TMS320F28377D 是 TI 德州仪器公司 C2000 系列的一款双核的芯片，其用法与 TMS320F28335 接近，但主频，外设都有进一步提高与增强，其中双核 CPU 架构更增强了其处理能力。那我们如何对双核 CPU 进行程序编写、仿真、下载等操作呢？我们将详细介绍 TMS320F28377D 双核的程序编写、仿真、下载。

TMS320F28377D 的双核架构 CPU1 和 CPU2 它们有各自的独立内核、内存、中断、总线等，但可以共享其余的外设，两个 CPU 可以独立运行自己的程序，也可以通过“处理器通信模块（IPC）”进行数据的交互。因此，对 TMS320F28377D 程序编写、仿真、下载等操作，就是分别对 CPU1、CPU2 进行相关操作，具体操作下文详细叙述。

1.实验目的

(1) 掌握 TMS320F28377D 的双核程序编写方法；

(2) 掌握 TMS320F28377D 的双核程序仿真运行方法；

(3) 掌握 TMS320F28377D 的双核程序下载运行方法；

2. 实验器材

(1) YXDSP-F28377D 开发板一个；

(2) YXDSP-XDS110 仿真器一个；

(3) 装有 CCS9.2 的电脑一台；

3. 实验步骤

双核程序编写

双核程序导入

双核仿真配置与编译

双核程序 RAM 下载与运行（在线仿真运行）

双核程序 FLASH 下载与运行（离线下载与运行）

3.1 双核程序的编写

双核程序的编写相较于单核来说，主要是多了一段引导程序以及对 CPU2 外设的配置。在 CPU1 的程序中，需要在主函数前加上以下程序进行预定义处理：

```
#ifdef _STANDALONE
#ifdef _FLASH
IPCBootCPU2(C1C2_BROM_BOOTMODE_BOOT_FROM_FLASH);
#else
IPCBootCPU2(C1C2_BROM_BOOTMODE_BOOT_FROM_RAM);
#endif
#endif
```

这段预定义程序和前面 Predefined_Name 的设置相照应，引导程序从 RAM 运行或者从 FLASH 中运行。并且在离线时，用 CPU1 调动 CPU2 的启动。

正常的初始化后，对 IO 进行配置：

```
GPIO_SetupPinOptions(0, GPIO_OUTPUT, GPIO_PUSHPULL);
GPIO_SetupPinOptions(1, GPIO_OUTPUT, GPIO_PUSHPULL);
GPIO_SetupPinMux(1, GPIO_MUX_CPU2, 0);
GPIO_SetupPinMux(0, GPIO_MUX_CPU1, 0);
```

值得注意的是，并不是某个 IO 需要由 CPU2 控制，就一定要在 CPU2 的程序里配置。在 CPU1 的 GPIO_SetupPinMux 库函数中，直接设置 IO0 由 CPU1 控制，IO1 由 CPU2 控制，且都作为 GPIO 功能。在 GPIO_SetupPinOptions 函数中，我们分别设置两个 GPIO 作为输出使用。经过初始化配置后，在后续程序 for 循环中对 GPIO0 置高和置低，并且加上相应的延时函数即可控制 LED1 的亮灭。

```
GpioDataRegs.GPADAT.bit.GPIO0 = 0;
DELAY_US(1000 * 500);
GpioDataRegs.GPADAT.bit.GPIO0 = 1;
DELAY_US(1000 * 500);
```

对于 CPU2 程序的编写，需要另起程序文件，在主函数中加上一段 FLASH 初始化函数：

```
#ifdef _FLASH
 InitFlash();
```

GPIO1 已经在 CPU1 程序中进行了配置。在初始化后，在 CPU2 的程序中，例如 for 循环中，对 GPIO1 置高和置低，并且加上相应的延时函数即可控制 LED2 的亮灭。

```
GpioDataRegs.GPADAT.bit.GPIO1 = 0;
DELAY_US(1000 * 500);
GpioDataRegs.GPADAT.bit.GPIO1 = 1;
DELAY_US(1000 * 500);
```

具体的程序文件参考例程中 lab01_led_cpu1，lab01_led_cpu2 程序文

件。

3.2 双核程序导入

双核烧写是分别对两个核烧写独立的程序。首先创建或者导入要烧写的双核程序。这里导入了 CPU1 烧写的程序“lab01_led_cpu1”和 CPU2 烧写的程序“lab01_led_cpu2”。如图 3.1 所示：

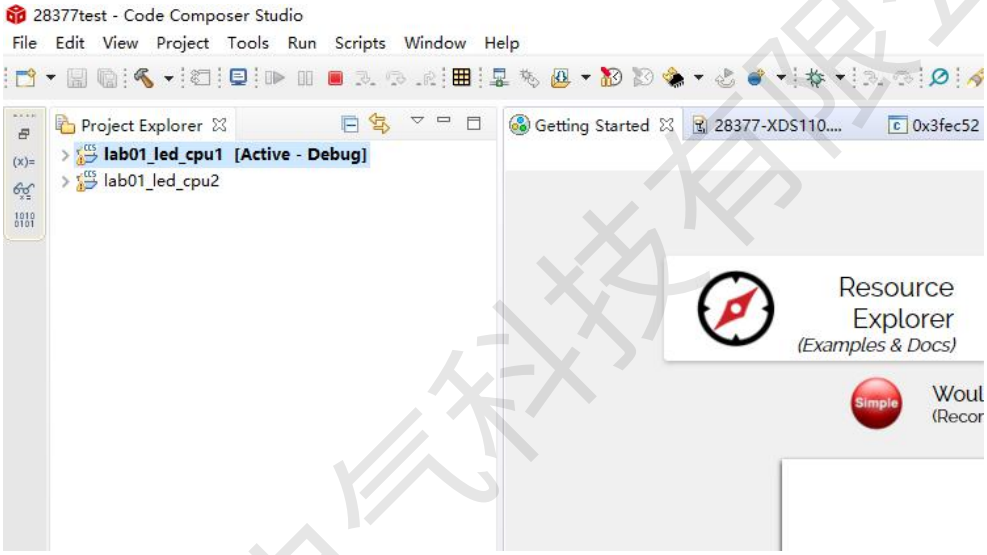

图 3.1 双核例程导入

3.3 双核仿真配置

CPU1 的仿真配置

右键 CPU1 的工程，在属性（properties）里，我们找到高级选项（Advanced Options）菜单下的 预定义符号（Predefined Symbols）界面。在预定义名称（Predefined Name）菜单栏下，只定义 CPU1。如果此时 Pre-defined Name 栏下没有定义 CPU1 选项或者有其它的预定义（如_FLASH 或者 STANDALONE），通过右边的添加或者删除按钮自行修改。最终设置如图 3.2 所示

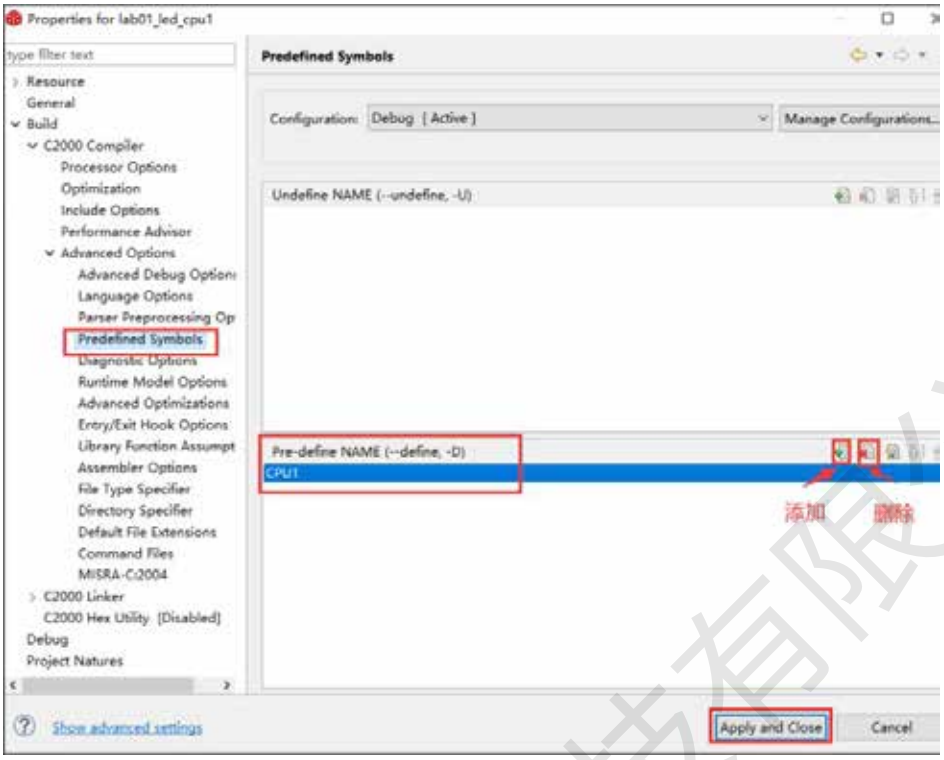

图 3.2 CPU1 仿真设置

设置完成后，选择 Apply and Close，在 CPU1 工程的 CMD 文件下，屏蔽_FLASH 的 CMD，如图 3.3 所示，然后编译工程。

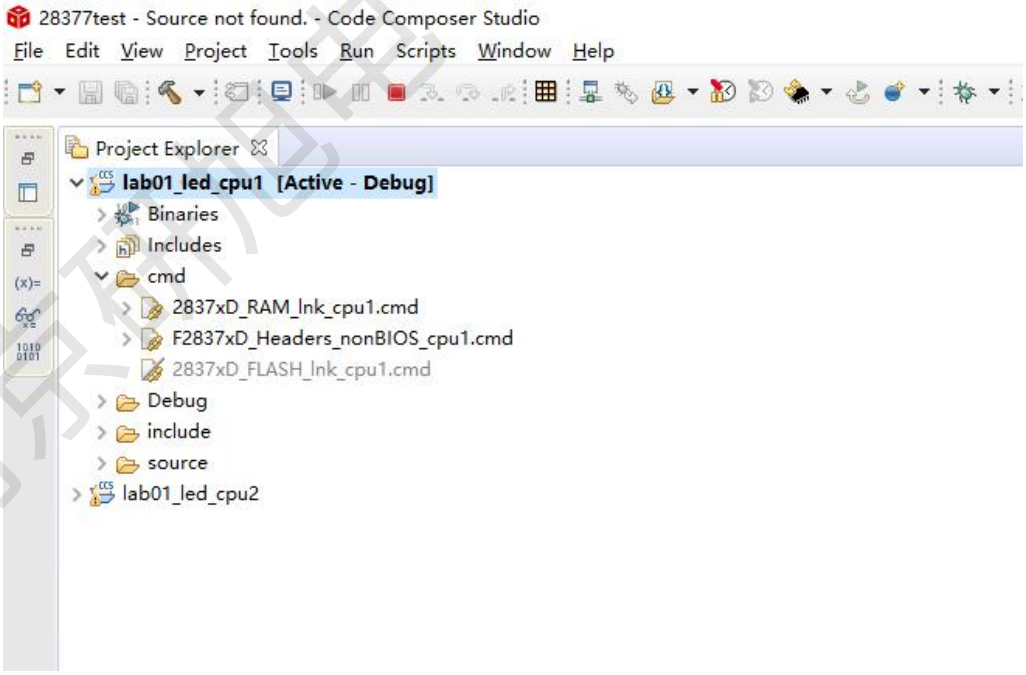

图 3.3 CPU1 CMD 文件设置

CPU2 的仿真配置

CPU2 的仿真配置与 CPU1 类似，右键 CPU2 的工程，在 properties 里，我们找到 Advanced Options 菜单下的 Predefined Symbols 界面。在 Pre-defined Name 菜单栏下，只定义 CPU2。如果此时 Pre-defined Name 栏下没有定义 CPU2 选项或者有其它的预定义（如_FLASH），通过右边的添加或者删除按钮自行修改。最终设置如图 3.4 所示。

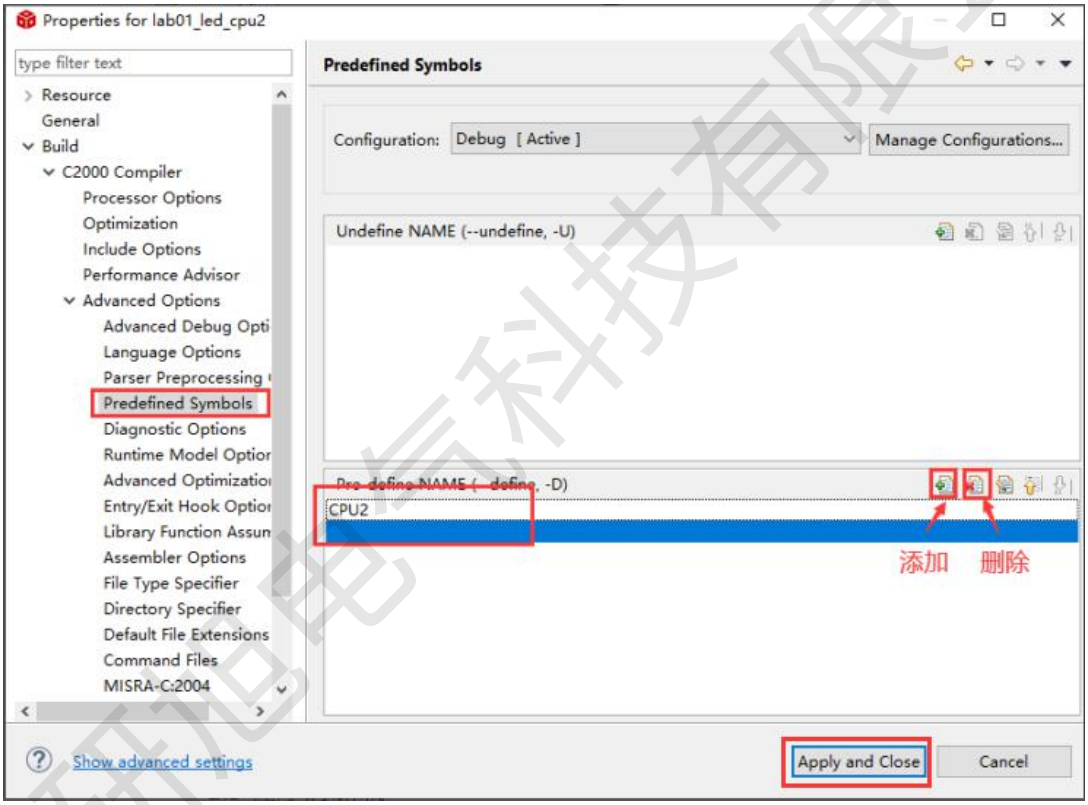

图 3.4 CPU2 仿真设置

设置完成后，选择 Apply and Close，在 CPU2 的 CMD 文件夹下，屏蔽_FLASH 的 CMD 文件，如图 3.5 所示。然后编译工程。

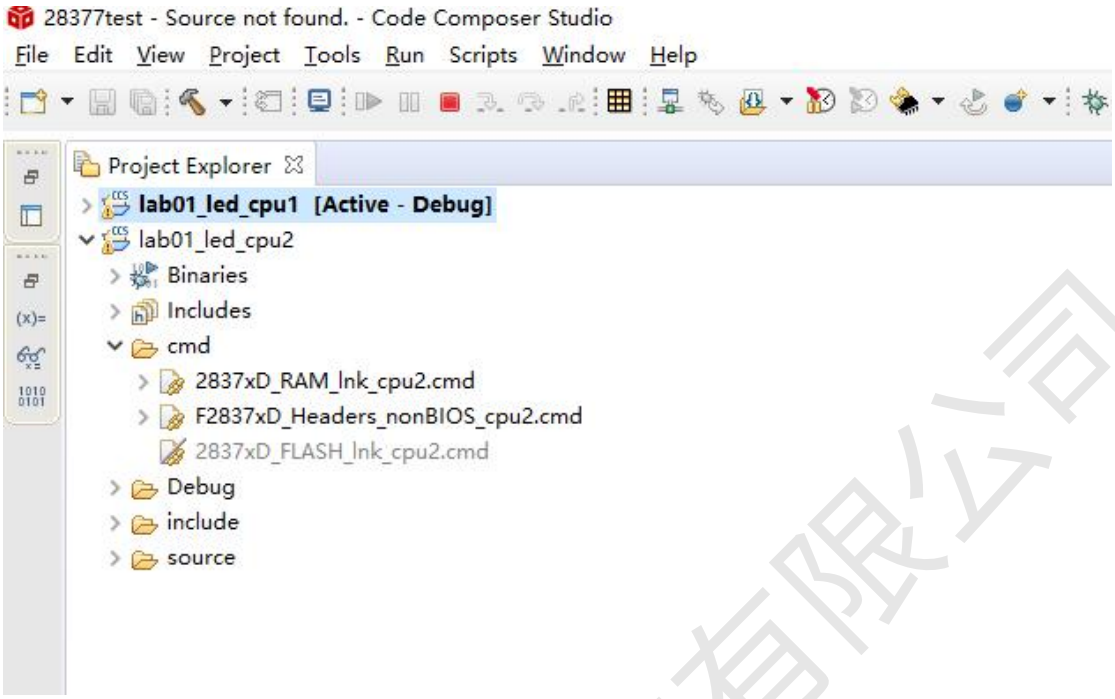

图 3.5 CPU2 CMD 配置

3.4 双核 RAM 加载与运行

首先，确保自行设置的连接文件配置是正确的。本例程使用的是 F28377 搭配 XDS110 仿真器。Launch 对应的.ccxml 文件，此时观察 DEBUG 窗口，如图 3.6 所示。有 4 个连接选项，分别为 CPU1、CLA1、CPU2、CLA2。此时它们的状态均为 Dsiconnected。

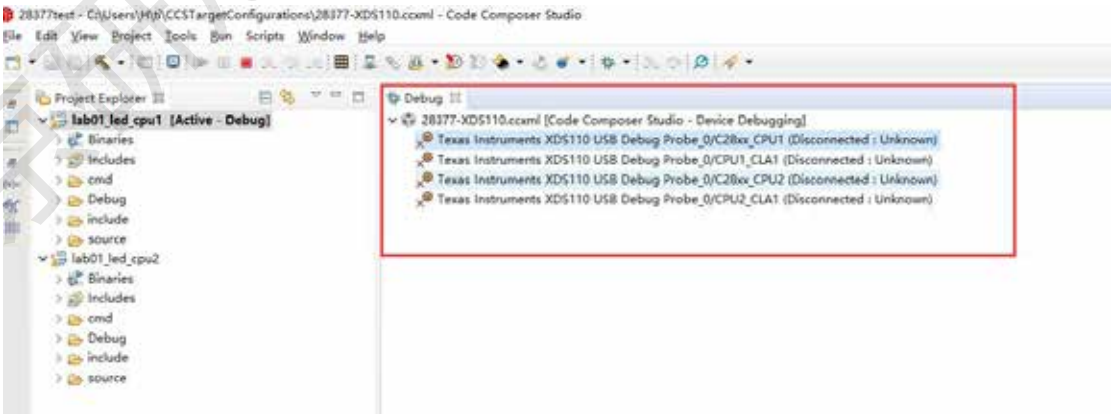

图 3.6 DUBG 窗口

鼠标选中 DEBUG 界面下的 CPU1，然后点击连接按钮，如图 3.7 所示：

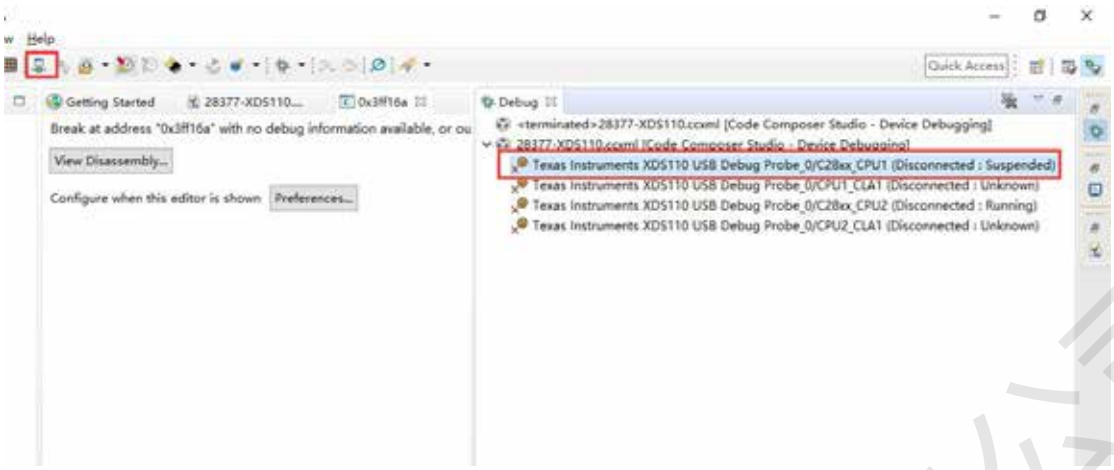

图 3.7 CPU1 的连接

烧写 CPU1 的程序，烧写完成后，不要点击运行。打开 DEBUG 界面，此时 CPU1 的状态如图 3.8 所示：

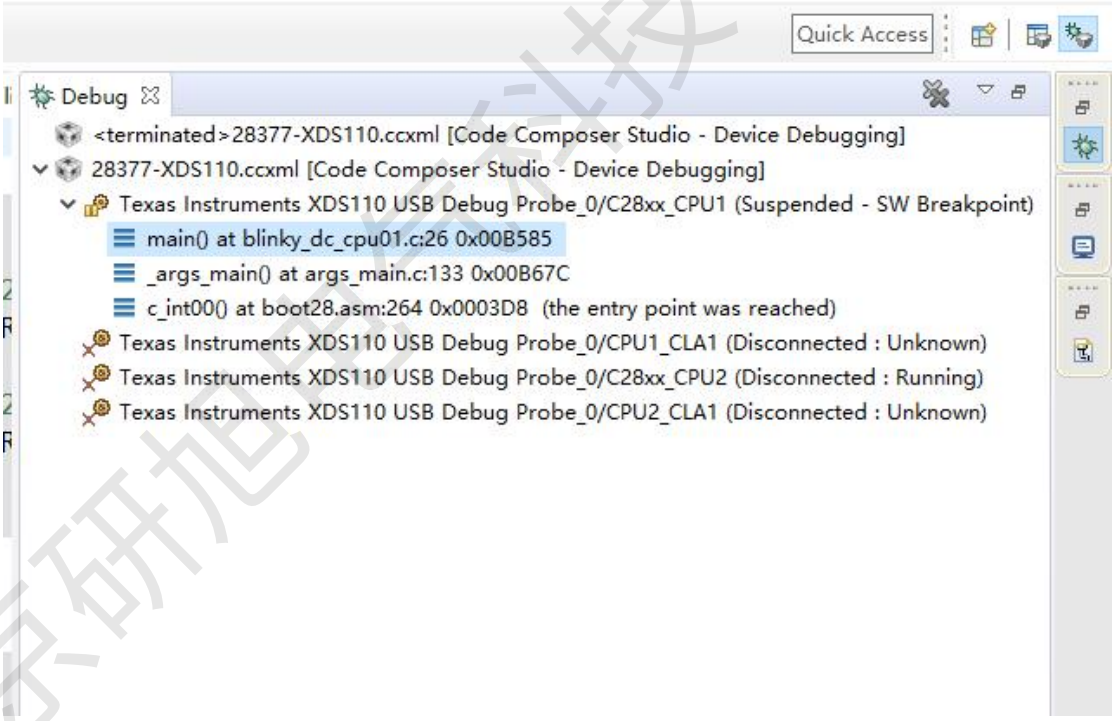

图 3.8 CPU1 的烧写

完成 CPU1 的烧写后，鼠标再次选中 DEBUG 窗口下的 CPU2,然后点击连接按钮，如图 3.9 所示。

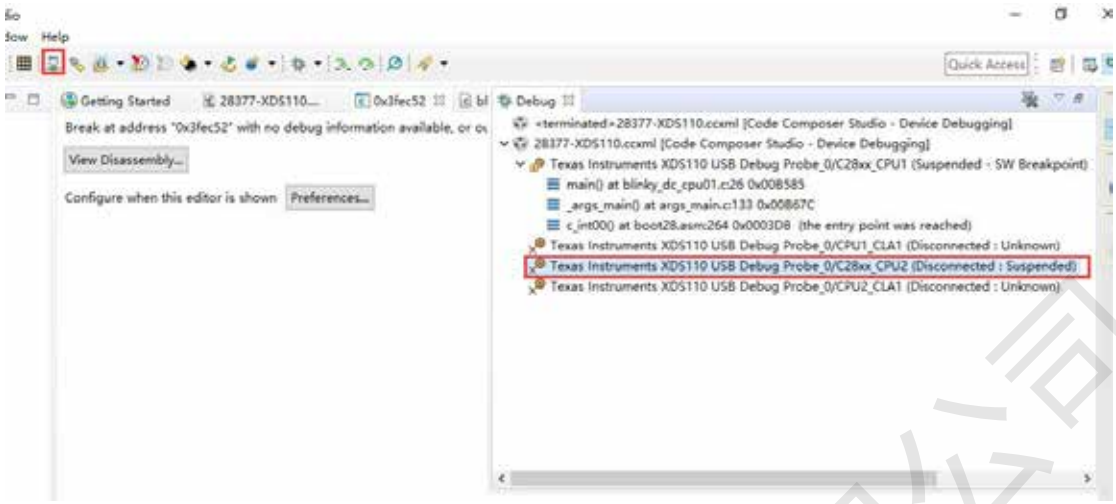

图 3.9 CPU2 的仿真烧写

CPU2 连接后，加载 CPU2 的程序（注意选择正确的程序），烧写完成后的 DEBUG 界面如图 3.10 所示：

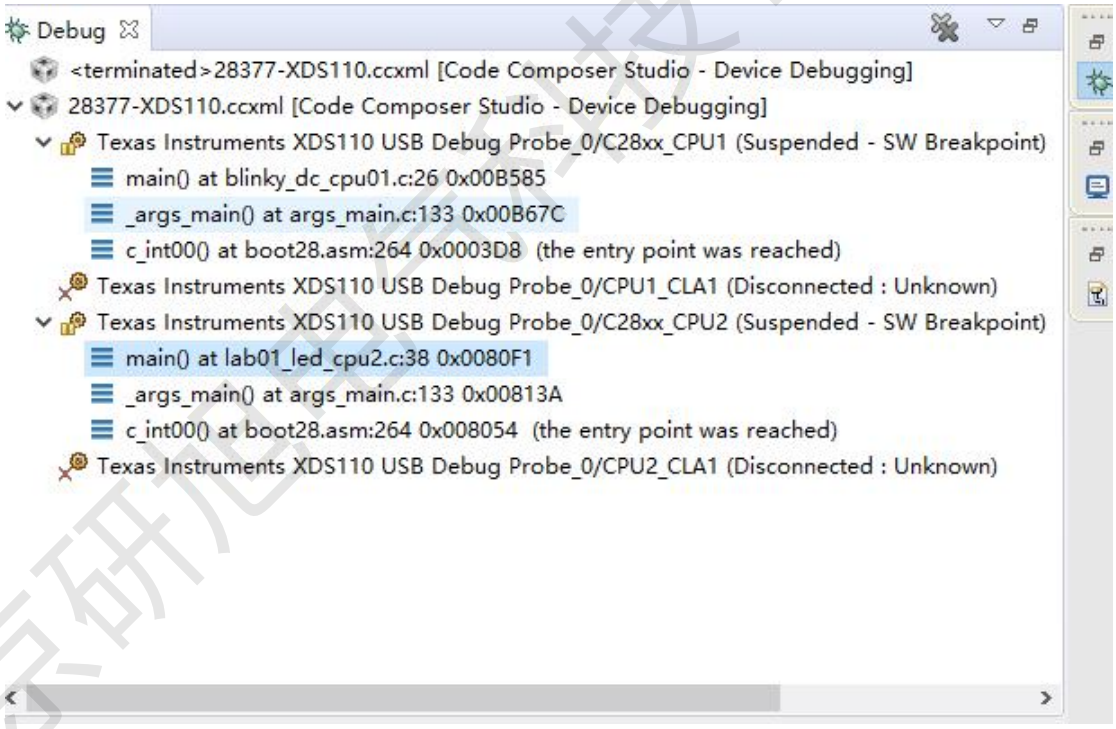

图 3.10 CPU2 的烧写

此时鼠标放在 DEBUG 界面的 CPU1 下，点击运行。如图 3.11 所示，在程序里，配置了 CPU1 来控制 D1，此时观察到开发板的 D1 开始闪烁。

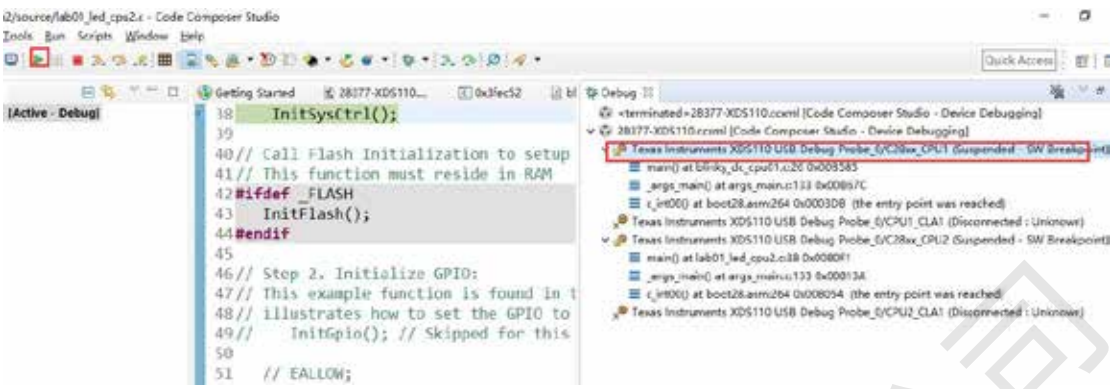

图 3.11 CPU1 运行

然后鼠标再放在 DEBUG 界面的 CPU2 下，点击运行，如图 3.12，在程序里，配置了 CPU2 来控制 D2,此时观察到开发板的 D2 也开始闪烁。

图 3.12 CPU2 运行

当 CPU1 和 CPU2 程序都加载完成后，此时 DEBUG 界面如图 3.13 所示：双核均处于运行状态。

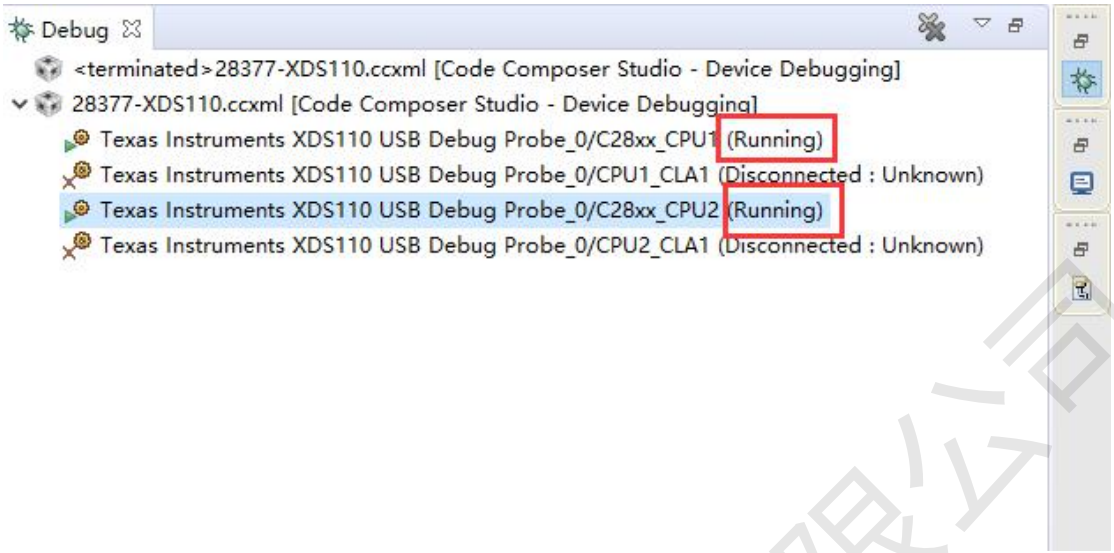

图 3.13 28377D 双核运行状态

至此，28377D 双核的仿真结束。

其余仿真器的 RAM 加载与运行过程是类似的，在此不赘述。

3.5 双核离线配置

CPU1 的烧写配置

双核的离线烧写与仿真大致相同，右键 CPU1 的工程，在 properties 里，找到 Advanced Options 菜单下的 Predefined Symbols 界面。在 Pre-defined Name 菜单栏下，如果这里只定义了 CPU1。选择右边的添加按钮，添加_FLASH 和_STANDALONE，完成后如图 3.14 所示：

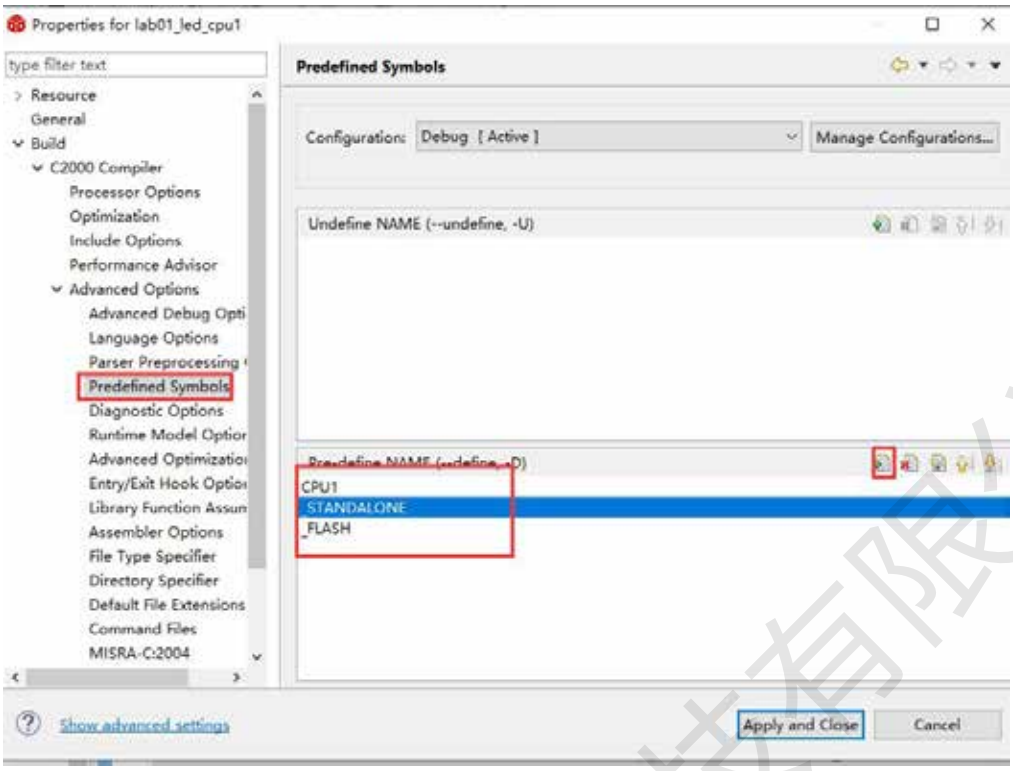

图 3.14 CPU1 的离线配置

设置完成后，选择 Apply and Close，在 CPU1 工程的 CMD 文件夹下，屏蔽_RAM 的 CMD，如图 3.15 所示，然后编译工程。

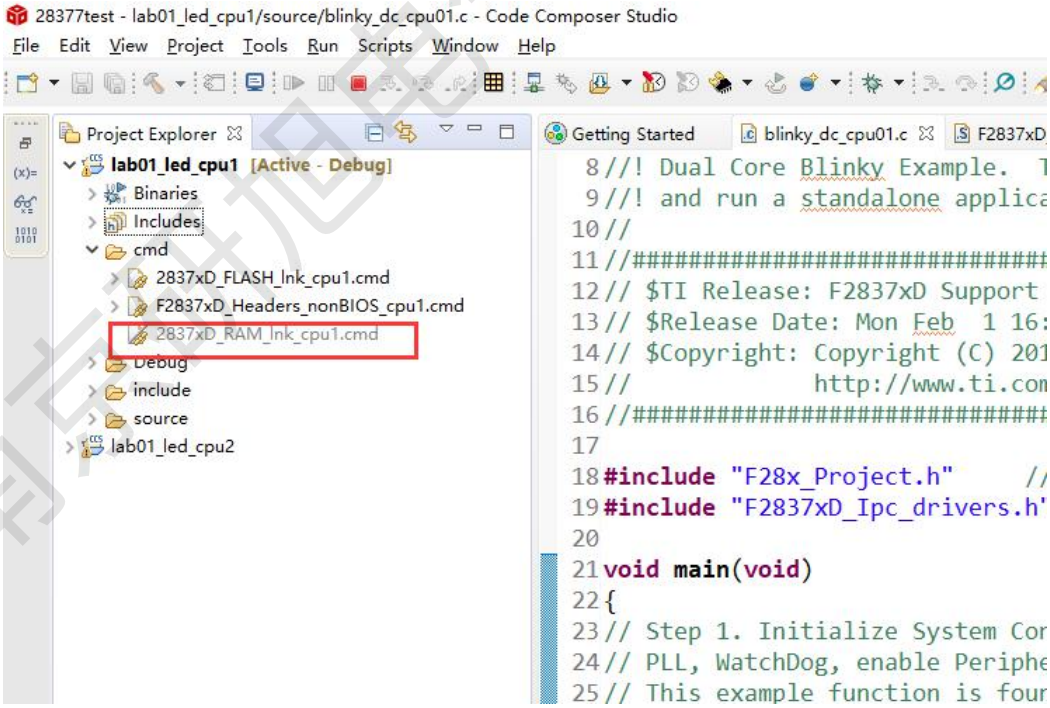

图 3.15 CPU1 的 CMD 设置

CPU2 的烧写配置

CPU2 的仿真配置与 CPU1 类似，右键 CPU2 的工程，在 properties 里，找到 Advanced Options 菜单下的 Predefined Symbols 界面。在 Pre-defined Name 菜单栏下，如果这里只定义了 CPU2，选择右边的添加按钮，添加_FLASH。最终设置如图 3.16 所示。

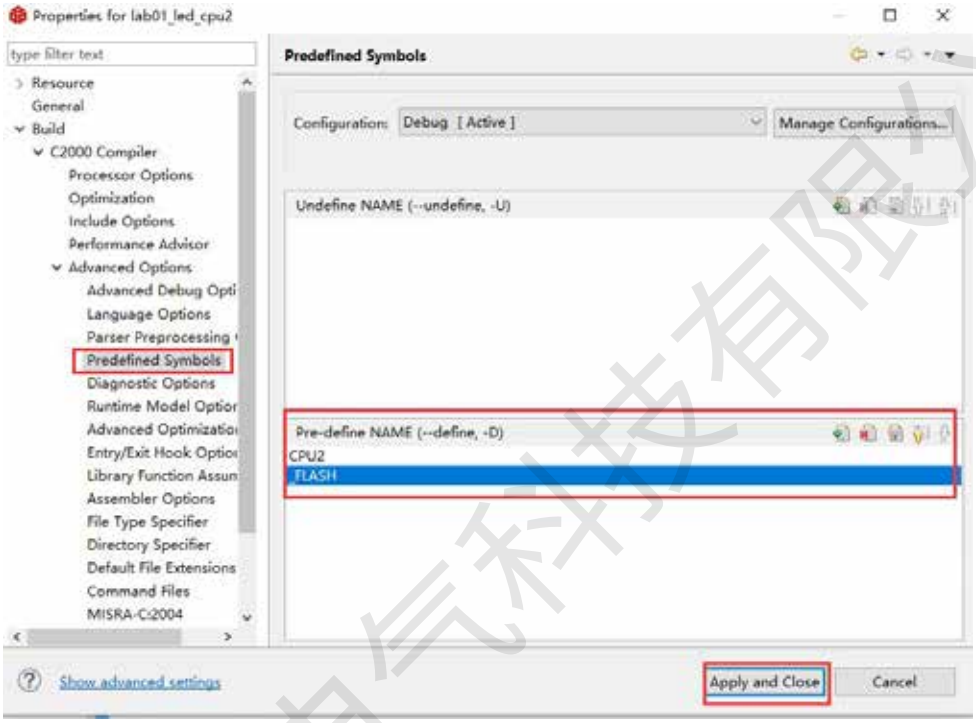

图 3.16 CPU2 配置

设置完成后，选择 Apply and Close，在 CPU2 工程的 CMD 文件夹下，屏蔽_RAM 的 CMD，如图 3.17 所示，然后编译工程。

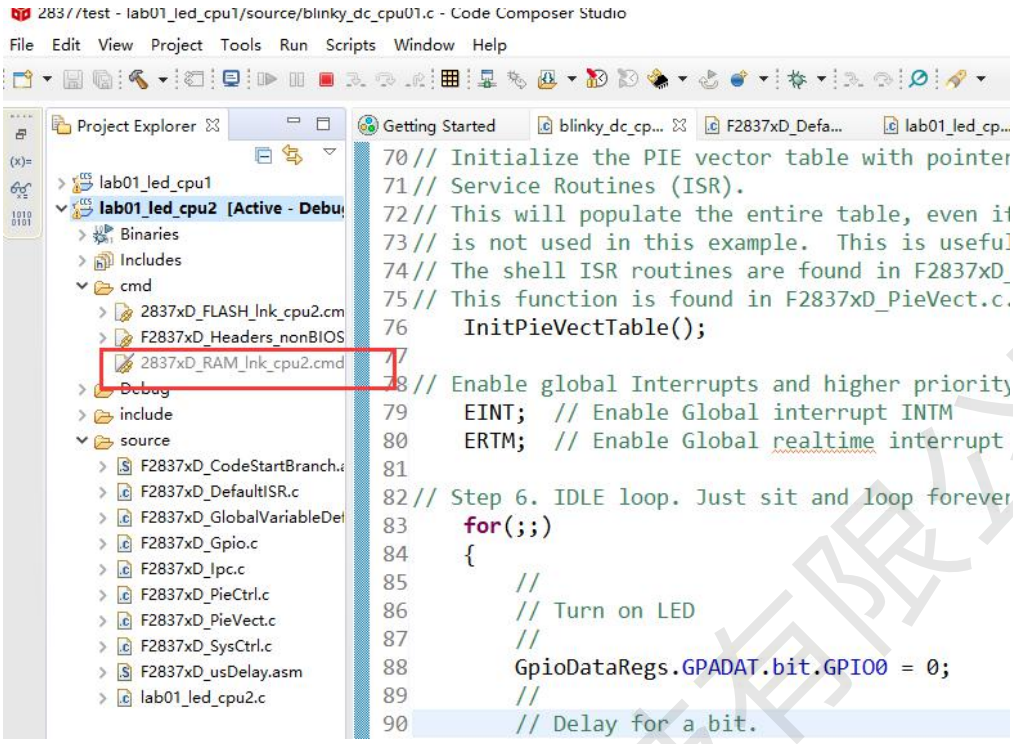

图 3.17 CPU2 的 CMD 设置

3.6 双核 FALSH 的加载与运行

首先，确定自行建立的连接文件配置是正确的。例程中使用的是 F28377 搭配 XDS110 仿真器。Launch 自己的.ccxml 文件，此时观察 DEBUG 窗口，如图 3.18 所示。有 4 个连接选项，分别为 CPU1、CLA1、CPU2、CLA2。此时它们的状态均为 Dsiconnected。

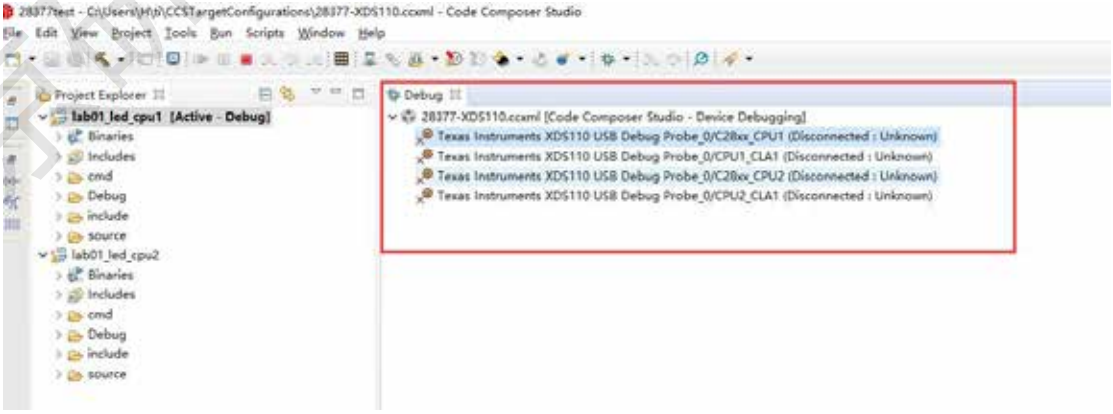

图 3.18 DEBUG 窗口

我们鼠标选中 DEBUG 界面下的 CPU1，然后点击连接按钮，如图

3.19 所示：

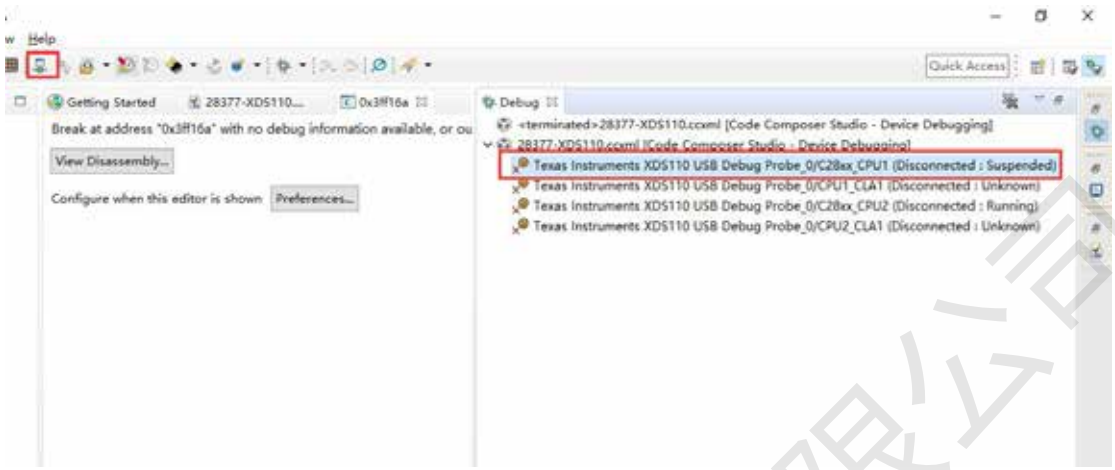

图 3.19 CPU1 的连接

烧写 CU1 的程序，烧写完成后，不要点击运行。打开 DEBUG 界面，此时 CPU1 的状态如图 3.20 所示：

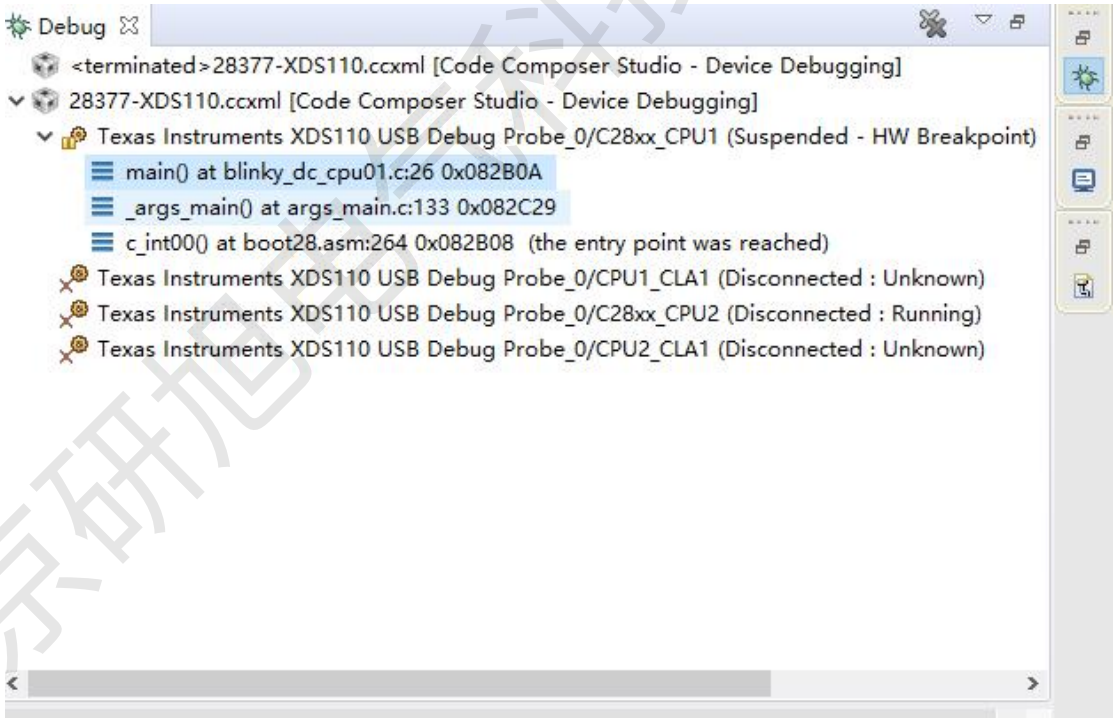

图 3.20 CPU1 的烧写

完成 CPU1 的烧写后，我们鼠标再次选中 DEBUG 窗口下的 CPU2，然后点击连接按钮，如图 3.21 所示：

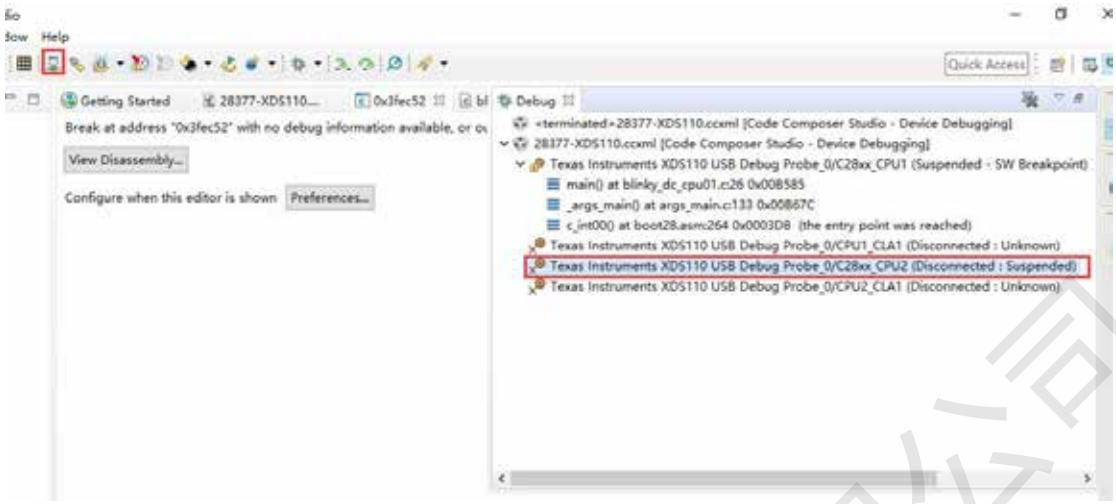

图 3.21 CPU2 的连接

CPU2 连接后，我们加载 CPU2 的程序（注意选择正确的程序），烧写完成后的 DEBUG 界面如图 3.22 所示：

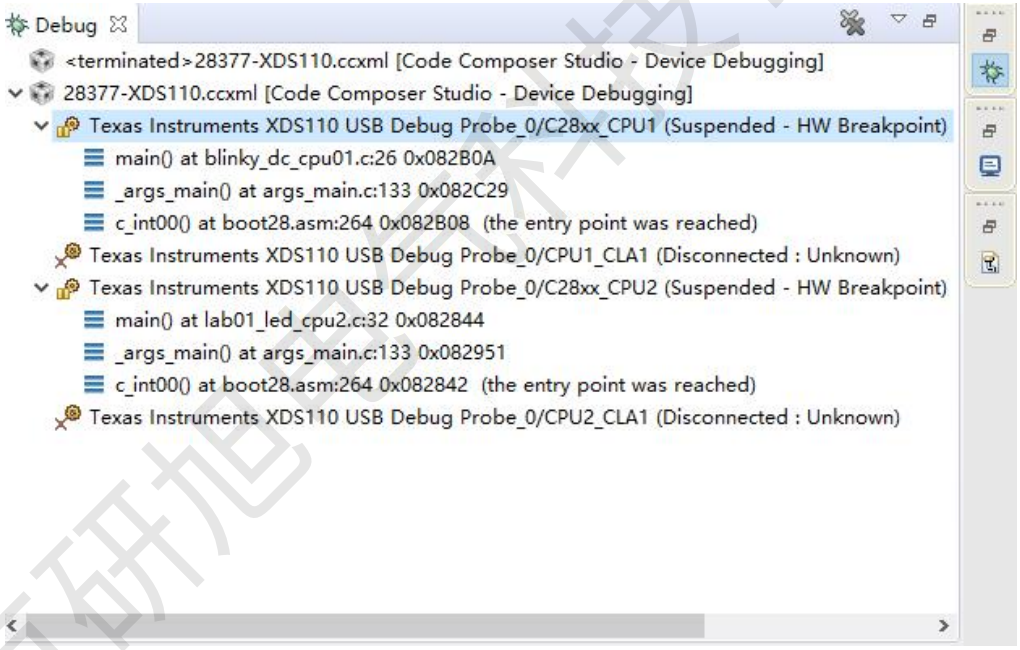

图 3.22 CPU2 的烧写

完成 CPU1 和 CPU2 的烧写后，分别运行 CPU1 和 CPU2 的程序，此时开发板并没有像仿真时有 LED 闪烁的现象。因为是脱机烧写，程序在 FLASH 中，程序中 FLASH 启动，需要重新引导，断开仿真器连接，给开发板重新上电，此时观察到 D1 与 D2 闪烁，CPU1 与 CPU2 程序开始运行。

4.实验现象

当完成 CPU1 和 CPU2 的仿真烧写后，我们点击运行 CPU1，可观察到开发板 D1 闪烁，如图 4.1 所示：

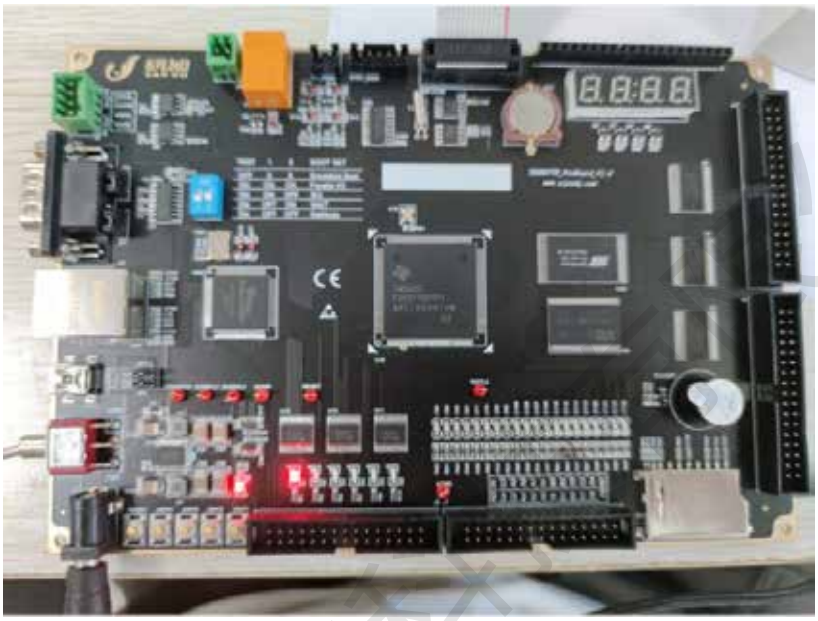

图 4.1 CPU1 运行现象

接着，我们再点击运行 CPU2,可观察到开发板上 D2 也开始闪烁，如图 4.2 所示：

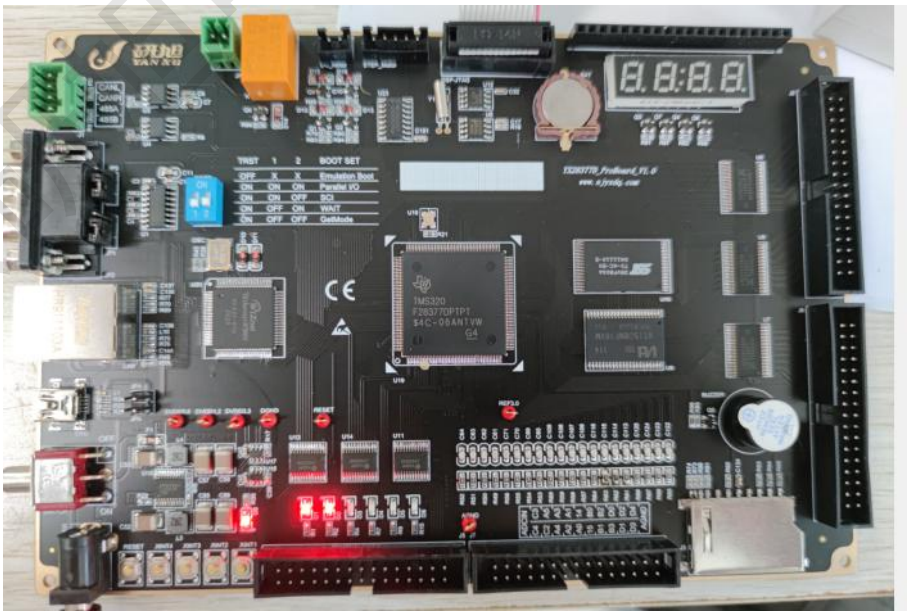

图 4.2 CPU2 运行现象

对于离线的烧写，则需要断电重启后可看到同样的现象。