嵌入式Linux学习笔记

开发板：正点原子阿尔法I.MX6ULL（EMMC版）

**目录**

[1. 学习Cortex-A汇编的目的 2](#_Toc25989)

[2. IO初始化流程 2](#_Toc27379)

[3. 程序编译步骤 2](#_Toc17667)

[4. 烧写代码 3](#_Toc31865)

[5. Ubuntu断网解决方法 3](#_Toc14107)

[6. 启动方式选择 3](#_Toc3669)

[7. 启动设备选择 3](#_Toc13492)

[8. BOOT ROM作用 4](#_Toc14629)

[9. 镜像文件介绍 4](#_Toc4631)

[10. C语言运行环境构建 4](#_Toc11683)

1. **学习Cortex-A汇编的目的**

①初始化一些SOC外设

②初始化DDR（I.MX6U不需要）

③设置sp指针，一般指向DDR，设置好C语言运行的环境

**注**：开发常用汇编：存储器访问指令

假设变量a的地址为0X20，变量b的地址为0X30，将b的值赋值给a，指令如下：

LDR R0, =0X30

LDR R1, [R0]

LDR R0, =0X20

STR R1, [R0]

**LDR**：从存储器读取数据 **STR**：向存储器写入数据

**R0、R1**：ARM汇编指令不能直接从RAM里读数据，所以需要借用寄存器R0、R1等通用寄存器。

1. **IO初始化流程**

**STM32的IO初始化**：

①使能GPIO时钟

②设置IO复用，默认为GPIO

③配置GPIO电气属性

④使用GPIO输出高低电平

**I.MX6U的IO初始化**：

①使能时钟，CCGR0 ~ CCGR6这7个寄存器控制所有外设时钟的使能，为方便起见，设置这七个寄存器全部 为0XFFFFFFFF，即使能所有外设时钟。（见参考手册699页）

②IO复用，点灯为例，GPIO1\_IO03为控制引脚，IOMUXC\_SW\_MUX\_CTL\_PAD\_GPIO1\_IO03为复用寄存器， 设置为0101，此时GPIO1\_IO03就复用为GPIO。（见参考手册1571页）

③配置电气属性，IOMUXC\_SW\_PAD\_CTL\_PAD\_GPIO1\_IO03为设置GPIO1\_IO03电气属性的寄存器，主要 设置：压摆率SRE、驱动能力DSE、速度SPEED、上下拉PUE、开漏ODE等。（见参考手册1793页）

④配置GPIO功能，配置GPIOx\_GDIR寄存器bit3为1，即设置GPIO1\_IO03为输出模式，配置GPIOx\_DR寄 存器bit3，为1表示高电平，为0表示低电平。（见参考手册1357页）

1. **程序编译步骤**

①将.c、.s文件编译为.o文件（使用软件：**arm-linux-guneabihf-gcc**）

如使用指令：arm-linux-guneabihf-gcc -g -c leds.s -o led.o 将led.s文件编译为led.o。

**-g**：产生调试信息 **-c**：编译源文件但不链接 **-o**：指定编译产生的文件名

②将所有的.o文件链接为elf格式的可执行文件（使用软件：**arm-linux-guneabihf-ld**）

如使用指令：arm-linux-guneabihf-ld -Ttext 0X87800000 led.o -o led.elf 将led.o文件链接到0X87800000这个地址上。

**-Ttext**：指定链接地址 **-o**：指定链接生成的elf文件名

**链接**：将众多的.o 文件链接到一个指定的链接位置，即链接起始地址。

**链接起始地址**：代码运行的起始地址。对于I.MX6ULL来说，链接起始地址指向RAM地址，RAM分为内部RAM和 外部RAM（DDR），内部RAM地址为：0X900000 ~ 0X91FFFF，正点原子的阿尔法EMMC版外部RAM地址为： 0X0X80000000 ~ 0X9FFFFFFF。后面要学习的Uboot的链接起始地址是0X87800000，为了不易记混，所以统一 使用DDR的0X87800000作为链接起始地址。

③将elf文件转为bin文件（使用软件：**arm-linux-guneabihf-objcopy**）

如使用指令：arm-linux-guneabihf-objcopy -O binary -S -g led.elf leed.bin 将led.elf文件转换为led.bin 文件。

**-O**：指定以什么格式输出 **binary**：表示以二进制格式输出

**-S**：表示不复制源文件中的重定位信息和符号信息 **-g**：表示不复制源文件中的调试信息

**注**：调试C语言时，可将elf文件转为汇编，即反汇编。（使用软件：**arm-linux-gnueabihf-objdump**）

如使用指令：arm-linux-gnueabihf-objdump -D led.elf > led.dis 将elf文件反汇编为dis文件。

**-D**：表示反汇编所有的段

1. **烧写代码**

I.MX6U支持从外置的NOR Flash、NAND Flash、SD/EMMC、SPI NOR Flash和QSPI Flash这些存储介质中启动，上电以后I.MX6U的内部BOOT ROM程序会将可执行文件拷贝到链接地址处。除了SD卡以外，其他的一般都是焊接到板子上的，无法直接烧写，所以选择将代码烧写到SD卡。烧步骤如下：

①准备工具：正点原子提供的imxdownload软件

②确定要烧写的SD卡：终端输入指令ls /dev/sd\*/ -l查看，如sdb即为SD卡

③给imxdownload软件可执行权限：chmod 777 imxdownload

④开始烧写：./imxdownload led.bin /dev/sdb

⑤烧成成功后将SD卡插到开发板，将开发板上的BOOT\_CFG拨码开关调为SD卡启动（1、7上拨，其余下拨）

**注**：可发现当前文件夹下生成了load.imx文件，因为imxdownload会向led.bin添加一个头部，从而生成了load.imx文件，此 文件才是烧写到SD卡中的文件。

1. **Ubuntu断网解决方法**

①打开终端

②输入：sudo service NetworkManager stop

③输入：sudo gedit /var/lib/NetworkManager/NetworkManager.state

④文件内将 NetworkingEnabled=false 修改为 NetworkingEnabled=true

⑤输入：sudo service NetworkManager start

⑥关闭终端

1. **启动方式选择**

I.MX6U 芯片上电以后，芯片会根据BOOT\_MODE[1:0]的设置来选择BOOT方式。BOOT\_MODE[1:0]的值有两种改变方式，一种是改写 eFUSE(熔丝)，一种是修改相应的GPIO高低电平。第一种修改eFUSE的方式只能修改一次，后面就不能再修改了，所以我们不使用。我们使用的是通过修改 BOOT\_MODE[1:0]对应的GPIO高低电平来选择启动方式，I.MX6U有四个BOOT 模式，00：从FUSE启动、01：串行下载、10：内部BOOT模式、11：保留。只使用第二、三种，介绍如下：

①**串行下载**： BOOT\_MODE1为0，BOOT\_MODE0为1，通过USB或者UART将代码下载到板子上的外置存储设备 中，这个下载是需要用到 NXP 提供的一个软件，一般用来最终量产的时候将代码烧写到外置存储设备中。

②**内部BOOT模式**：BOOT\_MODE1为1，BOOT\_MODE0为0，芯片会执行内部的BOOT ROM代码进行硬件初始化(一 部分外设)，然后从BOOT设备(存放代码的设备、比如SD/EMMC、NAND)中将代码拷贝出来复制到指定的RAM （一般是DDR）中。

1. **启动设备选择**

启动方式设置为内部BOOT模式后，I.MX6U支持从外置的NOR Flash、NAND Flash、SD/EMMC、SPI NOR Flash和QSPI Flash这些存储介质中启动。通过配置BOOT\_CFG1[7:0]、BOOT\_CFG2[7:0]和BOOT\_CFG4[7:0]这24个IO进行设备选择，同样有两种配置方式：改写eFUSE和修改GPIO，我们选择修改GPIO。这24个IO不需要全部配置，开发板中需要配置的IO为： BOOT\_CFG2[3]、BOOT\_CFG1[3] ~ BOOT\_CFG1[7]。加上BOOT\_MODE[1:0]一共8位IO就可完成启动设备的选择，设置方式如下：

**编号**：1：BOOT\_MODE[1] 2：BOOT\_MODE[0] 3：BOOT\_CFG2[3] 4：BOOT\_CFG1[3]

5：BOOT\_CFG1[4] 6：BOOT\_CFG1[5] 7：BOOT\_CFG1[6] 8：BOOT\_CFG1[7]

**设置1 ~ 8**： 01xxxxxx：串行下载，可以通过USB烧写镜像文件

10000010：SD卡启动

10100110：EMMC启动

10001001：NAND FLASH启动

1. **BOOT ROM作用**

①初始化时钟，将内核时钟为396MHz，此外还设置System PLL=528Mhz，USB PLL=480MHz，AHB=132MHz，

IPG=66MHz，后面学习系统时钟会具体介绍。（见参考手册261页）

②使能内存管理单元（MMU）和缓存去提高启动速度。下载镜像时打开L1指令缓存；验证镜像时打开L1数据缓存、 L2缓存和MMU；镜像验证完成时关闭L1数据缓存、L2缓存和MMU。（见参考手册263页）

③从BOOT\_CFG设置的外部存储中读取镜像，然后做相应处理。

1. **镜像文件介绍**

STM32可烧写编译生成的.bin文件，但I.MX6U不能直接烧写.bin文件，需要在.bin文件前面添加一些头部信息，构成

最终可烧写的文件，即.imx文件。.imx文件由四部分组成：IVT + Boot data + DCD + 用户代码及数据。其中ROM将IVT固定在启动设备中一个固定的地址，该地址根据启动设备决定，其余三部分的地址由IVT决定。

①Image vector table（IVT）：镜像向量表，大小为32个字节。包含镜像程序的入口点和Boot Data、DCD地址定位的 信息，BOOT ROM根据这些地址定位信息找到对应部分的存储位置。

②Boot data：启动数据，大小为12个字节。设置镜像数据的大小和要拷贝到的目标地址等。

③Device configuration data（DCD）：设备配置信息，大小不能超过1768字节。BOOT ROM使用DCD中的配置参数 来初始化DDR，配置时钟，以及其他启动时必要的硬件设置。

④用户代码及数据，如led.bin。

启动设备以SD卡为例，初始载荷区域大小为4K字节，即启动时BOOT RAM会拷贝SD卡中前4K字节到片内RAM中，这4K字节中必须包含IVT、Boot data和DCD，BOOT RAM就是根据这4K字节的信息对镜像文件做相应的处理。其中前1K字节固定保存SD卡的分区表信息，是不可以改变的，所以IVT、Boot data和DCD只能存储在后3K字节中。固定IVT存储在最前面，Boot data其次，DCD接在Boot data后面。（见参考手册309页）

1. **C语言运行环境构建**

实际工作中，很少用汇编写驱动，大部分情况下是先使用汇编初始化C语言环境，然后使用C语言编写驱动。汇编初始化C语言环境内容如下：

①**设置I.MX6ULL处于SVC运行模式下**。通过配置CPSR状态寄存器的M[4:0]位为10011，即0X13，配置状态寄存器 需要使用MRS和MSR指令。汇编指令如下：

mrs r0, cpsr /\* 读取cpsr到r0 \*/

bic r0, r0, #0x1f /\* 清除cpsr的M[4:0]位 \*/

orr r0, r0, #0x13 /\* 选择SVC模式 \*/

msr cpsr, r0 /\* 将r0写入到cpsr中，进入SVC模式\*/

BIC Rd, Rn, #immed @Rd = Rn & (~#immed) 位清除

ORR Rd, Rn, #immed @Rd = Rn | #immed 按位或

MRS R0, CPSR @将特殊寄存器CPSR里面的数据传递给R0，即 R0=CPSR

MSR CPSR, R0 @将R0中的数据复制到CPSR中，即 CPSR=R0

②**设置sp堆栈指针**。我们设置指针指向DDR，栈大小设置为0X200000 = 2MB。Cortex-A7 架构中栈的增长方式为向 下增长（从高地址向低地址增长），本开发板DDR范围为0X80000000 ~ 0X9FFFFFFF，所以设置sp指针指向 0X80200000。汇编指令为：ldr sp, =0x80200000

③**跳转到C语言main函数**。汇编指令为：b main

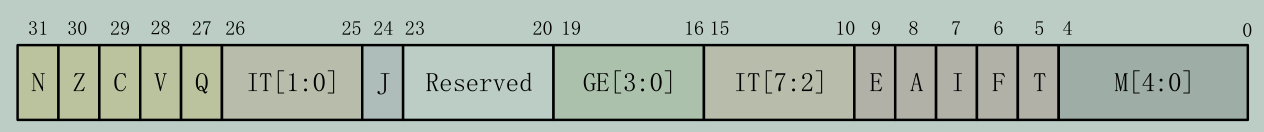
**注**：①**运行模式**

I.MX6ULL使用的是Cortex-A7 架构，Cortex-A7架构的一个优点是功耗低。Cortex-A有9种运行模式，介绍如下。其中除了用户模式外，其他8种模式均为特权模式，可由软件任意切换，也可通过中断或异常进行切换；用户模式只能通过中断或异常进行模式切换，用户模式下不能访问系统的所有资源，有些资源是受限的。

1. User(USR)：用户模式
2. FIQ：中断模式，用于进入FIQ中断异常
3. IRQ：一般中断模式
4. Supervisor(SVC)：超级管理员模式，供操作系统使用
5. Monitor(MON)：监视模式，用于安全扩展
6. Abort(ABT) ：数据访问终止模式，用于虚拟存储以及存储保护
7. Hyp(HYP)：超级监视模式，用于虚拟化扩展
8. Undef(UND)：未定义指令终止模式
9. System(SYS) ：系统模式，用于运行特权级的操作系统任务

②**CPSR寄存器**

当前程序状态寄存器，该寄存器包含了条件标志位、中断禁止位、当前处理器模式标志等一些状态位以及一些控制位。结构图如下，其中M[4:0]为处理器模式控制位，配置信息如下：



10000：User模式 10001：FIQ模式 10010：IRQ模式

10011：Supervisor(SVC)模式 10110：Monitor(MON)模式 10111：Abort(ABT)模式

11010：Hyp(HYP)模式 11011：Undef(UND)模式 11111：System(SYS)模式

③读取特殊寄存器（如CPSR和SPSR）的数据时只能使用MRS和MSR这两个指令。

④有的芯片在设置sp指针之前需要初始化DDR，对于本芯片，在生成的.imx文件中的头部信息中已经完成了DDR的 初始化，所以这里直接设置sp指针就可以了。