## 开始前准备

为方便用户快速使用 Primace 设计 M7 器件，本手册将基于软件安装包自带的例程和 M7 器件，介绍 M7 器件 ucLinux 的完整设计流程。开始之前，请先确保：

- **Primace（7.0），Keil，USB 及 ADGI 驱动已经安装，并能正常运行；**
- **准备好 CME-M7 器件及下载线；**
- **准备好演示文件（位于软件安装目录下，默认：**

  ***C:\capital_micro\primace7.0\Examples\M7\primace\M7***

  ***C:\capital_micro\primace7.0\Examples\M7\3rdParty\keil\M7_release\Combo\CME_Linux_RunByBootloader***

本实例演示基于以下平台和器件：

- **Primace7.0 设计套件**
- **Keil µVision 工具**
- **CME-M7-EVB-BGA484 开发板**

## linux 设计实例

本实验是在虚拟机中安装的 fedora14 进行的。

将 primace7.0 安装目录下 X:\capital_micro\primace7.0\Examples\M7\3rdParty 中的 uclinux.tar.gz 放到 linux 中的某一目录下，并解压，本实验以放到“/”目录下为例。

### 步骤1　启动

在 linux 中，打开终端，进入到刚刚解压的工程目录下，输入 source ACTIVATE.sh，执行启动命令，如下如所示。

```
[root@test /]# ls
bin     dev   lib                  media  opt   sbin     sys             usr
boot    etc   linux-cortexm-1.12.0 mnt    proc  selinux  tmp             var
cgroup  home  lost+found           null   root  srv      uclinux.tar.gz
[root@test /]# cd linux-cortexm-1.12.0/
[root@test linux-cortexm-1.12.0]# ls
A2F  ACTIVATE.sh  linux  projects  tools  u-boot
[root@test linux-cortexm-1.12.0]# source ACTIVATE.sh
```

## 步骤2　添加工程

把自己的工程文件加入到 project/ucliux/hello 目录下，本实验里已经放入的工程， hello 工程实现的功能是打印“Hello, A2F-Linux!”文字。

```
[root@test linux-cortexm-1.12.0]# cd projects/
[root@test projects]# ls
developer  hello  networking  Rules.make  uclinux
[root@test projects]# cd uclinux/
[root@test uclinux]# ls
bin2array.c  hello  Makefile         uclinux.initramfs
builder      local  uclinux.busybox  uclinux.kernel.CMEM7
[root@test uclinux]# cd hello/
[root@test hello]# ls
hello.c  hello.h  Makefile
```

## 步骤3　添加应用到 initramfs

在 uclinux.initramfs 文件后面添加‘/bin/hello’，本工程中已经添加好了，如下图所示

```
[root@test hello]# cd ..
[root@test uclinux]# ls
bin2array.c  hello  Makefile         uclinux.initramfs
builder      local  uclinux.busybox  uclinux.kernel.CMEM7
[root@test uclinux]# vim uclinux.initramfs
```

下图是 uclinux.initramfs 文件。

```
file /bin/hello ${INSTALL_ROOT}/projects/${SAMPLE}/hello/hello 755 0 0

dir /mnt 755 0 0
```

## 步骤4　修改 CUSTOM_APPS

把自己的应用名赋值给 project/uclinux/目录下的 Makefile 文件中的 CUSTOM_APPS，如下图所示。

```
[root@test uclinux]# vim Makefile
```

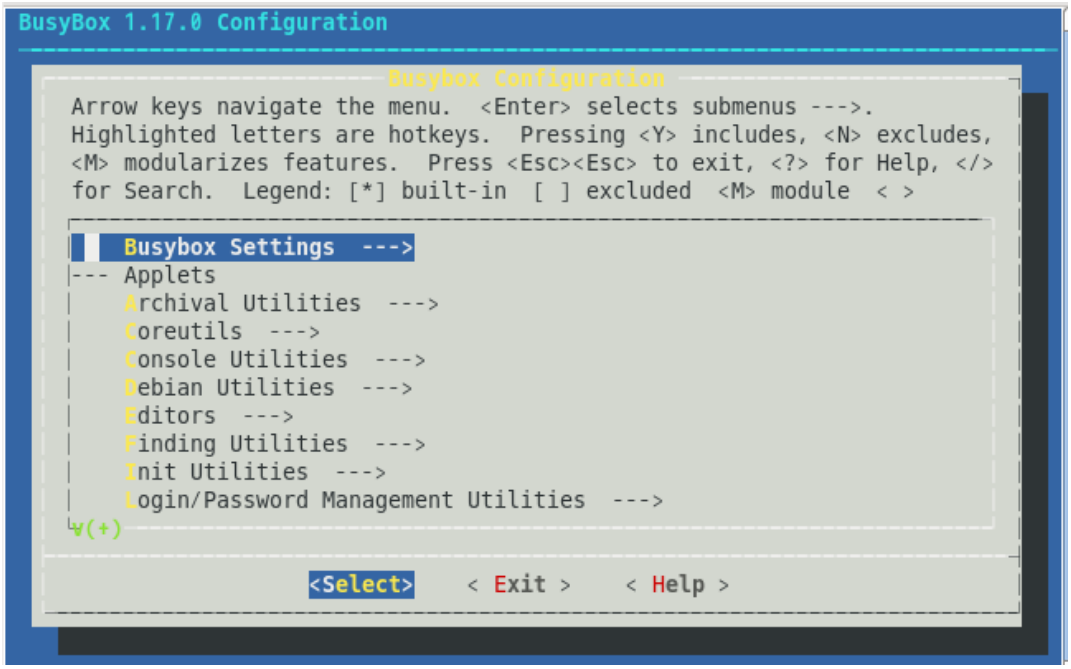
下图是 Makefile 文件。

```
# Makefile

SAMPLE          := uclinux
CUSTOM_APPS     := hello

include ../Rules.make

# end of file
~
```

## 步骤5　busybox 配置

进入到 uclinux 目录，执行 make bmenuconfig 进行 busybox 配置。如下图所示。

```
[root@test uclinux]# make bmenuconfig
```




## 步骤6　建立 busybox

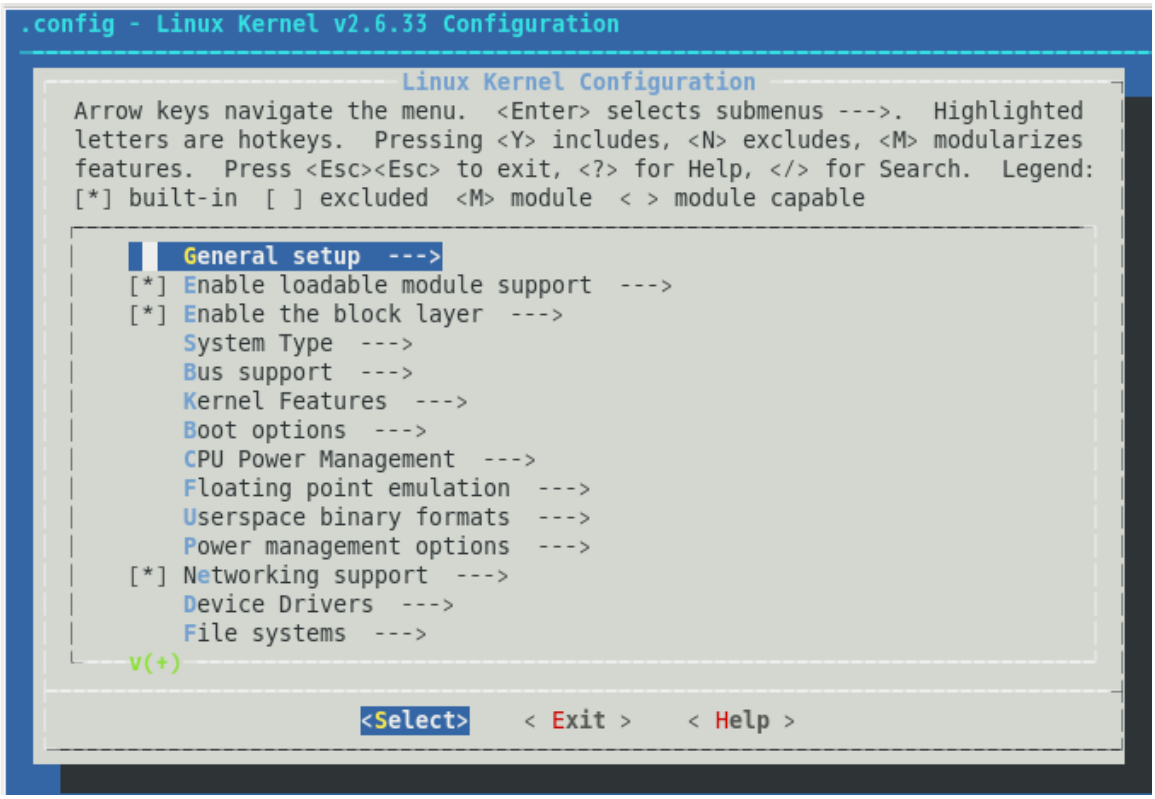
执行 make busybox，建立 busybox，如下图。

```
[root@test uclinux]# make busybox
```

## 步骤7　linux kernel 配置

执行 make kmenuconfig 命令，进行 linux kernel 配置，如下图所示。

```
[root@test uclinux]# make kmenuconfig
```




## 步骤8　生成 image 文件

步骤 5、6、7 若已经配置好，则可以跳过，直接执行下面命令

执行 make linux，建立 kernel 指令创建 initramefs，在当前目录下生成 image 文件，如下图所示。

```
[root@test uclinux]# make linux
```

```
[root@test uclinux]# ls
bin2array    builder  image  Makefile         uclinux.initramfs
bin2array.c  hello    local  uclinux.busybox  uclinux.kernel.CMEM7
```

## 步骤9　生成 image.hex 文件

将生成的 image 文件从 fedora 导出到 win7 中，更改文件类型，添加后缀名为.bin(在 fedora 中生成的 image 本身就是数据流的文件，可以再)。从网上下载一个 bin 文件和 hex 文件互换的转换工具，将 image.bin 转换成 image.hex 文件。如果想在 fedora 中直接生成带后缀的 image 文件，可在/project 目录下修改 Rules.make 文件。

## 步骤10　生成 zImage.c 文件（可选）

生成 hex 文件在前一步已经完成了，执行 make install，生成 zImage.c 文件，是一个用在在线调试的文件，可以加入到 keil 中调试。本实验不使用此种方法。

## ARM 设计实例

下面将介绍如何使用 Keil 生成.hex 文件。开始前，请确保 Keil 已经正确安装，同时下载线已连接至 PC。

Keil 工程主要是系统启动代码，实现 UART、FLASH、DDR 的初始化,然后将内核代码（即由 linux 生成的 image.hex 文件）由 spi-flash 拷贝至 DDR，拷贝完成后程序指针跳转到内核程序首地址。

使用 DDR 的原因是，M7 内部程序存储程序空间有限，需要扩展 DDR memory 来运行内核代码。

## 步骤1　运行 Keil

## 步骤2　打开工程

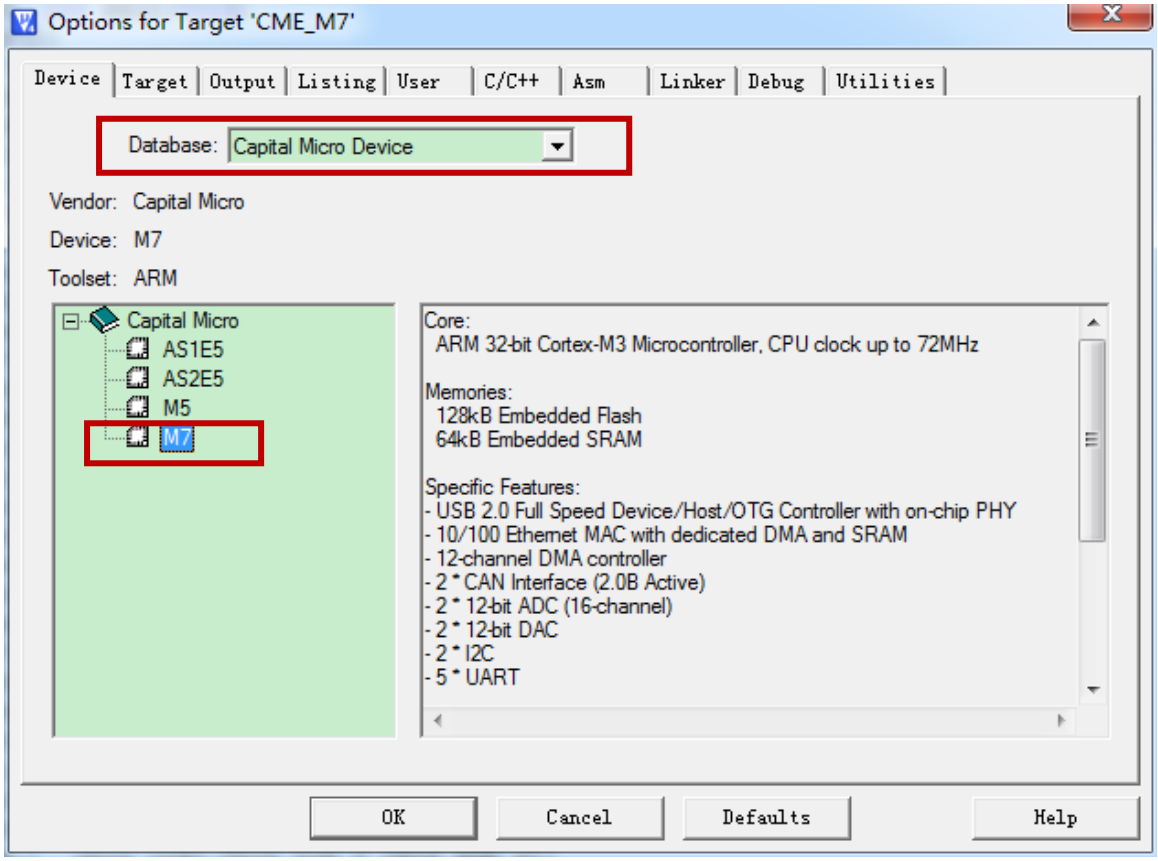
**运行 Keil 后，打开工程。此处以 UART 工程为例，工程路径：**

***C:\capital_micro\primace7.0\Examples\M7\3rdParty\keil\M7_release\Combo\CME_Linux_RunByBootloader\Bootloader***

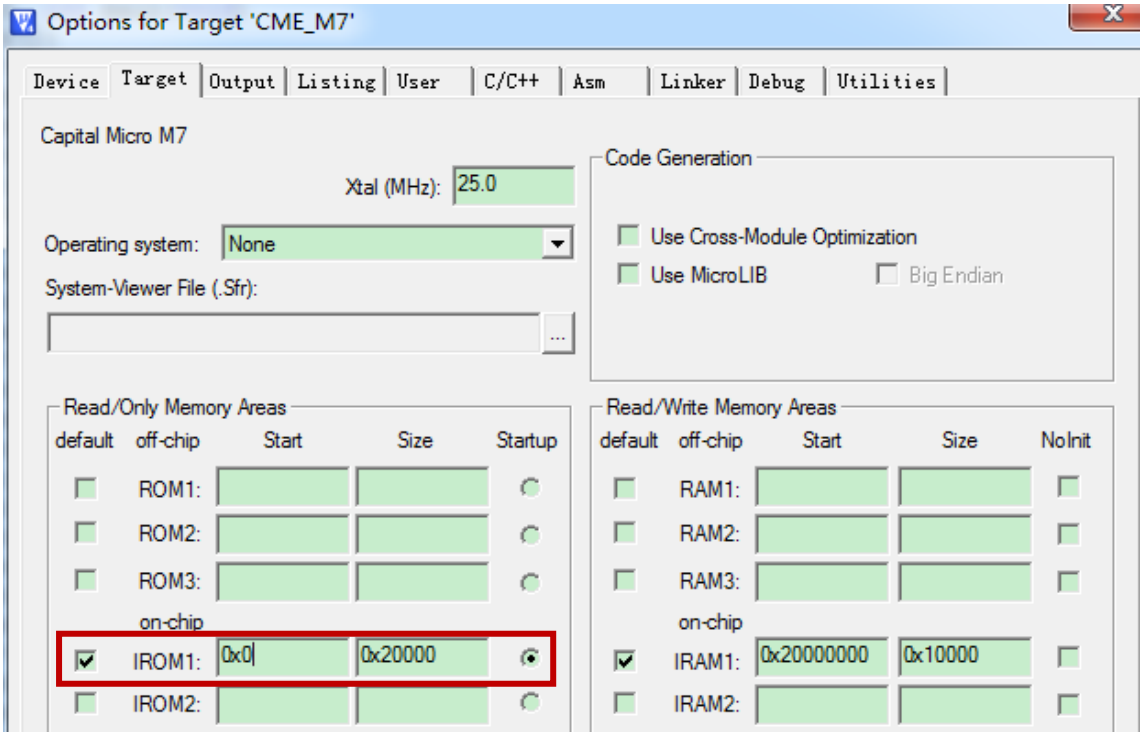
## 步骤3　设置 Keil 选项

点击  ，打开设置选项卡，分别进行如下设置。

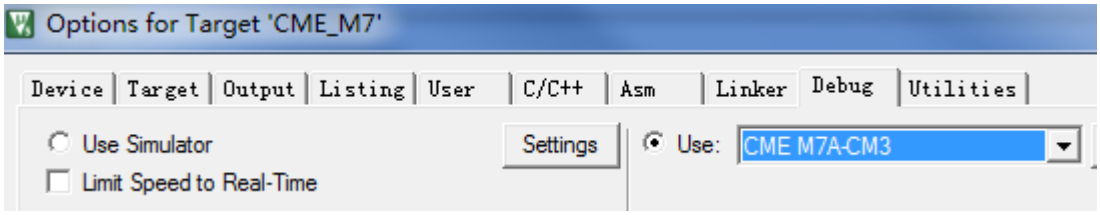
在 **Device** 页面，设置 **Database** 为 **Capital Micro Device**,**Capital Micro** 选择 **M7**



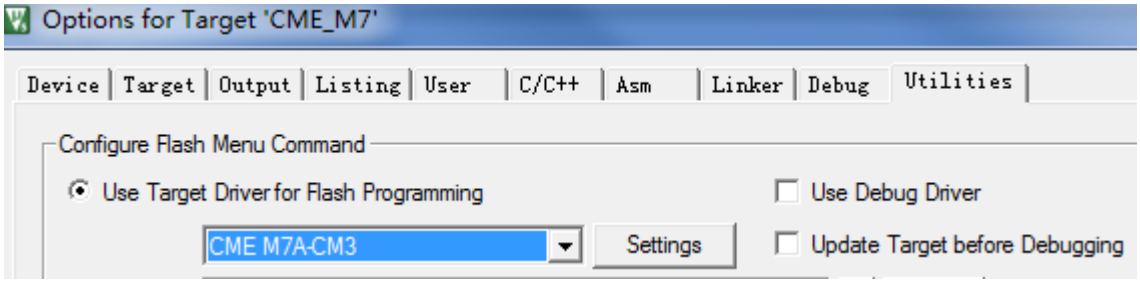

在 **Target** 页面，设置 **IROM1 的 Start** 为 **0x0**。




在 **Debug** 页面，设置 **Use** 为 **CME M7A-CM3**，如果不在线调试可以不进行此设置。




在 **Utilities** 页面，设置 **Use Target Driver for Flash Programming** 为 **CME M7A-CM3**，如果不在线调试可以不进行此设置。




提示

关于 Emulator 更多信息，可在 Primace 主界面点击 **Help**，打开 ***CME Emulator Quick Start*** 手册。

## 步骤4　编译

**点击  ，将生成一个 m7.hex 文件，文件保存路径：**

***C:\capital_micro\primace7.0\Examples\M7\3rdParty\keil\M7_release\Combo\CME_Linux_RunByBootloader\Bootloader\CME_M7***

## 步骤5　在线调试（可选）

点击  进行调试，Keil 界面下方的状态条会显示调试进度。

提示

关于 Emulator 更多信息，可在 Primace 主界面点击 **Help**，打开 ***CME Emulator Quick Start*** 手册。

# Primace 设计实例

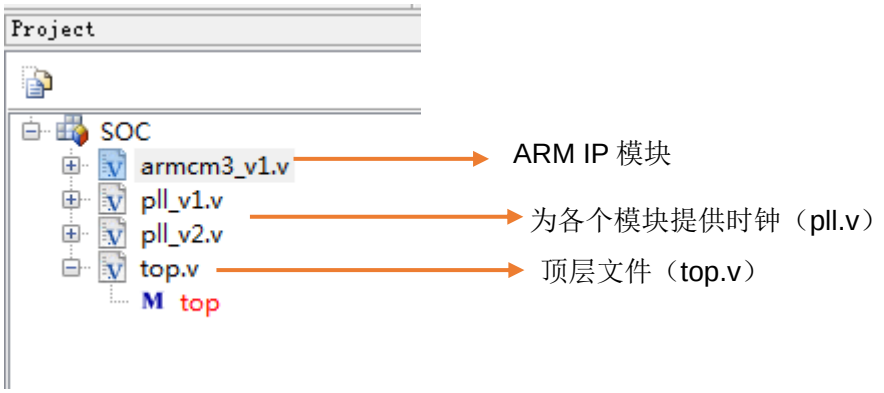
## 步骤1　运行 Primace

双击 图标或通过开始菜单运行 Primace。

## 步骤2　打开 Primace 工程

点击 **Project – Open Project**，打开 Primace 安装包自带的已经准备好的设计实例，路径：
***C:\capital_micro\primace7.0\Examples\M7\primace\M7***

此时，Primace 主界面的 **Project** 视图将列出所有工程文件，见下图。



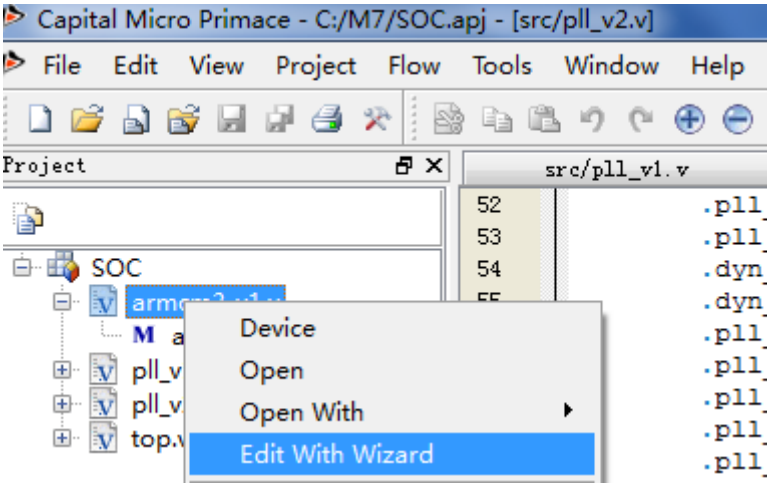

Linux 运行需要的最小系统硬件包括至少包括 PLL、arm-cortex IP wizard 中相关选项中勾选 UART、DDR。

注意

**使用 M7 器件时，需导入 ARM 编译后的输出*.hex 文件。.hex 文件的生成方式请参考“ARM 设计实例”一节。**

## 步骤3　导入 hex 文件

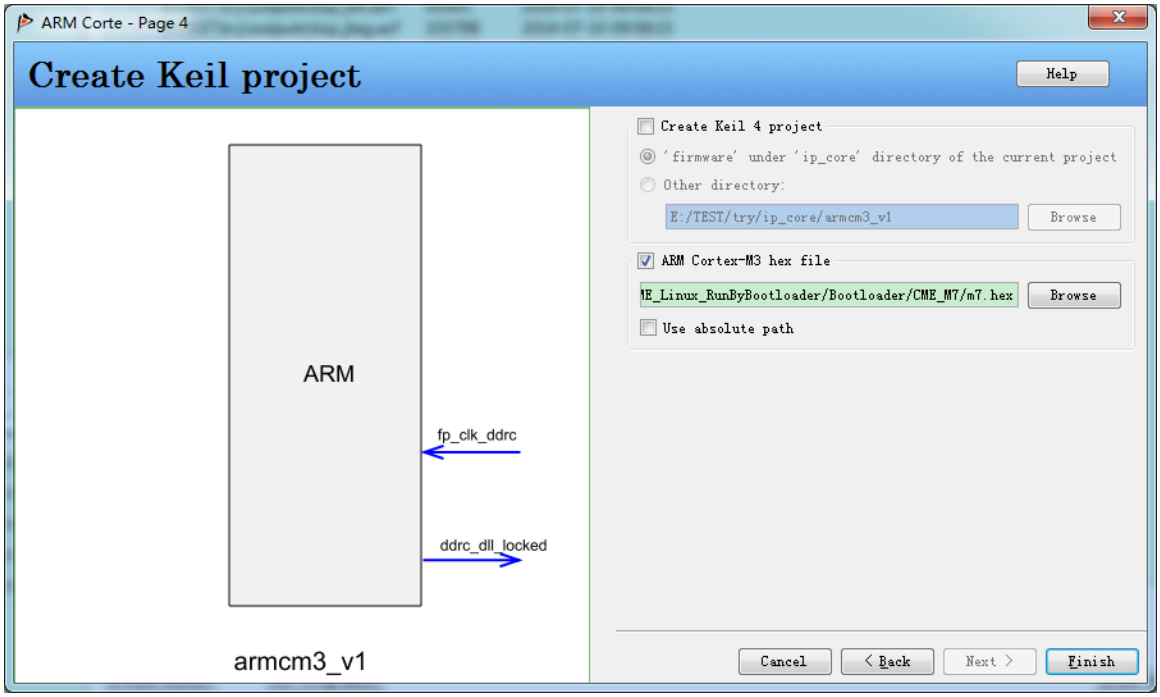
1). 展开 SOC 工程，在 **armcm3_v.1.v** 文件上点击鼠标右键，并选择 **Edit With Wizard**。




2). 打开 **Wizard**，连续点击 **Next**，直到进入 **Create Keil project** 页面。在 **ARM Cortex-M3 hex file** 前打勾，点击 **Browse**，点击 Browse 选择 keil 所生成的 hex 文件（**m7.hex**）。文件生成方式请参考“**ARM 设计实例**”一节），点击 **Finish**。

默认路径：

***C:\capital_micro\primace7.0\Examples\M7\3rdParty\keil\M7_release\Combo\CME_Linux_RunByBootloader\Bootloader\CME_M7***




## 步骤4　运行 Primace 工程

点击 Flow – Run Project 或 运行工程。

### 步骤5　生成比特流

点击 **Flow – Run Bitgen** 或  生成用于配置芯片的比特流文件。

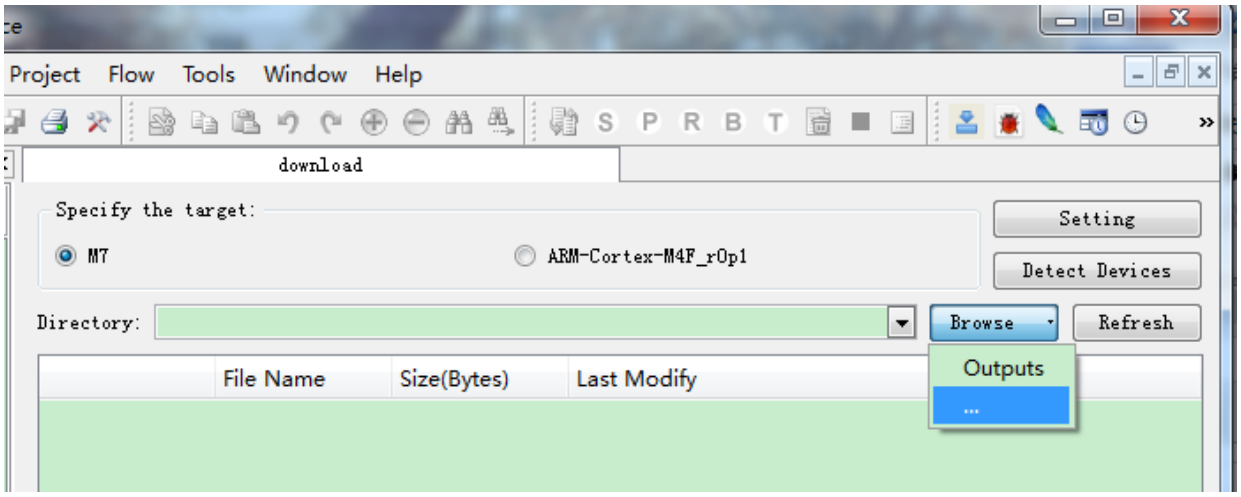
提示

**此时，生成的比特流文件已包含 ARM 程序代码。（关于 ARM 程序代码添加方式，请参考“导入 hex 文件”）**

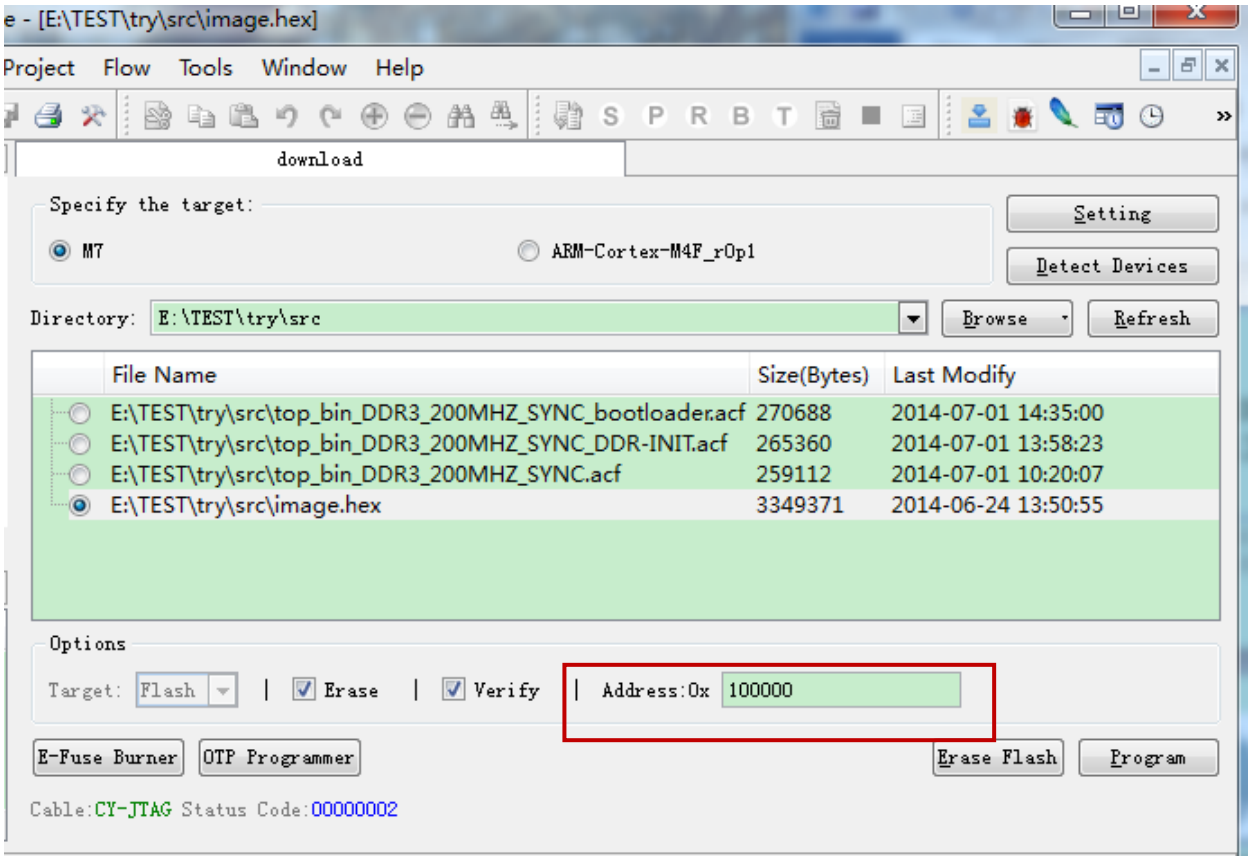
## M7 bitstream 和 uclinux image 下载

### 步骤1　烧写 image.hex 到 flash

1. 在 Primace 主界面点击 **Tools – Downloader** 或直接点击  按钮进入下载界面，选择目标器件和文件。
2. 点击 Browse，选择…，如下图所示。




3. 选择需要烧写的 image.hex 文件
4. 修改烧写地址的位置为 0x100000。如下图所示。




5. 点击  ，进行烧写。


## 步骤2　烧写.acf 文件到 flash

1. 打开 primace 的 downloader。
2. 点击 Browse，选择 Outputs。
3. 选择需要烧写的 primace 已经生成的 acf 文件
4. 修改烧写地址的位置为 0x0。
5. 点击 Program，进行烧写。

# 验证

- **系统启动过程**

若 spi-flash 空间（至少 2MByte）可以容纳内核程序，则可以将内核烧写至 spi-flash。这样系统需要两组代码，第一组代码是系统启动代码，实现 DDR 的初始化、然后将内核代码由 spi-flash 拷贝至 DDR，拷贝完成后程序指针跳转到内核程序首地址；第二组是系统内核代码。

1、FP 上电配置，拷贝初始化启动代码至 ARM-ROM 地址 0x00000000；
2、程序启动后，初始化 DDR，然后由 spi-flash 内核烧写地址拷贝内核代码到 DDR 地址 0x60000000；
3、程序指针跳转至 DDR 首地址 0x60000000 开始启动内核；
4、待内核启动后进行其他操作。

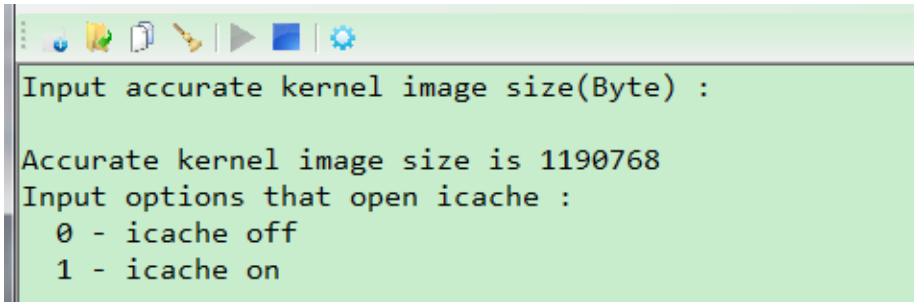
### 1) 连接器件和下载线

a. 使用下载线将预先准备好的 M7 器件和电脑连接，并打开器件；

b. 将串口数据线与 PC 连接，打开超级终端，并进行如下设置。

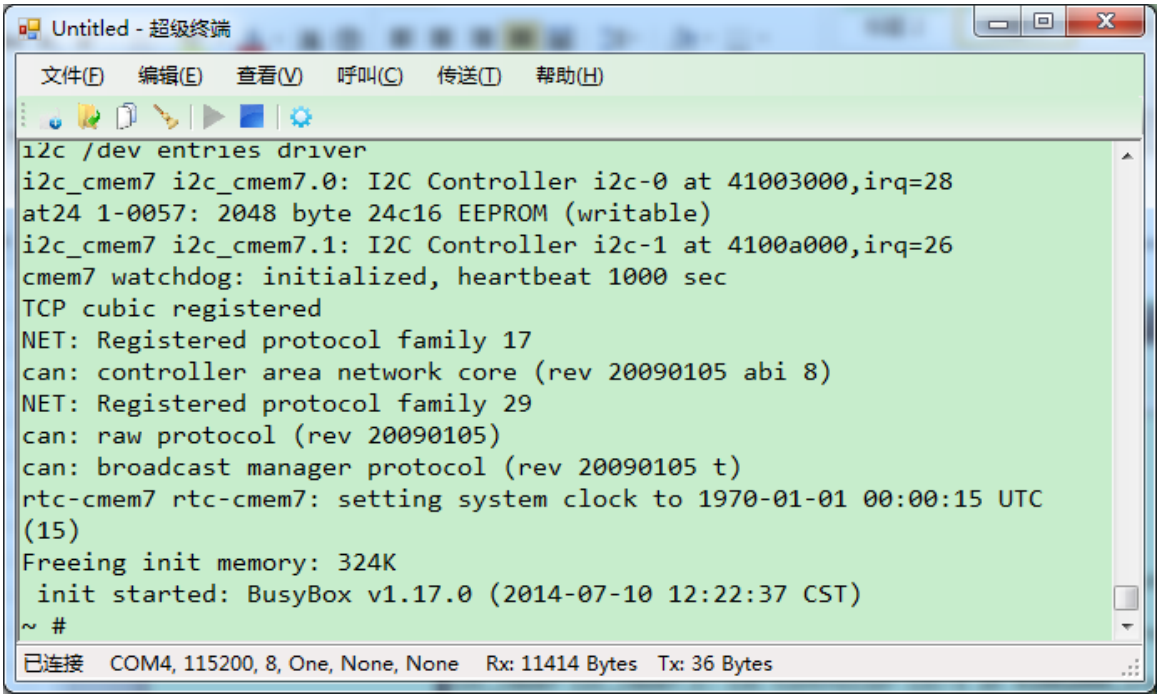

### 2) 效果

a. 当芯片上电后会打印出“Input accurate kernel image size(Byte) :”在串口助手的发送端口写入 image 文件的大小（注意是 iamge.bin 的大小，不是所生成的 image.hex 文件的大小，注：系统启动时，需准确的内核代码字节数，否则内核程序无法启动），按回车完成发送，如下图所示。

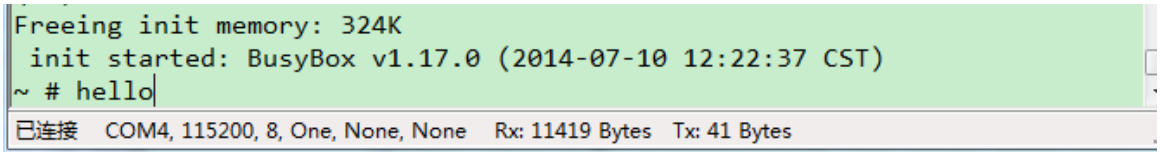



b. 对于出现“Input options that open icache：”询问。在输入窗口输入 0 或者 1，同样附加位也是回车符，点击发送。程序会执行解压 LINUX，引导内核等操作。现象如下图所示。




c. 在命令行输入 hello，效果如下图所示。




执行后：