1 Altera 串口在线升级操作说明

我们在工程中将 FLASH 主要分为两部分：0x000000~0x3BFFFF 叫做 bootloader 区，用于存放引导程序，0x400000~0xFFFFFF 叫做 APP 用户代码区，工程支持存放两个用户工程代码，分为 APP1、APP2，APP1 的起始地址为 0x400000，APP2 的起始地址需要我们通过串口指令进行指定。通过 bootloader 区存放的引导程序，对用户代码区的程序进行更新，接下来我们将对如何进行更新以及需要发送的指令进行说明。（程序基于小梅哥 AC108 开发板进行开发，可直接使用 AC108 进行验证，如果想用于自己的硬件，需要自行修改代码）

1.1 只需要 1 个 APP 区

很多情况下，用户只需要一个 APP 区便可以达到自己需要实现的功能，下面我们将针对只需要一个 APP 区的用户进行说明。

1.1.1 合成我们需要的 jic 文件

打开我们提供的 Factory 工程，依次点击 File->Convert Programming Files...，操作如下图 1-1 所示。

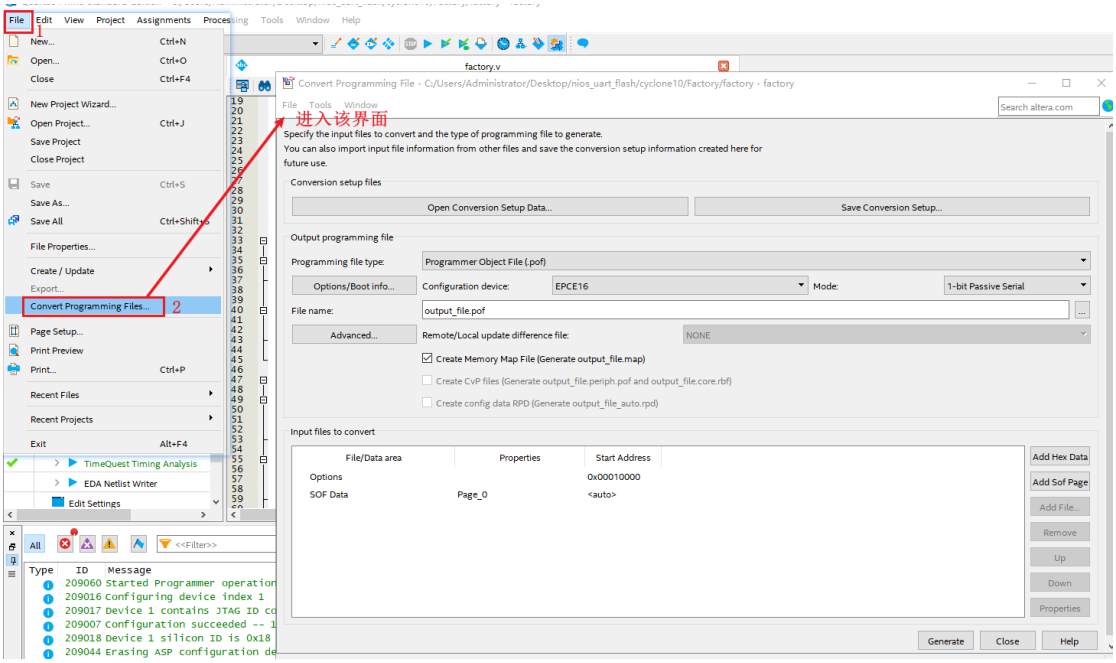

图 1-1 进入 Convert Programming Files 界面

然后是 Output programming file 一栏的配置。在 Programming file type 一栏

中选择 JTAG Indirect Configuration File (.jic)，Configuration device 一栏选择 EPCQ128，File name 可以根据自己的需要进行修改，我们这里将其命名为 factory.jic，如下图 1-2 所示。

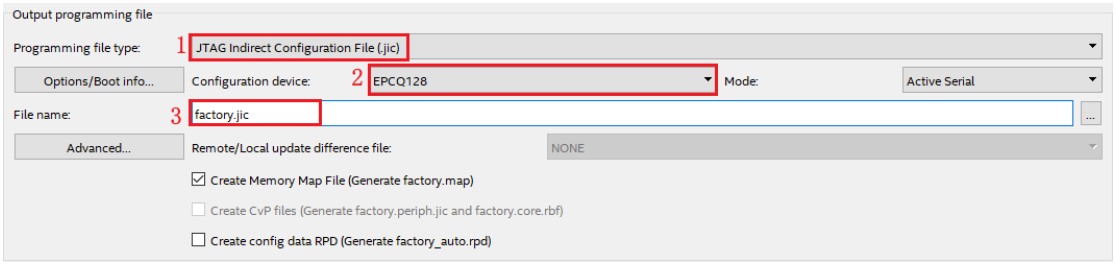

图 1-2 Output programming file 一栏配置

最后是 Input files to convert 一栏的配置。

（1）依次点击 Flash Loader->Add Device->Cyclone 10 LP->10CL025Y，如下图 1-3 所示。

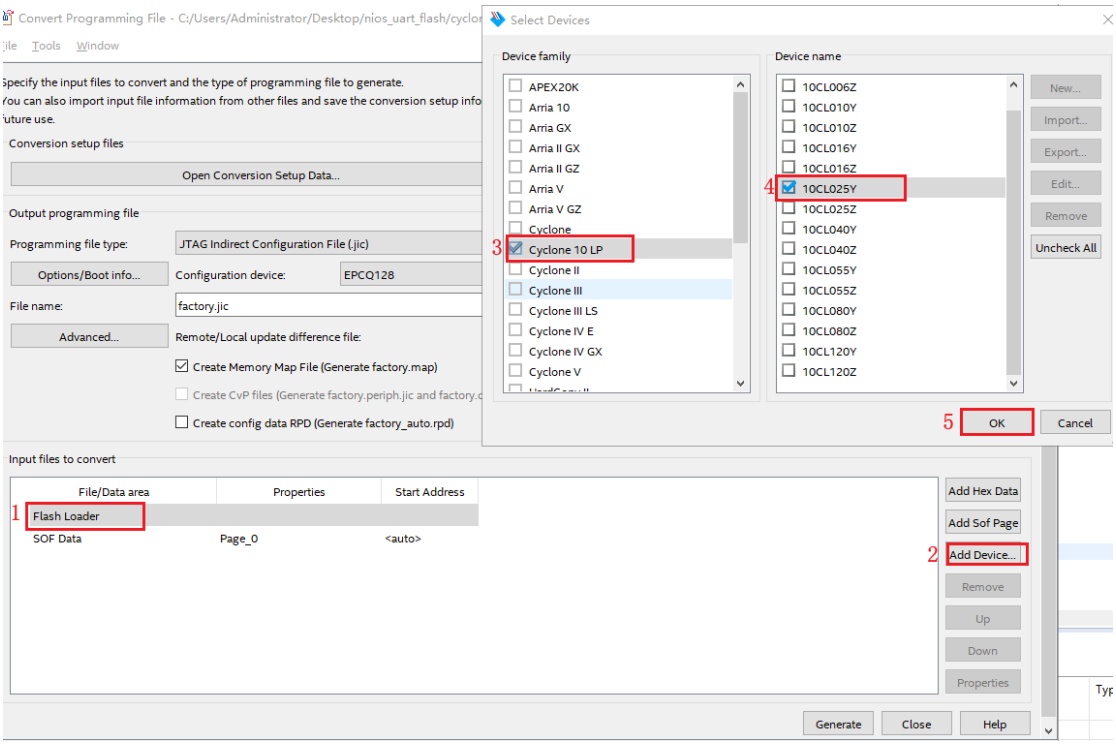

图 1-3 配置 Flash Loader

（2）点击 SOF Data->Properties->Block->End address->0x3BFFFF，用于存放 Factory 工程生成的 sof 文件，如下图 1-4 所示。

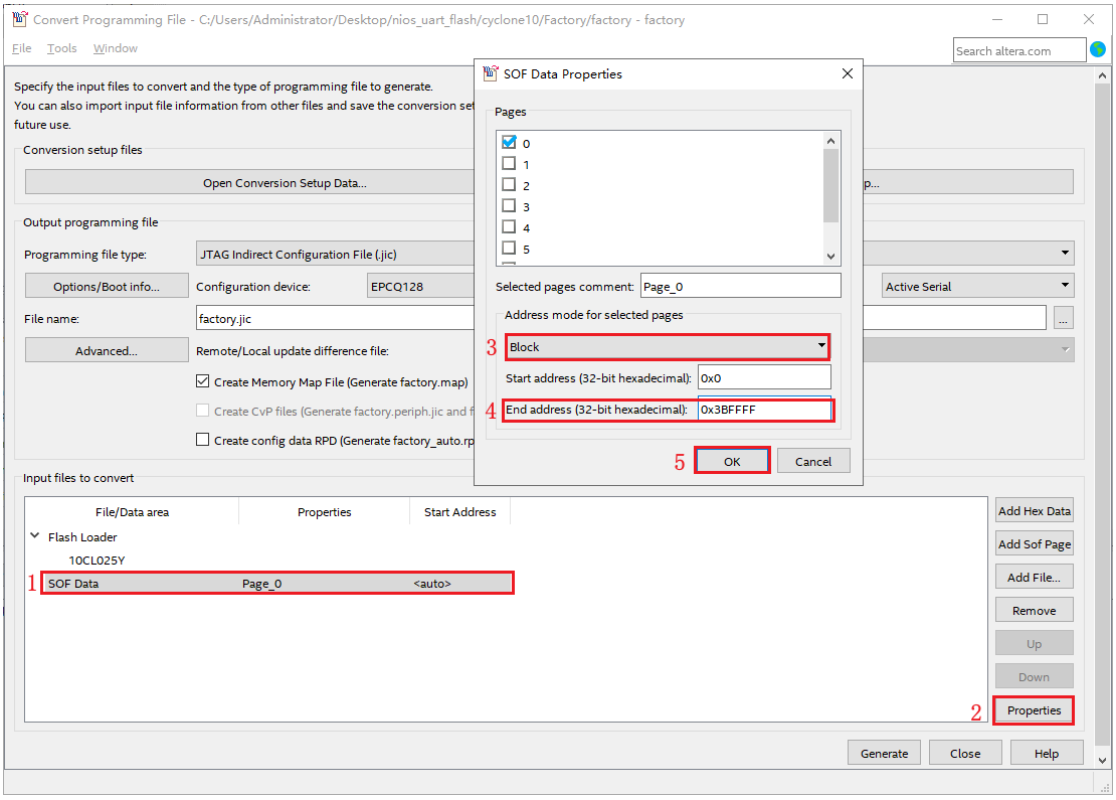

图 1-4 分配 Factory 工程 SOF 文件的起始地址

（3）点击 SOF Data->Add File，添加 Factory 工程生成的 SOF 文件，如下图 1-5 所示。

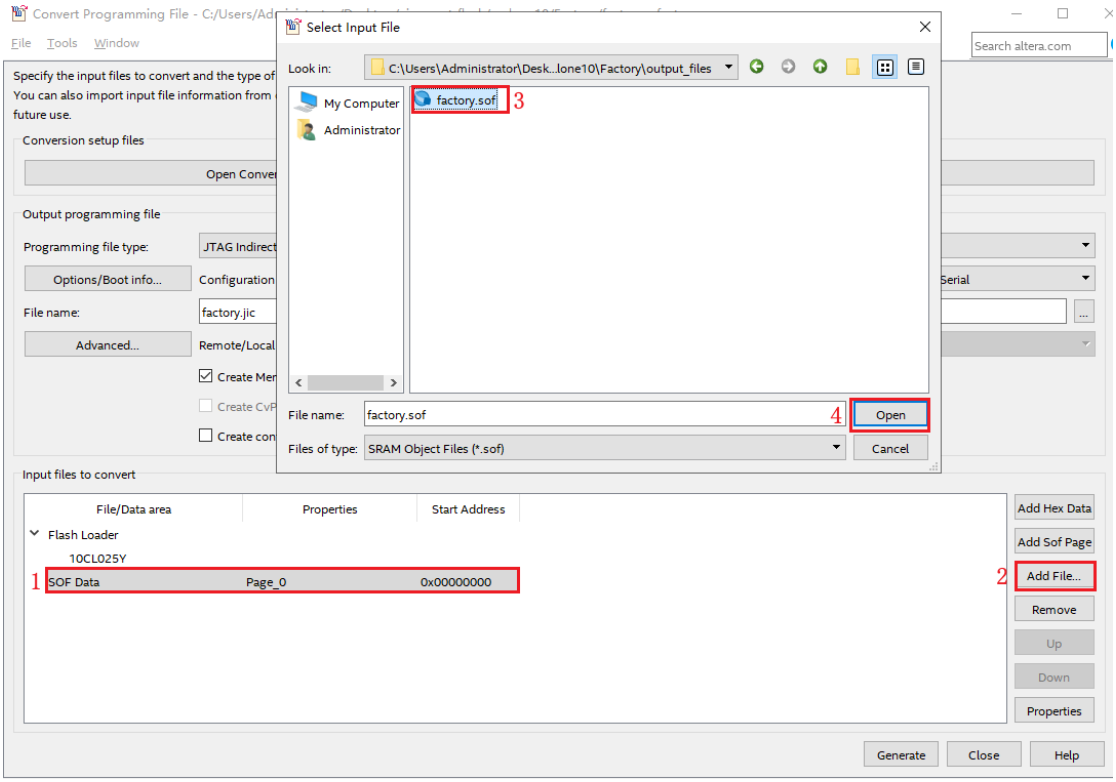

图 1-5 添加 factory.sof 文件

（4）点击 Add Hex Data 添加 Nios 编写的程序，这个首先需要我们打开 Factory 工程下的 Nios 程序生成 HEX 文件，右击 factory，选择 Make Targets->Build->mem_init_generate->Build，如下图 1-6 所示。

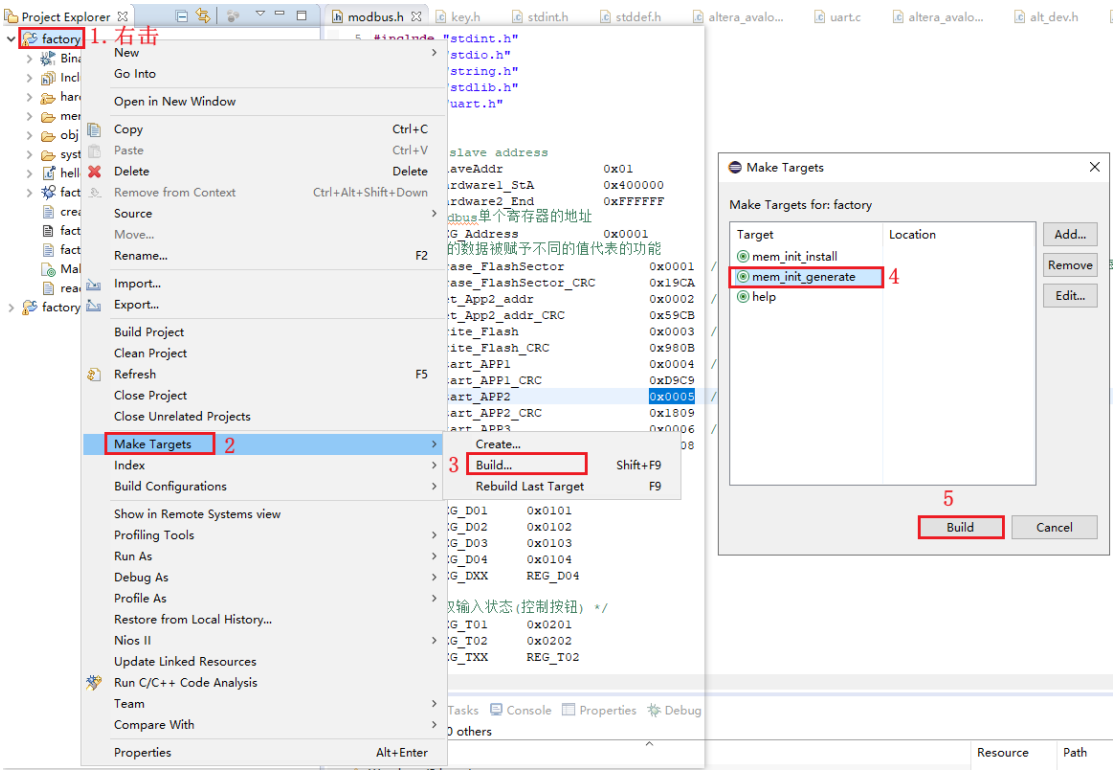

图 1-6 生成 Nios 工程的 HEX 文件

然后在我们工程 software 文件夹下会有一个 mem_init 的文件夹，里面有一个 epcq_controller.hex 文件，这个就是我们需要的文件，用户可以根据自己的实际需求进行文件名的修改，这里我们不做修改，如下图 1-7 所示。

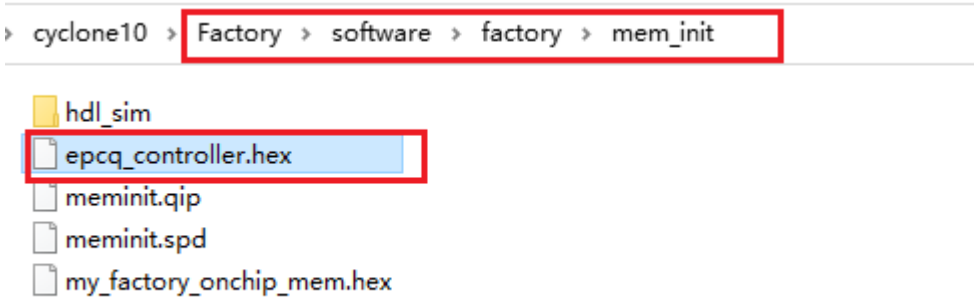

图 1-7 生成的 hex 文件所在位置

然后回到我们的 Convert Programming File 配置界面添加生成的 HEX 文件，依次点击 Add Hex Data，找到我们刚刚生成的 HEX 文件，如下图 1-8 所示。

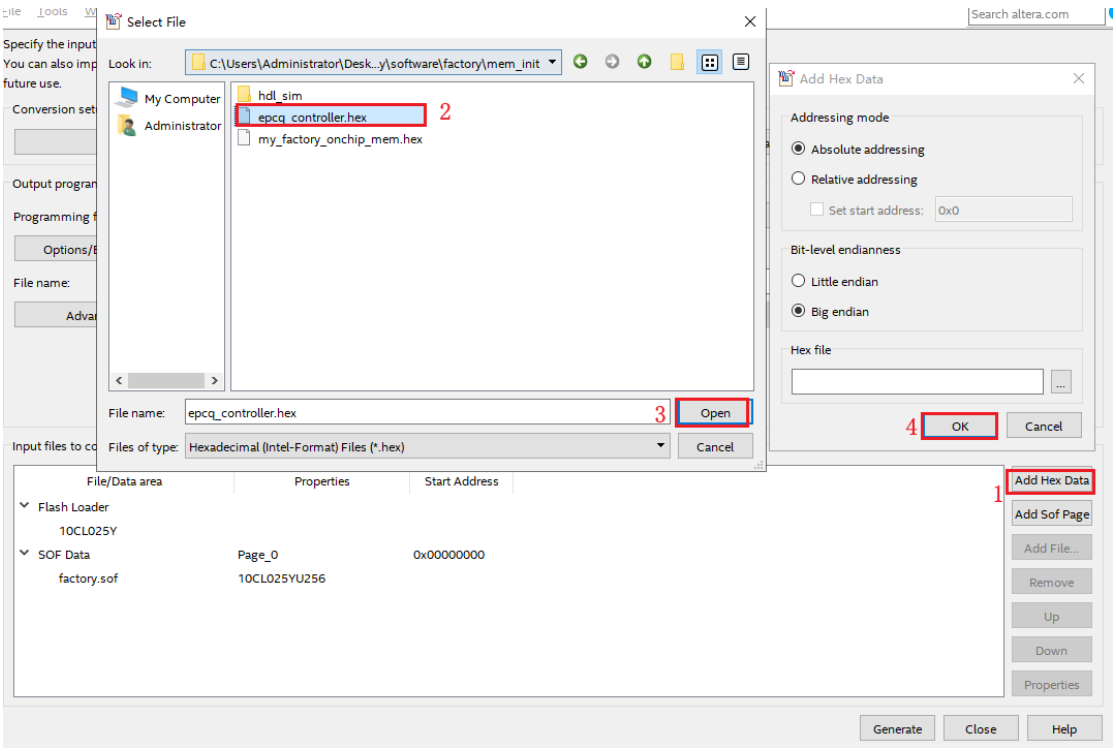

图 1-8 添加生成的 HEX 文件

（5）根据实际需求，添加出厂存放的 APP 工程文件，点击 Add Sof Page，然后点击 Properties->Block->Start address 设置为 0x400000，如下图 1-9 所示。

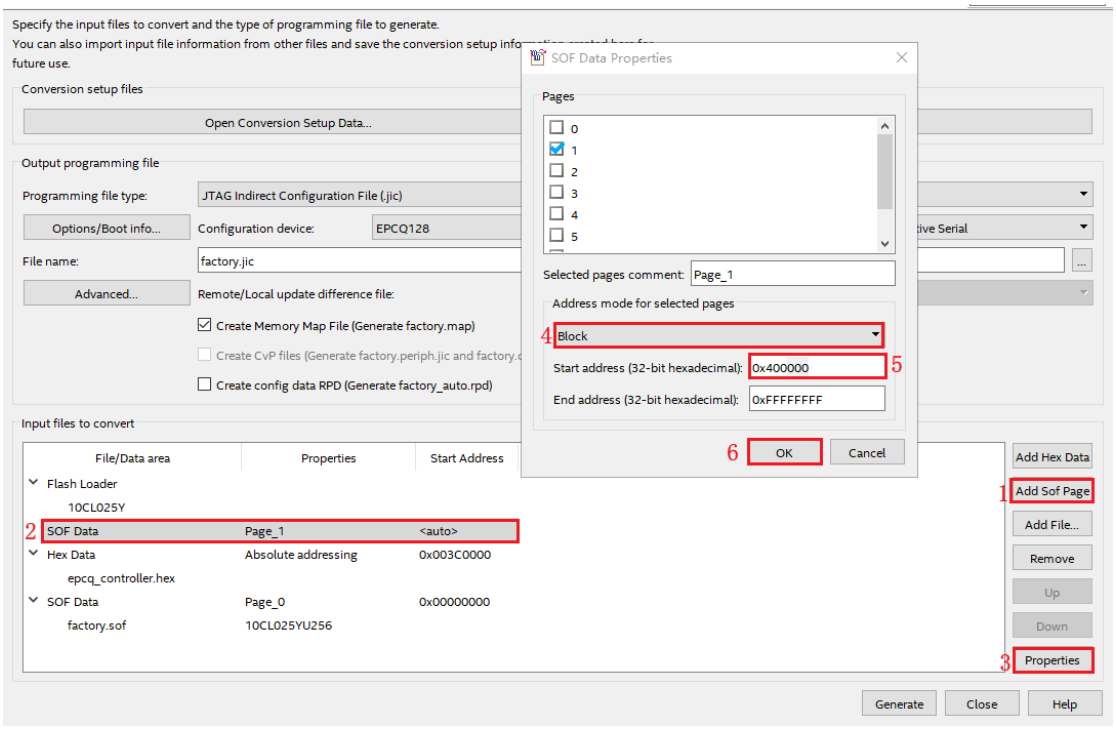

图 1-9 设置 APP 工程的起始地址

（6）点击 Add File 添加 APP 工程的 sof 文件，本次实验我们添加的是 ac108_verilog_demo 程序中的 Clock_PCF8563 工程的 SOF 文件，该工程上 实现的功能就是在数码管上显示时间或者日期。

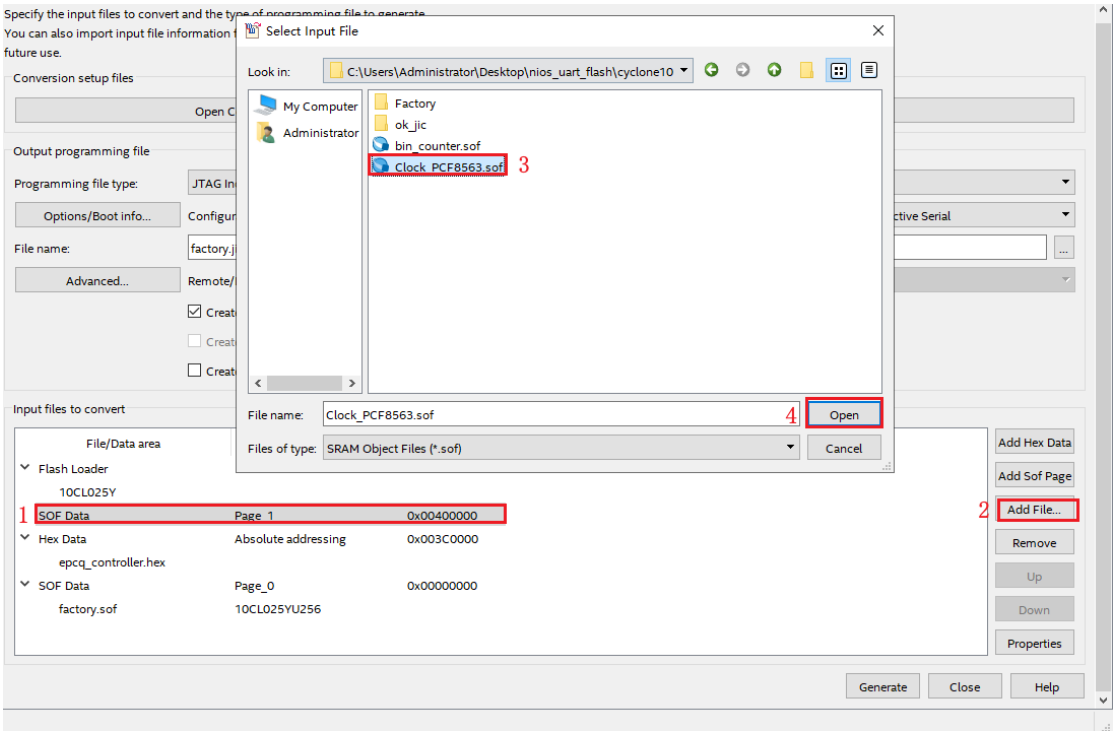

图 1-10 添加 APP 工程的 SOF 文件

如果 APP 工程中有 Nios 相关的程序，可以再次点击 Add Hex Data 添加相关的 Hex 文件，我们的工程中暂时不包含，这里就不举例说明了。

最终配置完成之后，如下图 1-11 所示。

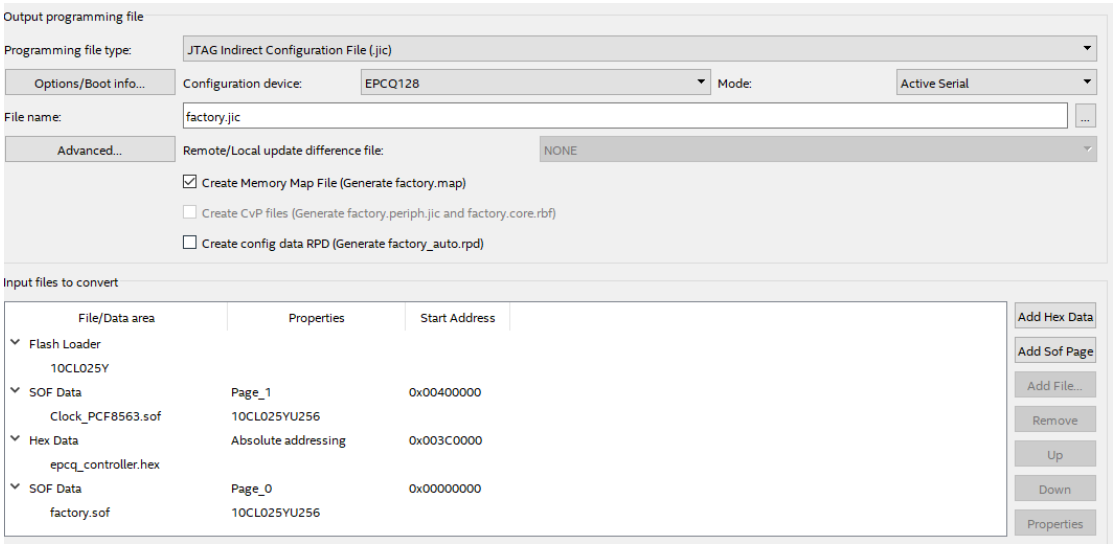

图 1-11 Convert Programming File 界面最终配置界面

然后点击 Generate，生成 factory.jic，如下图 1-12 所示。

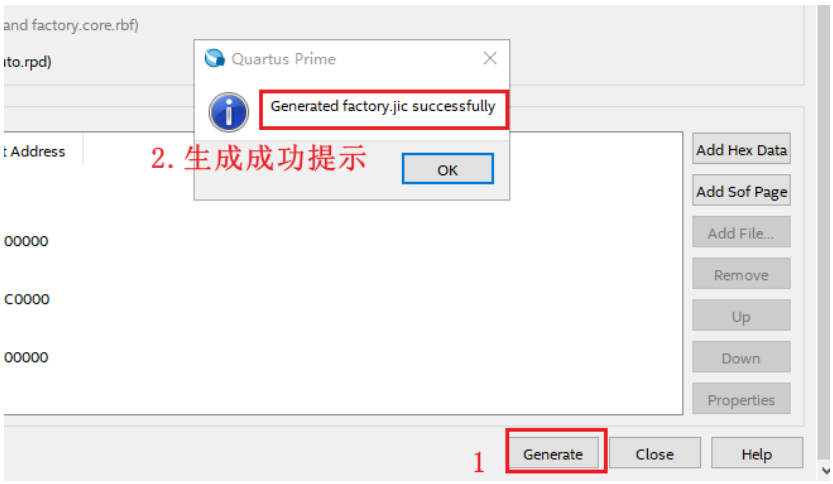

图 1-12 成功生成 jic 文件

1.1.2 下载 factory.jic 文件

我们依次点击 Tools->Programmmer->Add File，添加我们刚刚生成的 factory.jic 文件，勾选 Program/Configure，点击 Start 开始下载，如下图 1-13 所示。

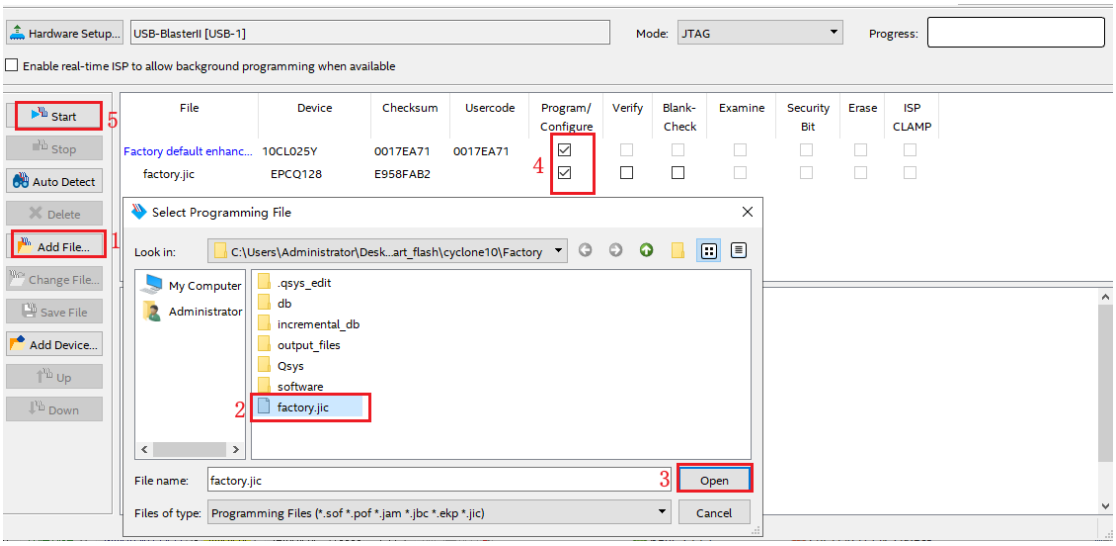

图 1-13 下载 jic 文件

下载的完成之后，我们首先打开串口调试助手，将波特率设置为 115200，方便我们观察现象，如下图 1-14 所示。

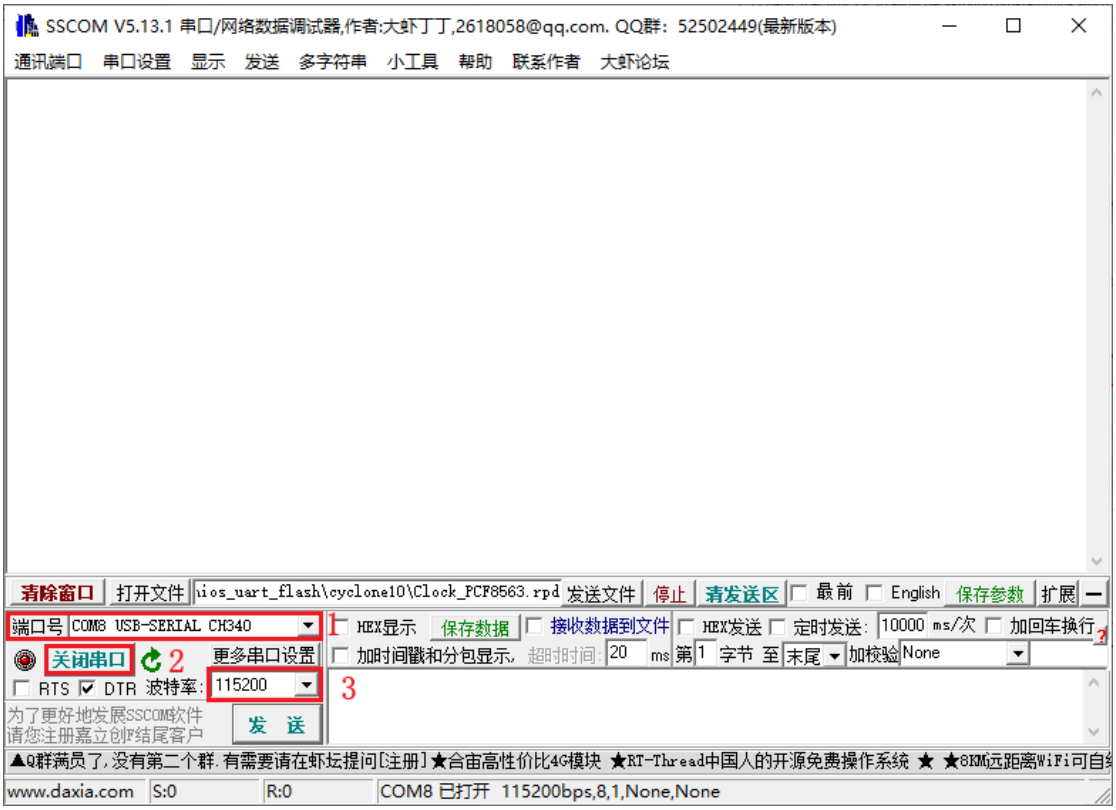

图 1-14 打开串口调试助手观察现象

然后给开发板重新上电，可以看到我们的串口上显示如下图 1-15 内容。

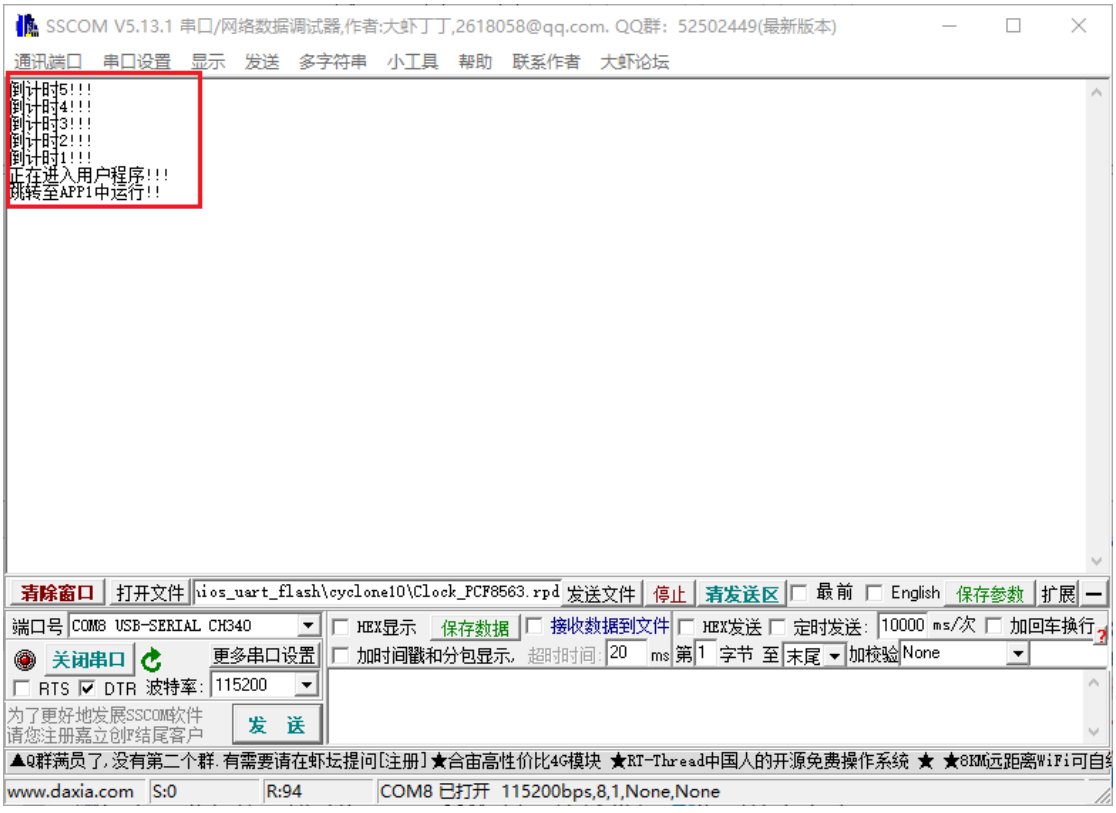

图 1-15 Factory 工程串口打印信息

可以看到，首先打印的是倒计时 5~1，这里的倒计时是提供给用户是否需要升级 APP 程序的考虑的时间，如果不更新，将会在 5S 之后，打印出正在进入用户程序，跳转至 APP1 中运行程序，此时我们边可以看到数码管上显示时间，如果显示不对，按下 S0，可以时间从 12-00-00 开始增加，如下图 1-16 所示。

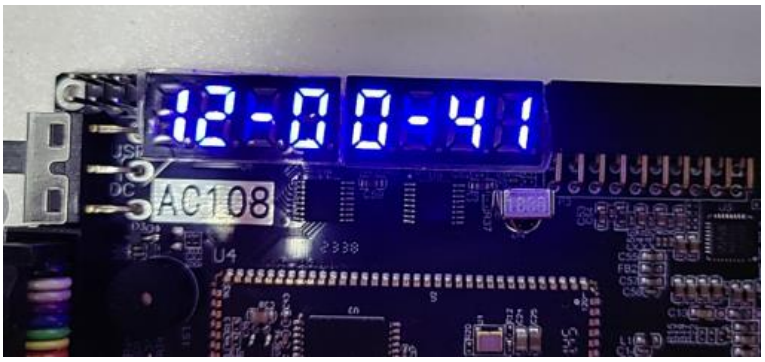

图 1-16 APP 工程功能显示

下面我们将讲解如何更新我们的用户程序。

1.1.3 更新用户程序操作

1. 生成需要更新的程序的.rpd 文件，这里我们假设我们需要更新的程序是 ac108_verilog_demo 里面的 bin_counter 程序，打开工程，依次点击 File->Convert Programming Files，然后按照前面我们生成 jic 文件的步骤，最终界面配置如下所示，一定要注意必须要勾选 Create config data RPD (Generate bin_counter_auto.rpd)

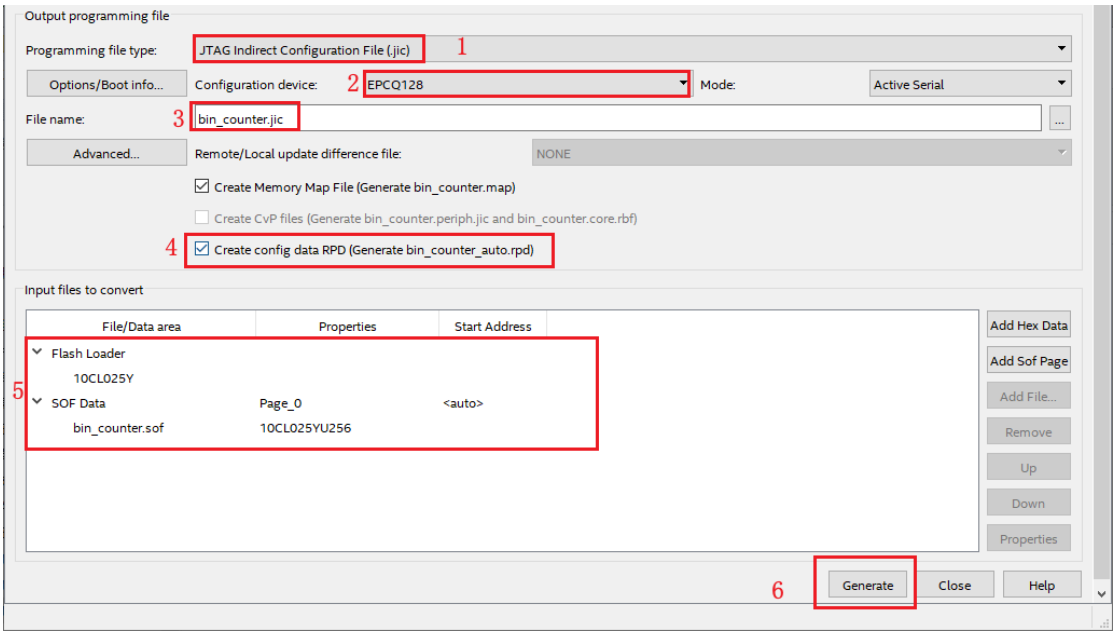

图 1-17 生成.rpd 的配置

如果包含 Nios 的程序，在这里面添加对应的 HEX 文件，再生成即可。生成完成之后，我们边可以看到，成功生成了 bin_counter_auto.rpd 文件，如下所示。

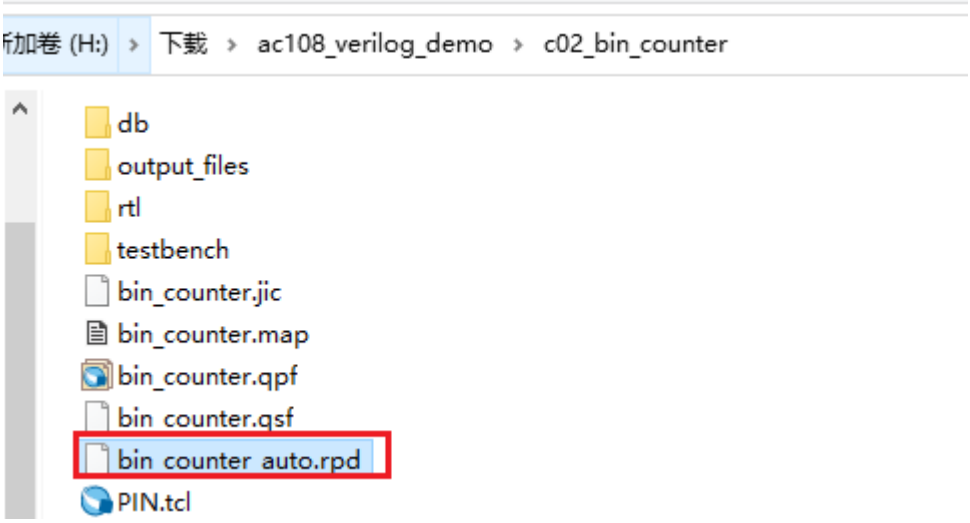

图 1-18 成功生成.rpd 文件

2. 配置串口助手用于发送更新命令，打开串口调试助手，勾选 HEX 发送，在末尾加校验 ModbusCRC16，如下图 1-19 所示。

图 1-19 配置串口调试助手

3. 发送栏中输入擦除指令 01 06 00 01 00 01，擦除之前存放的数据，如下所示。

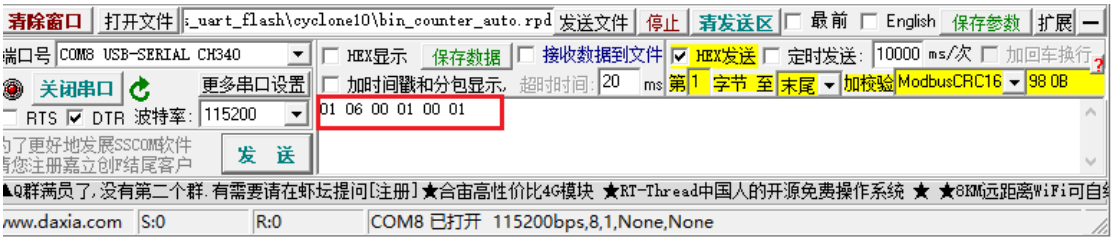

图 1-20 发送栏中输入擦除指令

4. 然后重新给开发板上电，在前 5S 内，点击发送，将擦除指令发出，可以看到串口打印如下图 1-21 内容。

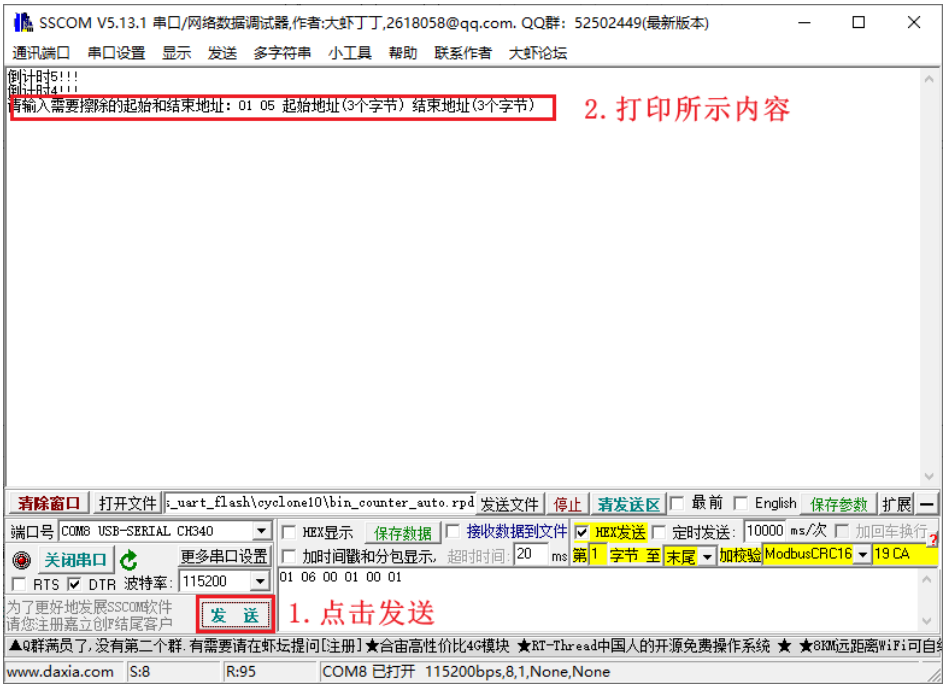

图 1-21 发送擦除指令

5. 给出需要擦除的起始地址，我们根据.rpd 文件的大小进行擦除，可以看到生成的 bin_conter_auto.rpd 的文件大学奥为 718569 字节，如下所示。

图 1-22 查看需要更新的.rpd 文件的大小

我们在擦除的过程中是按照 sector 进行擦除的，根据下图 1-23 所示的内容，EPCQ128 一共有 256 个 sector，每个 sector 的大小为 65536 个字节，我们擦除的起始地址是 0x400000，结束地址根据.rpd 文件可以计算得出：718569/65536，大约需要擦除 11 个块，那么我们需要擦除的结束地址至少需要大于：0x400000+0xB0000（11*65536）-1=0x4AFFFFF。

Details	EPCQ16	EPCQ32	EPCQ64	EPCQ128	EPCQ256	EPCQ512/A
Bytes	2,097,152 bytes [16 megabits (Mb)]	4,194,304 bytes (32 Mb)	8,388,608 bytes (64 Mb)	16,777,216 bytes (128 Mb)	33,554,432 bytes (256 Mb)	67,108,864 bytes (512 Mb)
Number of sectors	32	64	128	256	512	1,024
Bytes per sector	65,536 bytes [512 kilobits (Kb)]					
Total numbers of subsectors (8)	512	1,024	2,048	4,096	8,192	16,384
Bytes per subsector	4,096 bytes (32 Kb)					
continued...						

图 1-23 Supported Memory Array Organization in EPCQ Devices

这里我们根据提示输入需要擦除的起始地址命令：01 05 40 00 00 4A FF FF。可以看到串口打印如下图 1-24 内容。可以看到擦除的块为 64~74，刚好 11 个块，符合我们的需要。

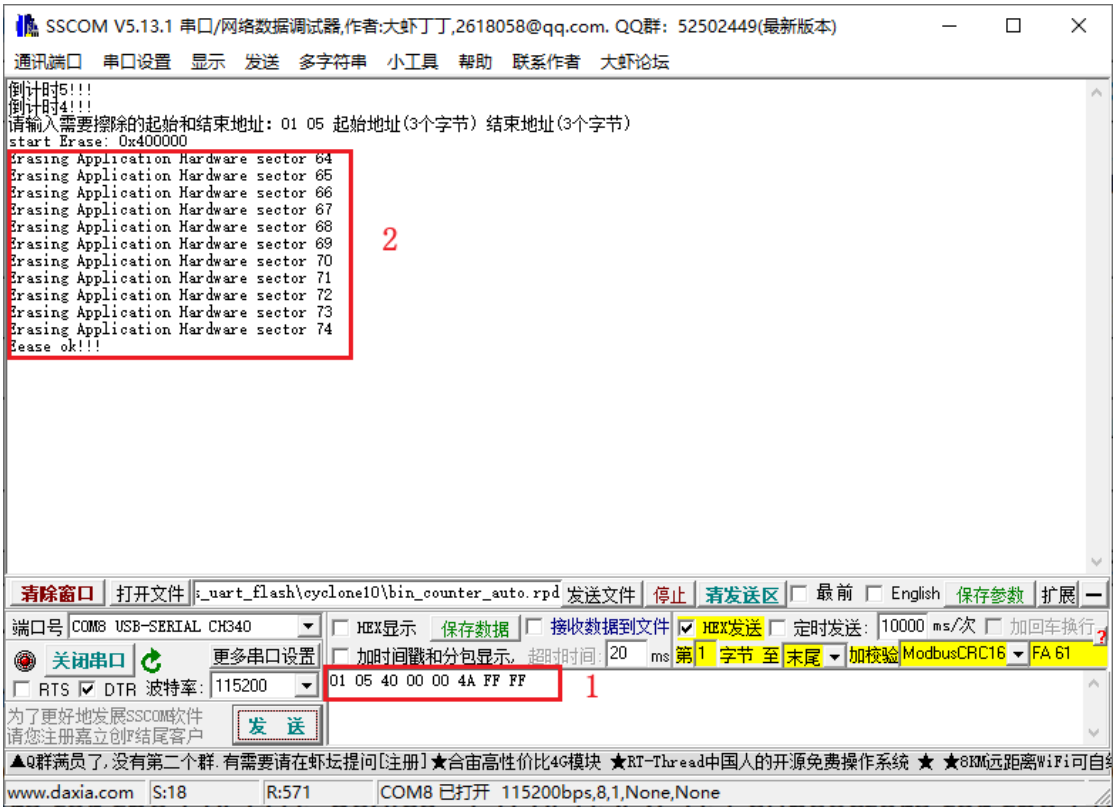

图 1-24 擦除之前存放的 APP1 的程序

6. 发出需要写入更新文件的命令：01 06 00 01 00 03，然后串口打印如下内容。

图 1-25 发出需要写入文件的命令

7. 点击打开软件，找到我们生成的 bin_counter_auto.rpd 文件，点击发送文件，如下图 1-26 所示，等待发送完成

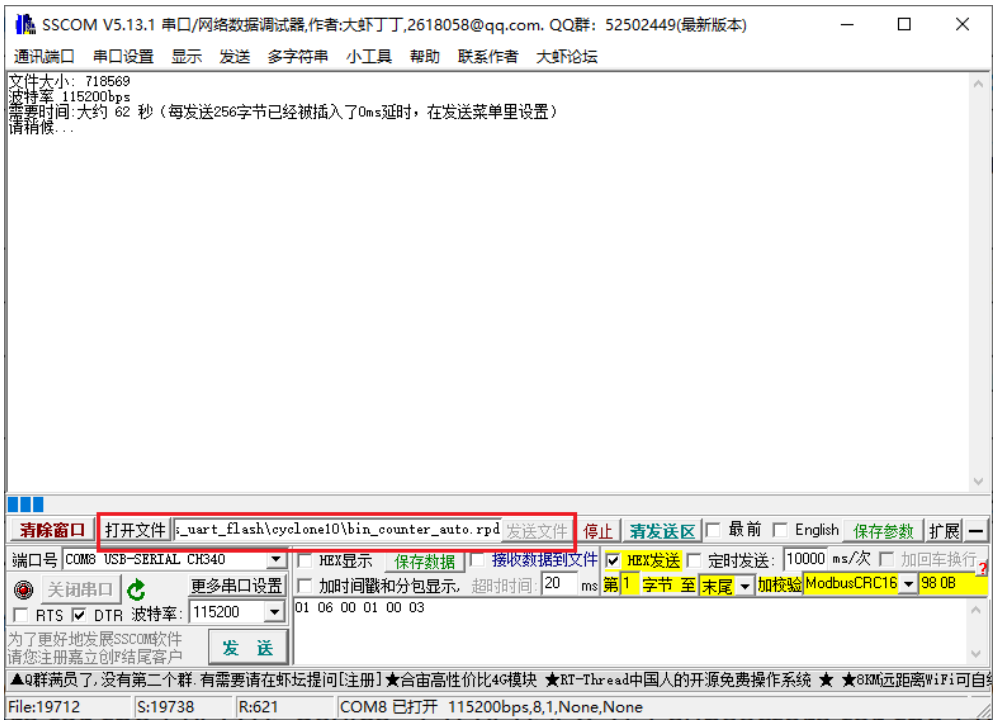

图 1-26 发送文件

8. 文件发送完成之后，将会打印如下内容，直接跳转至 APP1 中运行，如下图 1-27 所示。

图 1-27 文件发送完成

此时我们观察开发板上现象，此时数码管将不再显示时间了，说明之前的文件已经擦除完成了，开发板的右下角的 D0 开始闪烁，这也就是 bin_counter 工程实现的功能，这也就说明，我们需要更新的文件写入成功。

1.2 更新 APP2 区

有的用户需要两个 APP 区，下面我们将讲解如何更新 APP2 中的内容，关于 jic 文件以及 rpd 文件的生成，本章将不再进行说明，请用户查看前面 1.1 节的内容。

1. 程序上电 5S 内，发送 01 06 00 01 00 02，弹出如下界面，给出 APP2 的起始地址。

图 1-28 发出需要给 APP2 起始地址的命令

2. 然后给出 APP2 的起始地址，根据用户实际需求自行给出，比如根据我们前面 1.1 节的内容，bin_counter_auto.rpd 文件存放的地址为 0x400000~0x4AFFFFF，这里我们存放的文件初始地址必须大于 0x4AFFFFF，才能保证 APP1 中存放的程序不受影响，我们设置 APP2 的初始地址为 0x500000，那么就需要给出命令：01 05 50 00 00。如下所示。

图 1-29 给出 APP2 的起始地址

3. 发送擦除命令：01 06 00 01 00 01

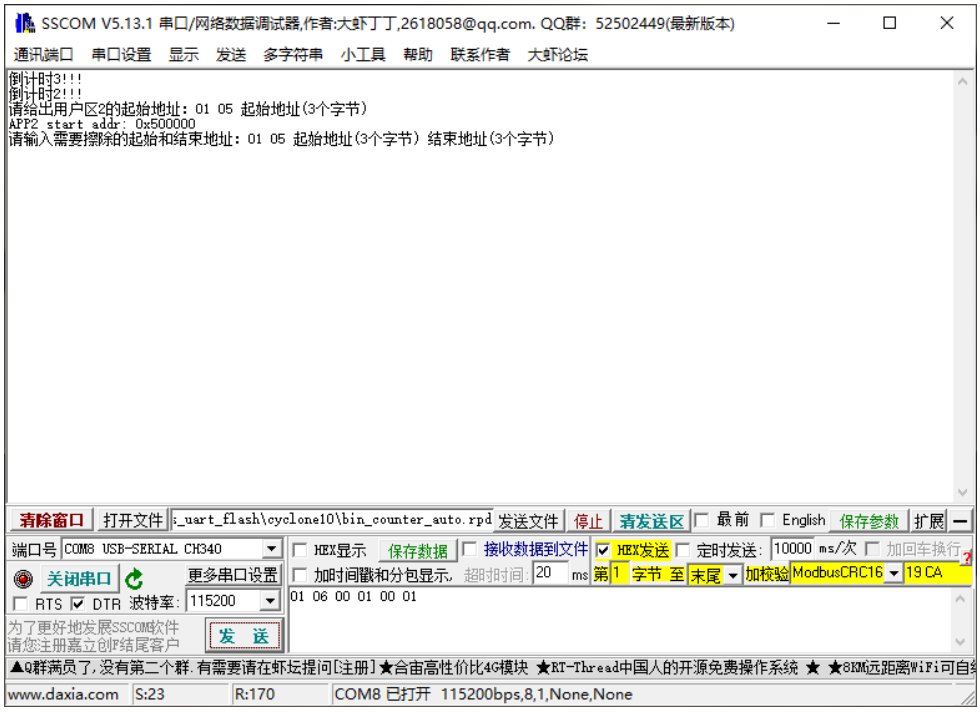

图 1-30 发送擦除命令

4. 给出需要擦除的起始地址，我们这里设置写入 APP2 的程序为 Clock_PCF8563.rpd，该文件大小为 718569 字节，至少需要 11 块，那么结束地址至少要大于 0x500000+0xB0000（11*65536）-1=0x5AFFFF，这里给出命令：01 05 50 00 00 5A FF FF，如下所示。

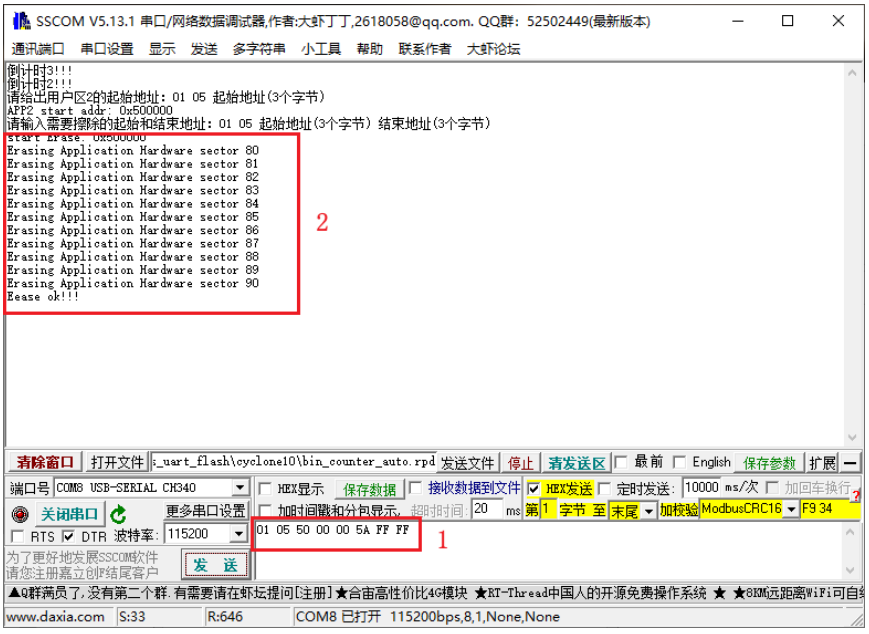

图 1-31 擦除 APP2 需要存放的程序空间

5. 这里需要注意的是，更新 APP2 中的文件时，必须优先发送需要更新的区的命令：01 05 02（01：APP1；02：APP2）；如下图 1-32 所示。

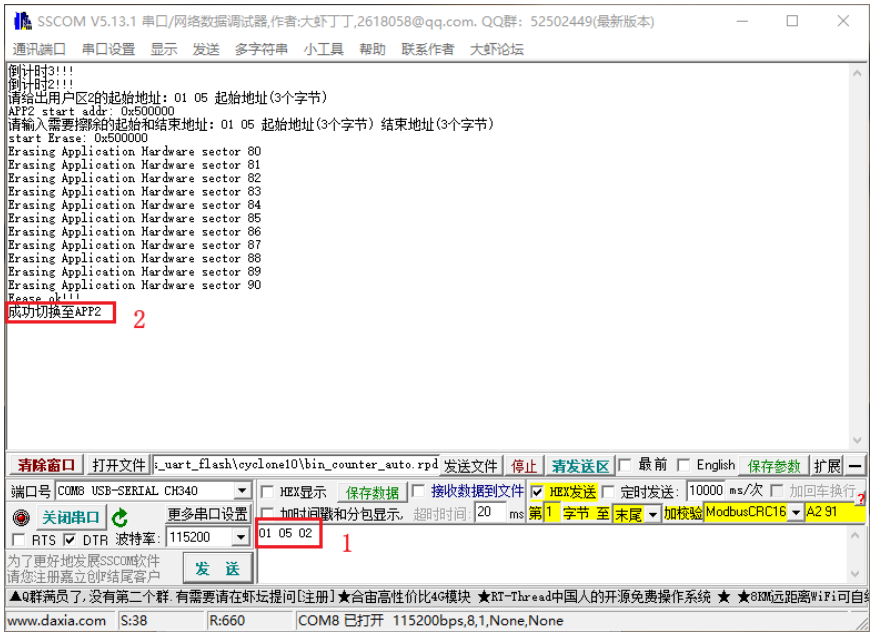

图 1-32 切换区

6. 发出写文件命令：01 06 00 01 00 03。

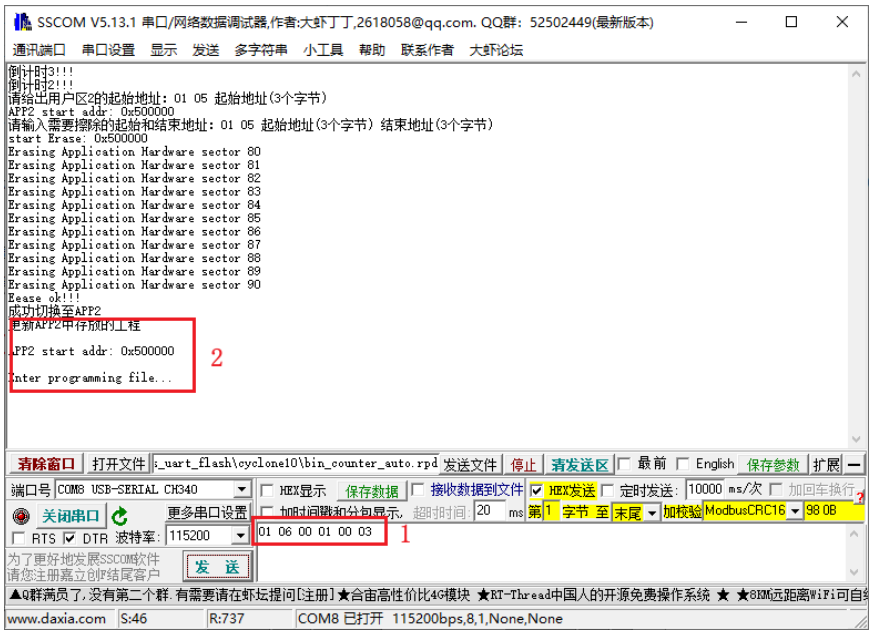

图 1-33 发出写文件命令

7. 然后打开文件，找到生成的 Clock_PCF8563.rpd 文件，点击发送，如下所示。

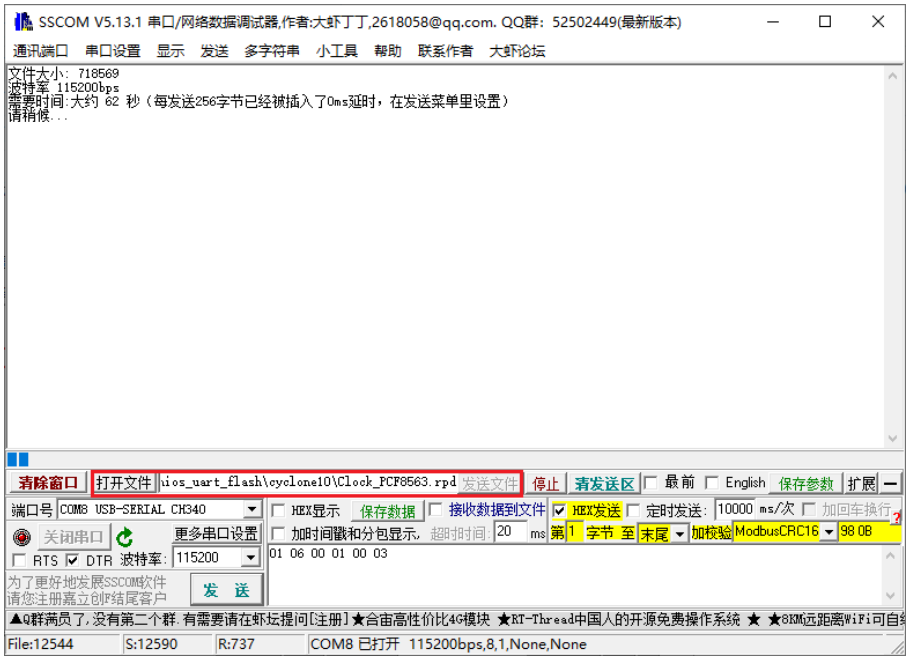

图 1-34 发送文件

8. 文件发送完成之后，如下所示。

图 1-35 文件发送完成

9. 由于现在有两个区，此时就需要指定运行哪个区中存放的程序，此时我们板子 APP1 中存放的应该是 bin_counter_auto.rpd 文件，实现的功能是开发板上的 D0 闪烁，APP2 中存放的是 Clock_PCF8563.rpd 文件，实现的功能是数码管上显示时间，此时我们发送跳转到 APP2 的命令：01

06 00 01 00 05，如下所示。

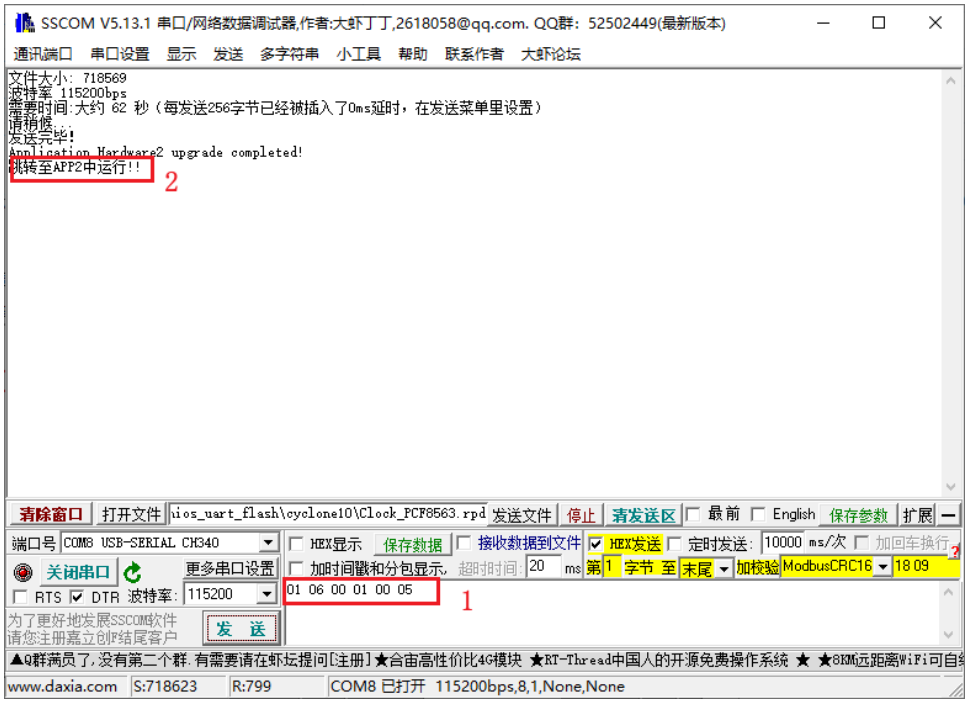

图 1-36 跳转到 APP2 运行

指令发送完成之后，可以看到开发板上的数码管开始显示时间，说明我们文件写入成功。

此时给开发板重新上电，重新开始倒计时，5S 之后默认跳转至 APP1 中运行，如果想要上电之后，跳转至 APP2 中运行，需要发送跳转指令 01 06 00 01 00 05。

1.3 涉及指令说明

针对工程中设置的指令，我们这里整理一个完整的表格进行说明。

命令内容	命令意义
01 06 00 01 00 01 +CRC 校验（通过软件自动添加）	擦除指令
01 06 00 01 00 02 +CRC 校验（通过软件自动添加）	分出 APP2 区的指令
01 06 00 01 00 03 +CRC 校验（通过软件自动添加）	向 Flash 中写入需要更新的程序的指令
01 06 00 01 00 04 +CRC 校验（通过软件自动添加）	跳转至 APP1 区运行指令
01 06 00 01 00 05 +CRC 校验（通过软件自动添加）	跳转至 APP2 区运行指令
01 05 起始地址（3 个字节） 结束地址（3 个字节）	给出需要擦除的起始和结束地址，在发送擦除指令之后，给出该指令之后，开始擦除用户指定的需要擦除的区域
01 05 APP2 的起始地址（3 个字节）	发出 01 06 00 01 00 02 指令之后，必须接着发送该指令指定 APP2 的起始地址
01 05 需要更新的区（1 个字节：01：APP1；02：	需要更新的 APP 区，这个只针对用户有两个 APP

APP2）	区时使用，在发送 01 06 00 01 00 03 指令之前，需要先发送该指令，指定需要更新的区