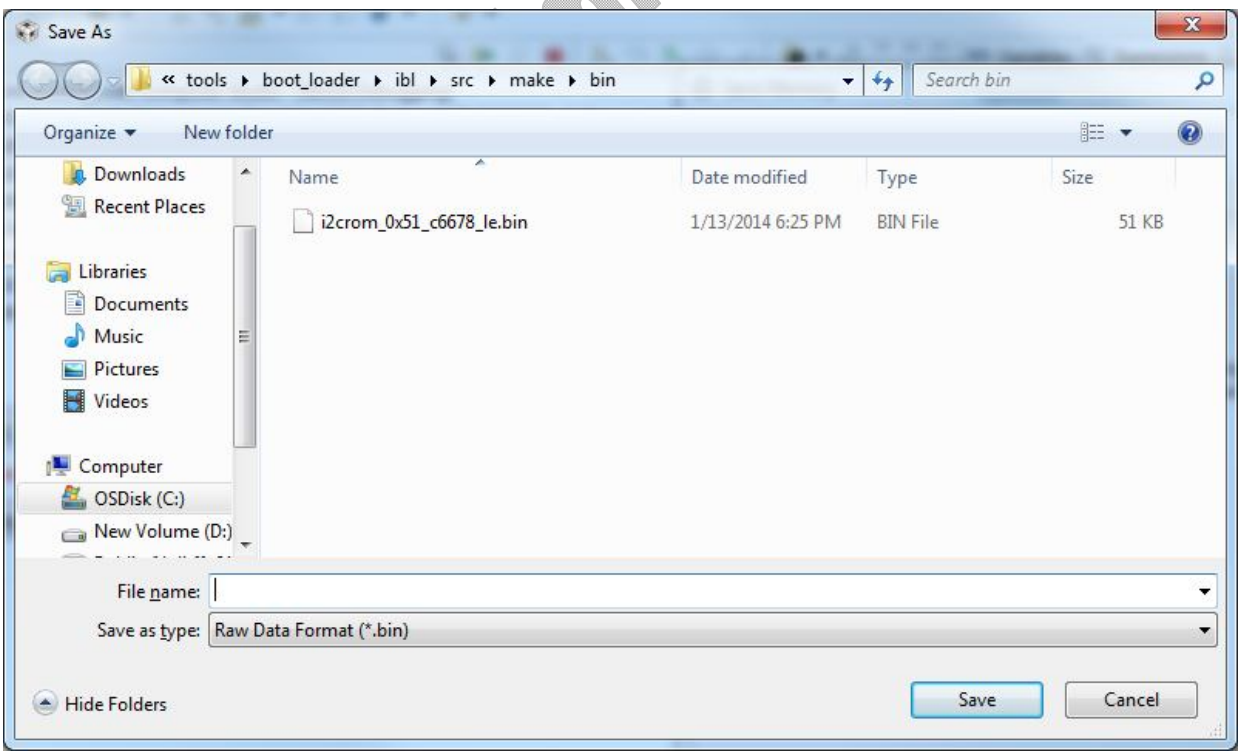
- BBLOB

BBLOB 文件格式是内存内容的直接映射，现在也可以通过 CCS 直接生成，但 BBLOB 文件只能支持连续的一整段内存地址空间，所以需要两个参数，起始地址和长度。通过 CCS 生成 BBLOB 格式的步骤如下，

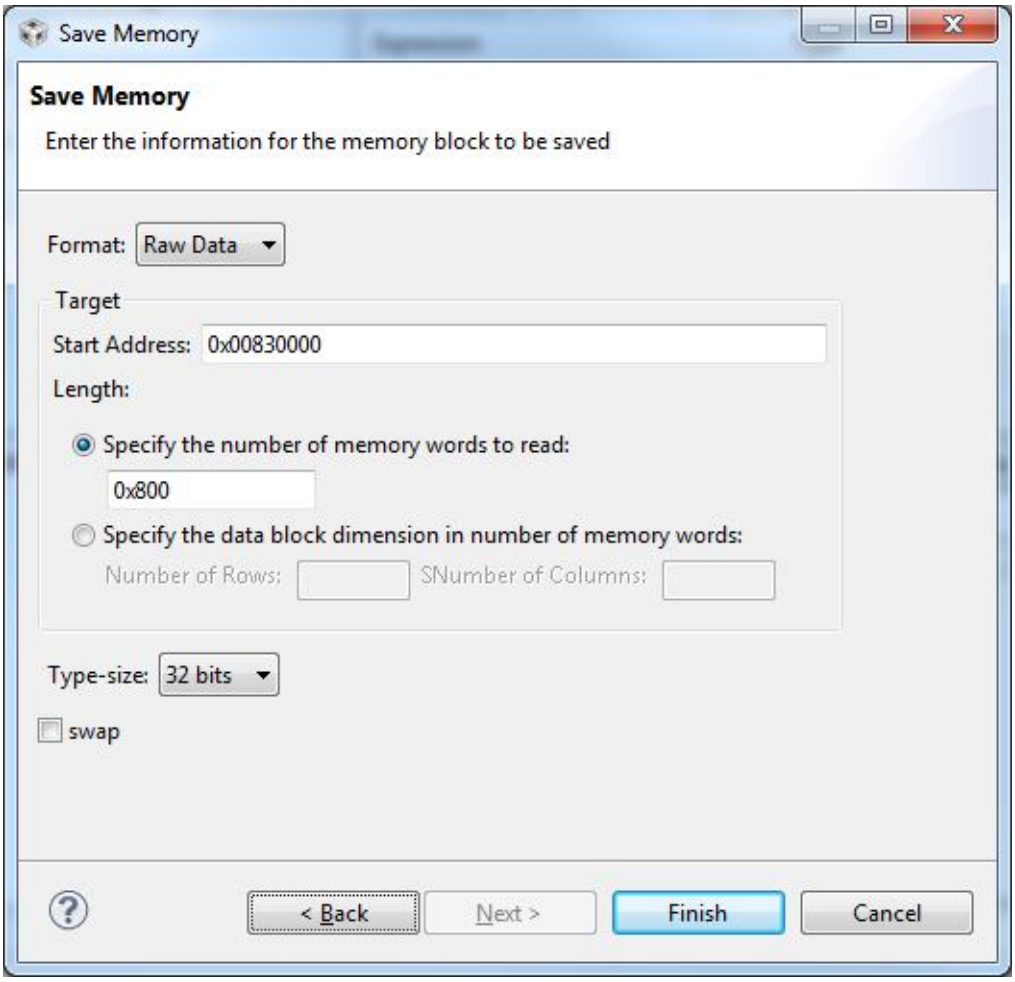
1. 在 NO BOOT 模式下通过 JTAG 连接 DSP，使用 GEL 脚本进行相应的初始化；
2. 加载可执行文件到 DSP；
3. 在 View 菜单下选取 Memory Browser，点击 Save Memory；



4. 保存文件时，选取 BIN 文件格式；



5. 输入需要保存的内存起始地址和长度即可，注意要用 32bit 方式保存；此时应记录三个参数，起始地址和长度，程序入口地址(_c_int00)以备 IBL 配置使用。



2.3.3 通过 GEL 配置 IBL

在 EVM 板上，IBL 启动模式（NAND FLASH,NOR FLASH,TFTP）可以通过 DIP 开关进行配置，用户可以参考<MCSDK>\tools\boot_loader\ibl\doc 内的说明文档。每一种启动模式下的启动参数，如 NAND FLASH 和 NOR FLASH 模式下的镜像文件位置和格式（ELF 或 BBLOB），BBLOB 文件写入的地址和长度以及程序入口地址，TFTP 模式下的镜像文件格式以及在服务器文件名，IBL 自身和服务器的 IP 地址，都可以通过 GEL 脚本进行配置，配置的步骤如下，

1. 将 DSP 置于 No-boot 启动模式，上电并通过 JTAG 连接，使用 GEL 初始化 DSP；

2. 加载参数烧写程序，对于 C6678 EVM 文件为

<MCSDK>\tools\boot_loader\ibl\src\make\bin\i2cparam_0x51_c6678_le_0x500.out；

3. 修改后加载<MCSDK>\tools\boot_loader\ibl\src\make\bin\i2cConfig.gel，例如需要修改C6678 为 NAND FLASH 启动，使用 ELF 文件格式，则需要在 GEL 文件中找到 setConfig_c6678_main 函数，将

```
ibl.bootModes[1].u.nandBoot.bootFormat      = ibl_BOOT_FORMAT_BBLOB;
```

修改为

```
ibl.bootModes[1].u.nandBoot.bootFormat      = ibl_BOOT_FORMAT_ELF;
```

如果使用 BBLOB 方式启动，还需要指定对应的写入地址和长度以及程序入口地址，通过对应参数进行配置，如

```
ibl.bootModes[1].u.nandBoot.blob[0][0].startAddress  = 0x00830000;
ibl.bootModes[1].u.nandBoot.blob[0][0].sizeBytes     = 0x02000;
ibl.bootModes[1].u.nandBoot.blob[0][0].branchAddress = 0x008315A0;
```

在 TFTP 启动方式下，用户可以自行配置服务器地址和使用的文件名，如

```
SETIP(ibl.bootModes[2].u.ethBoot.ethInfo.ipAddr,  192,168,2,100);
SETIP(ibl.bootModes[2].u.ethBoot.ethInfo.serverIp,  192,168,2,101);
SETIP(ibl.bootModes[2].u.ethBoot.ethInfo.gatewayIp, 192,168,2,1);
SETIP(ibl.bootModes[2].u.ethBoot.ethInfo.netmask,  255,255,255,0);
```


4. 运行第 2 步加载的可执行文件，程序将等待 GEL 的输入，此时运行 GEL 里相应的函数，等待 10 秒左右，在 CCS 里的控制台下输入回车使程序继续执行，相应的参数即可写入对应的 I2C EEPROM 参数区中，成功以后，可见如下回应。

```
Run the GEL for for the device to be configured, press return to program the I2C
I2c table write complete
```

2.3.4 烧写 NAND FLASH 或 NOR FLASH

使用<MCSDK>\tools\writer\nand(nor)\（因 MCSDK 的版本不同，目录可能略微不同）里提供的 NAND FLASH 和 NOR FLASH 烧写软件，可在 JTAG 连接的环境下烧写启动镜像文件到 NAND FLASH 或 NOR FLASH，,具体的操作步骤应参考<MCSDK>\tools\writer\nand(nor)\docs 内的说明文档，用户也可以修改 writer 的源代码以适应自己的硬件环境。

2.3.5 使用 TFTP

在 TFTP 启动模式下，用户可以在服务器启动 TFTP 服务，将对应的启动镜像文件拷贝到 TFTP 目录下，并将其改名为 IBL 或 GEL 脚本指定的文件名；IBL 将通过 TFTP 下载镜像并进行启动。

2.4 量产说明

在实际的应用中，用户可以先使用上述步骤通过 JTAG 在自己的硬件板上调试 IBL，直至功能正常以后，根据调试后稳定的 GEL 文件里的配置参数，修改 <MCSDK>\tools\boot_loader\ibl\src\util\iblConfig\src\device.c 对应参数并重新编译 IBL（重新编译 IBL 的步骤可参考<MCSDK>\tools\boot_loader\ibl\doc\build_instructions），之后重新烧写 IBL，再启动就不需要 GEL 脚本来配置参数，从而可以脱离 JTAG 调试环境，用户可以只关注于自己的程序调试工作。

如果用户觉得重新编译 IBL 比较麻烦，也可以通过复制 I2C EEPROM 内容的方法，将已经由 GEL 脚本配置好参数的 IBL 镜像和参数一并拷贝到新的硬件板上以实现 IBL 启动。

3 IBL 实例

本节将使用 OST 程序模拟用户程序在 C6678 EVM 板上进行 NAND FLASH 的启动，安装 MCSDK 后，POST 工程和代码可以在<MCSDK>\tools\post\evmc6670l 目录下找到。

1. 编译 POST 工程代码，注意输出格式为 ELF 格式；

2. 设置 EVM 跳线到 No Boot 启动并使用 GEL 脚本初始化 EVM，烧写 IBL 程序到 I2C EEPROM 0x51 处，并通过 GEL 脚本修改启动参数为 NAND FLASH 启动，启动的文件格式为 ELF；

3. 烧写 POST 程序文件到 NAND FLASH 里；

4. EVM 下电，设置 EVM 跳线为 IBL NAND FLASH 启动，因为 IBL 和 POST 均使用 UART 串口输出打印信息，此时连接 EVM 串口到 PC 机并做相应设置；

```
SW3 (off, off, on, off)
SW4 (on, off, on, on)
SW5 (on, on, on, off)
SW6 (on, on, on, on)
```

5. 重新上电启动 EVM，可以在串口观察到 IBL 的输出为

```
IBL version: 1.0.0.16

IBL: PLL and DDR Initialization Complete

IBL Result code 00

IBL: Booting from NAND

TMDXEVM6678L POST Version 01.00.00.06
```

```
----------------------------------------
SOC Information
FPGA Version: 0007
EFUSE MAC ID is: 90 C 36
SA is disabled on this board.
PLL Reset Type Status Register: 0x00000001
Platform init return code: 0x00000000
Additional Information:
  (0x02350014) :0BEF0000
  (0x02350624) :000215FF
  (0x02350678) :00831F70
  (0x0235063C) :00081800
  (0x02350640) :00091800
  (0x02350644) :000A1800
  (0x02350648) :000B1800
  (0x0235064C) :000C1800
  (0x02350650) :000D1800
  (0x02350654) :000E1800
  (0x02350658) :000F1800
  (0x0235065C) :00000009
  (0x02350660) :00832038
  (0x02350668) :0083204C
  (0x02350670) :00832060
  (0x02620008) :0300F015
  (0x0262000c) :0401412E
  (0x02620010) :00000000
  (0x02620014) :774A0000
  (0x02620018) :0009E02F
```

```
 (0x02620180) :0602F000

----------------------------------------

Power On Self Test

POST running in progress ...

POST I2C EEPROM read test started!

POST I2C EEPROM read test passed!

POST SPI NOR read test started!

POST SPI NOR read test passed!

POST EMIF16 NAND read test started!

POST EMIF16 NAND read test passed!

POST EMAC loopback  test started!

POST EMAC loopback  test passed!

POST external memory test started!

POST external memory test passed!

POST done successfully!

POST result: PASS
```


4 结论

从本文可以看出，通过 IBL 可以较好的弥补 RBL 不能满足的一些应用场景，如 NAND FLASH,NOR FLASH 和 TFTP 启动，ELF 和 BBLOB 文件格式的支持；同时，TI 开放了 IBL 的源代码，用户完全可以根据自己的需要，在此基础上修改出更适合自己的 IBL 实例。

参考文献

1. *C66x DSP Bootloader User Guide (SPRUGY5)*
2. *TMS320C6678 data manual (SPRS691)*
3. *BIOS-MCSDK User Guide*