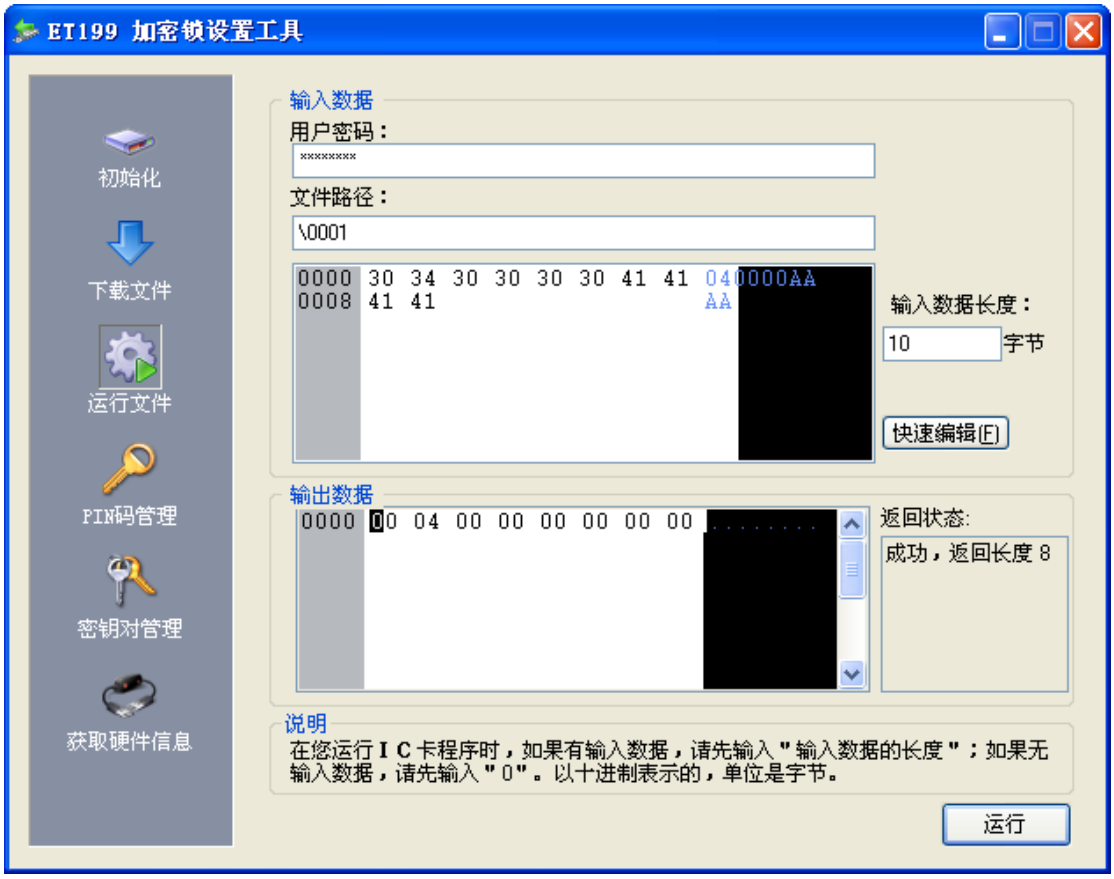
本篇文章中详细介绍了在 KEIL 环境中，使用 C51 开发时，进行调试的方法，希望对开发是遇到的问题有所帮助。

（1） 调试时使用_set_response，_exit 函数调用，程序遇到函数_exit 时就退出，可以检查中间变量。使用 C51 的 debug 不准确，还是在 drvset 中调试好。过程如下，如 write.c 程序：

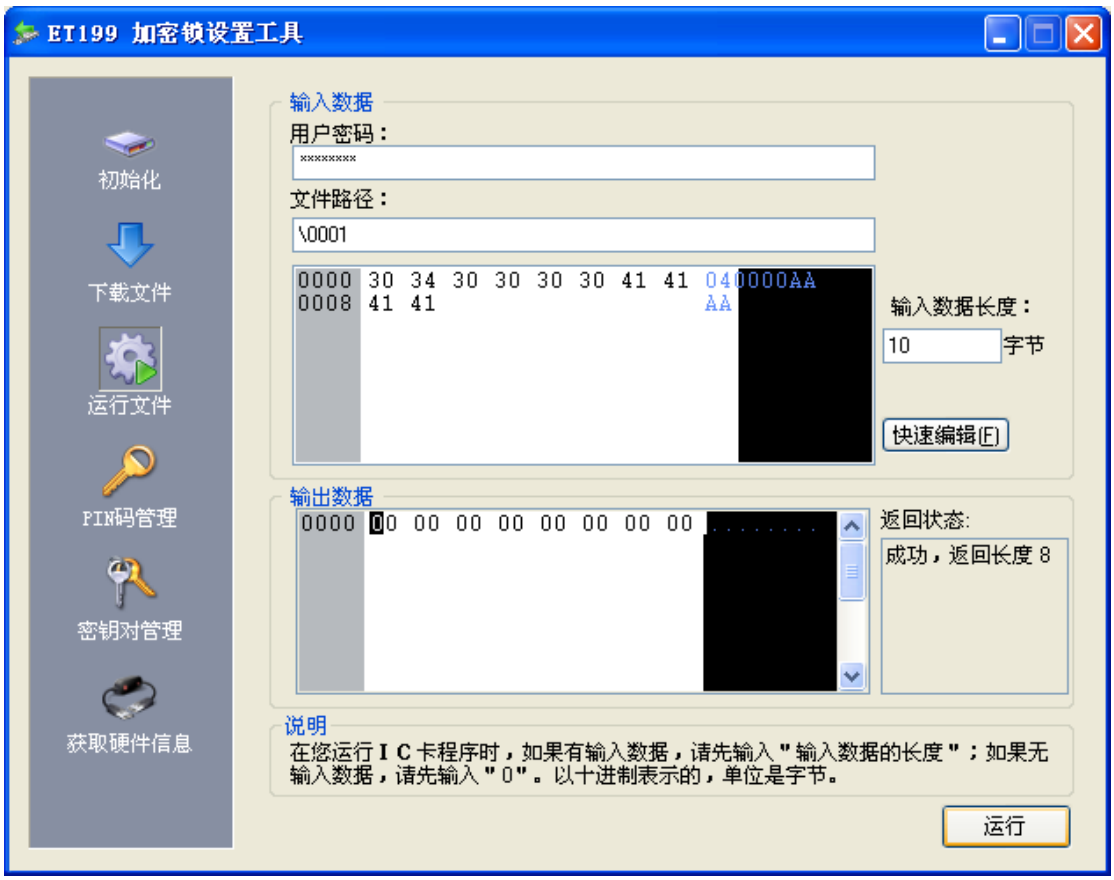
```
BYTE xdata bRes = 0;
WORD xdata dLen = 0; //本次要写入的数据长度
WORD xdata offset = 0; //本次要写入的数据偏移量
HANDLE xdata hFile = 0;
BYTE xdata pbData[250];
BYTE xdata dataArr[2];
BYTE xdata offsetArr[4];

memcpy(dataArr, pbInBuff, 2);//本次要写入的数据长度
dLen = strtol(dataArr,NULL,16);

memcpy(offsetArr, pbInBuff+2, 4); //本次要写入数据的偏移量
offset = strtol(offsetArr,NULL,16);

//屏蔽 1
//检查 dLen 转换后的值是否正确
//_set_response(8,&dLen);
//_exit();



//屏蔽 2
//检查 offset 转换后的值是否正确
//_set_response(8,&offset);
//_exit();

memcpy(pbData, pbInBuff+6, dLen);

//屏蔽 3
//检查 pbData 中的值是否正确
//_set_response(10,pbData);
//_exit();
```

打开屏蔽 1 时，程序运行到这里退出，使用 drvset 看返回值，返回是正确的，但第二次运行时，数据就不对了

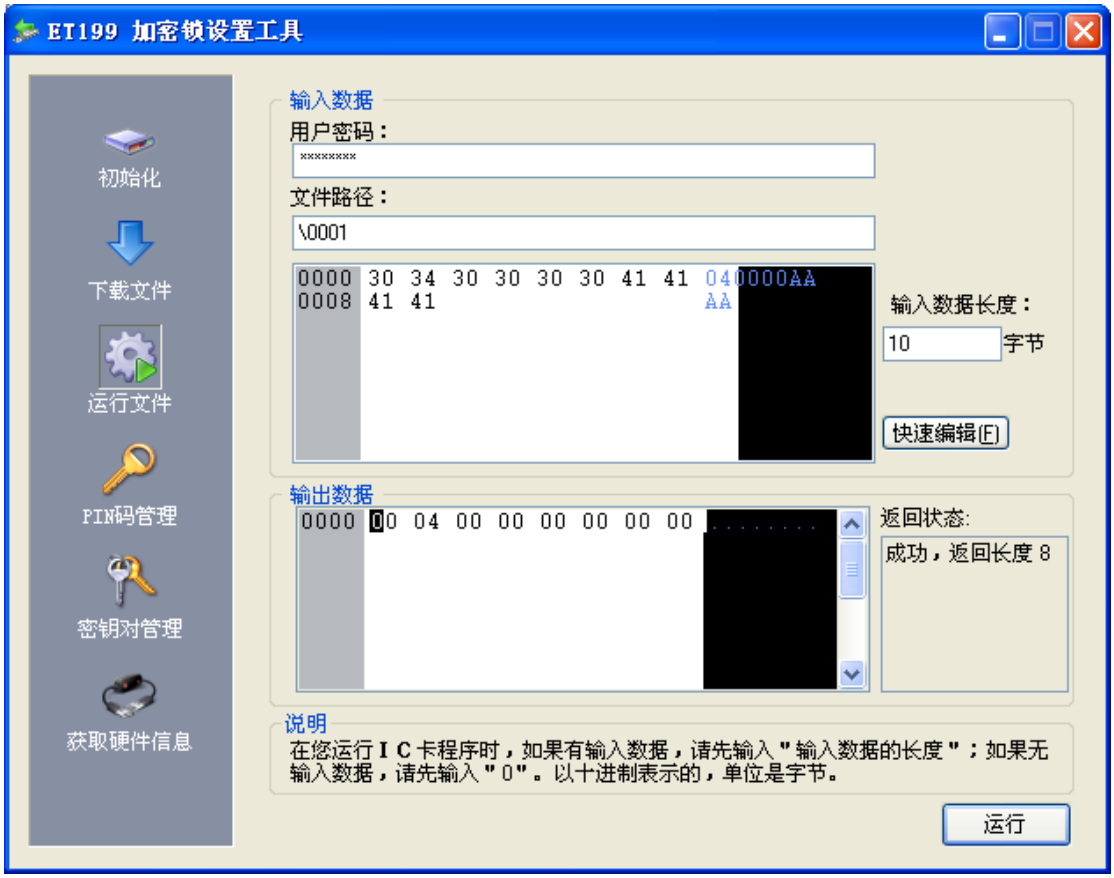
第一次结果：


ET199 加密锁设置工具
初始化
下载文件
运行文件
PIN码管理
密钥对管理
获取硬件信息
输入数据
用户密码：
****
文件路径：
\0001
0000 30 34 30 30 30 30 41 41 040000AA
0008 41 41 AA
输入数据长度：
10
字节
快速编辑(F)
输出数据
0000 00 04 00 00 00 00 00 00 ....
返回状态:
成功，返回长度 8
说明
在您运行ＩＣ卡程序时，如果有输入数据，请先输入"输入数据的长度"；如果无输入数据，请先输入"0"。以十进制表示的，单位是字节。
运行

第二次结果


ET199 加密锁设置工具
初始化
下载文件
运行文件
PIN码管理
密钥对管理
获取硬件信息
输入数据
用户密码：
****
文件路径：
\0001
0000 30 34 30 30 30 30 41 41 040000AA
0008 41 41 AA
输入数据长度：
10
字节
快速编辑(F)
输出数据
0000 00 00 00 00 00 00 00 00 ....
返回状态:
成功，返回长度 8
说明
在您运行ＩＣ卡程序时，如果有输入数据，请先输入"输入数据的长度"；如果无输入数据，请先输入"0"。以十进制表示的，单位是字节。
运行

可以看到 dLen 值发生了变化，而把屏蔽 1 移到 memcpy(offsetArr, pbInBuff+2, 4);前面则没有错误，于是怀疑是 strtol 函数在转换时将内存中改写了。因此改为使用 CharToWORD 函数，然后检查，每次运行后的结果都是正确的。

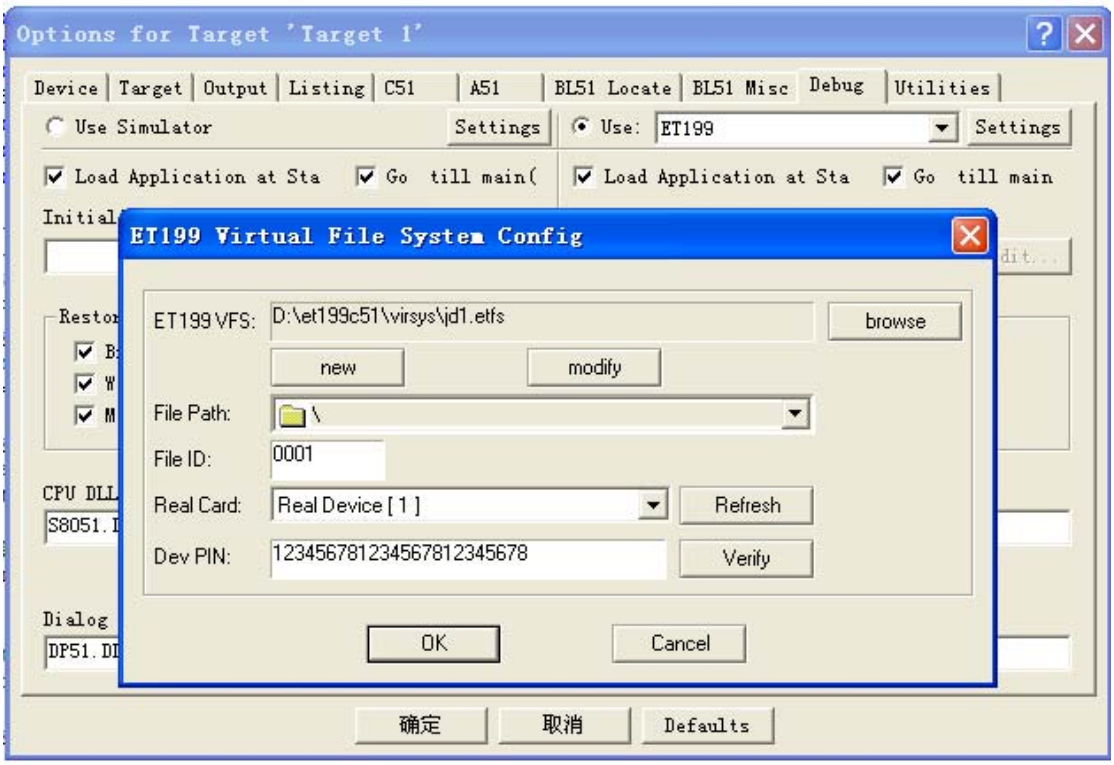


即：

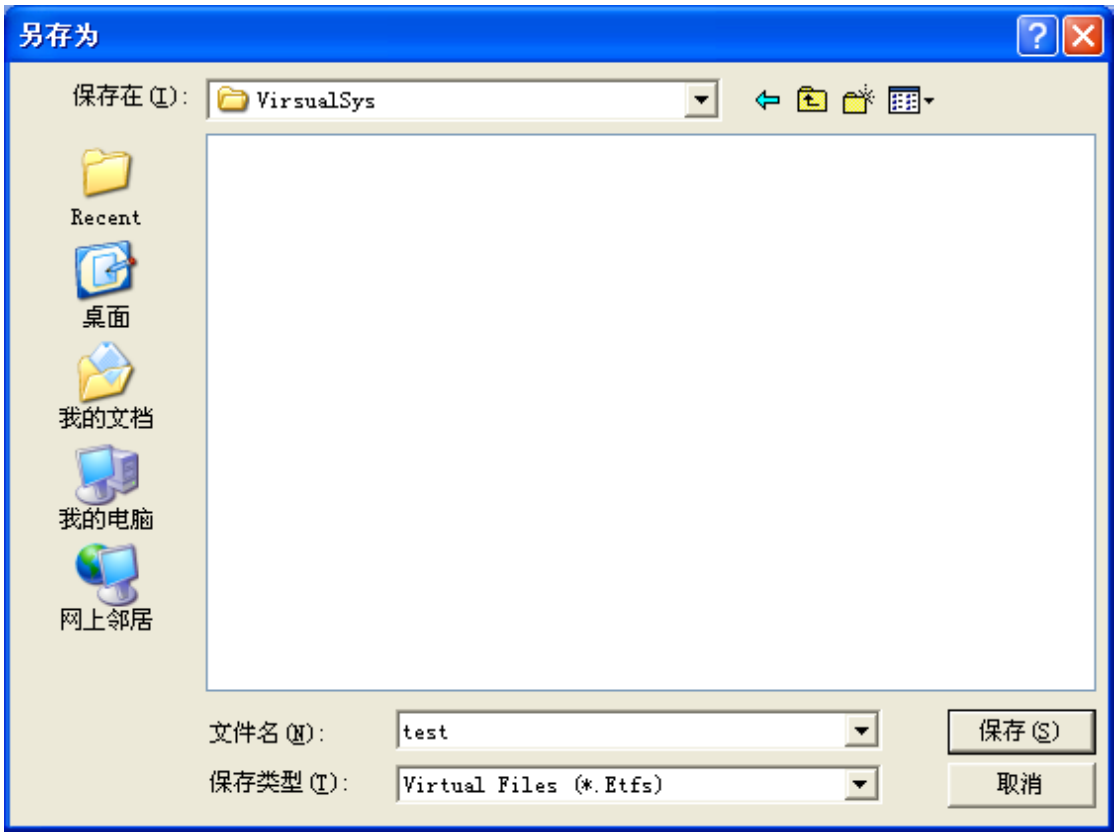
```
WORD CharToWORD(BYTE* pbData, BYTE bLen)
{
    BYTE i=0;
    BYTE bTemp=0;

    WORD wRet = 0;
    for(i=0; i<bLen; ++i)
    {
        wRet = wRet<<4;
        bTemp = pbData[i]-0x30;
        wRet += bTemp;
    }

    return wRet;
}
```

```
memcpy(dataArr, pbInBuff, 2);//本次要写入的数据长度
//dLen = strtol(dataArr,NULL,16);
dLen = CharToWORD(dataArr, 2);

memcpy(offsetArr, pbInBuff+2, 4); //本次要写入数据的偏移量
//offset = strtol(offsetArr,NULL,16);
offset = CharToWORD(offsetArr, 4);

//屏蔽 1
//检查 offset 转换后的值是否正确
//_set_response(8,&offset);
//_exit();

//屏蔽 2
//检查 dLen 转换后的值是否正确
//_set_response(10,&dLen);
//_exit();

memcpy(pbData, pbInBuff+6, dLen);

//屏蔽 3
//检查 pbData 中的值是否正确
//_set_response(10,pbData);
//_exit();
```

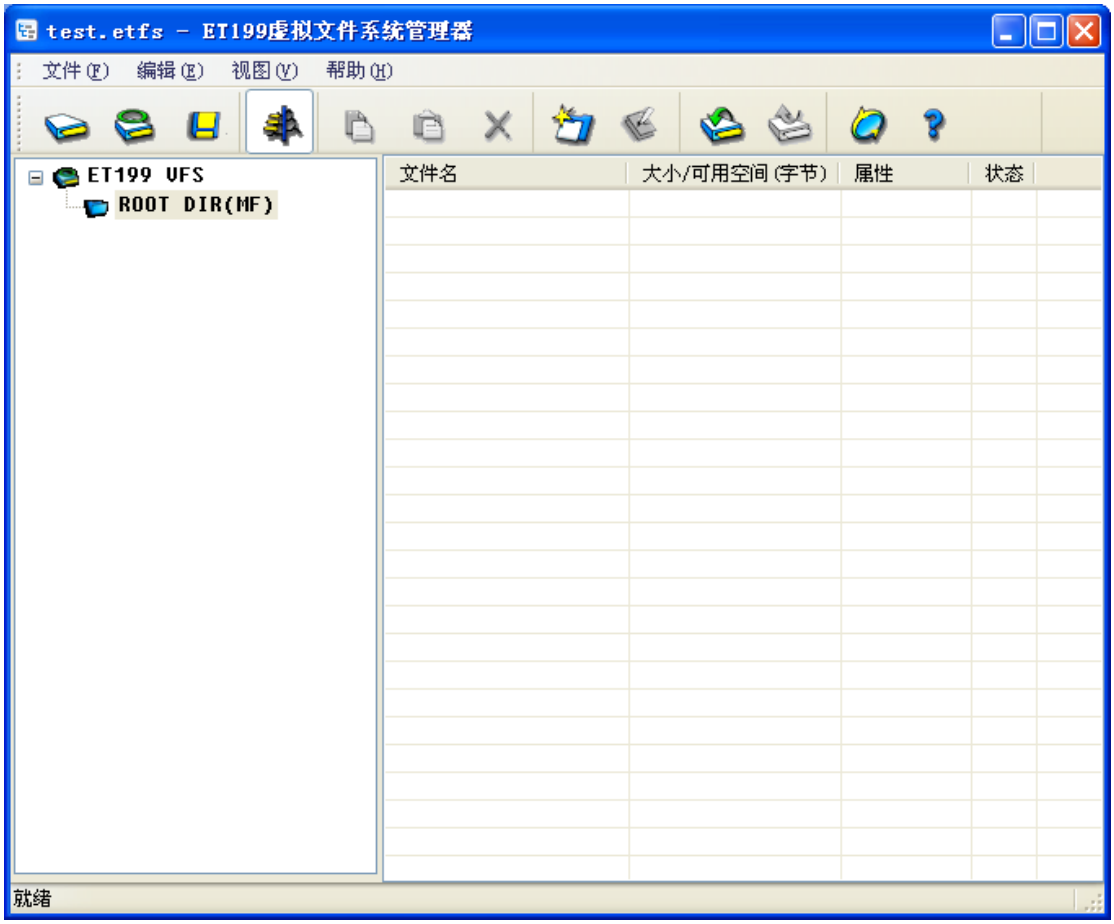
（2）　在 KEIL 中如何能一次下载所有编好的可执行文件，而不用每次都使用 drvset 导入。



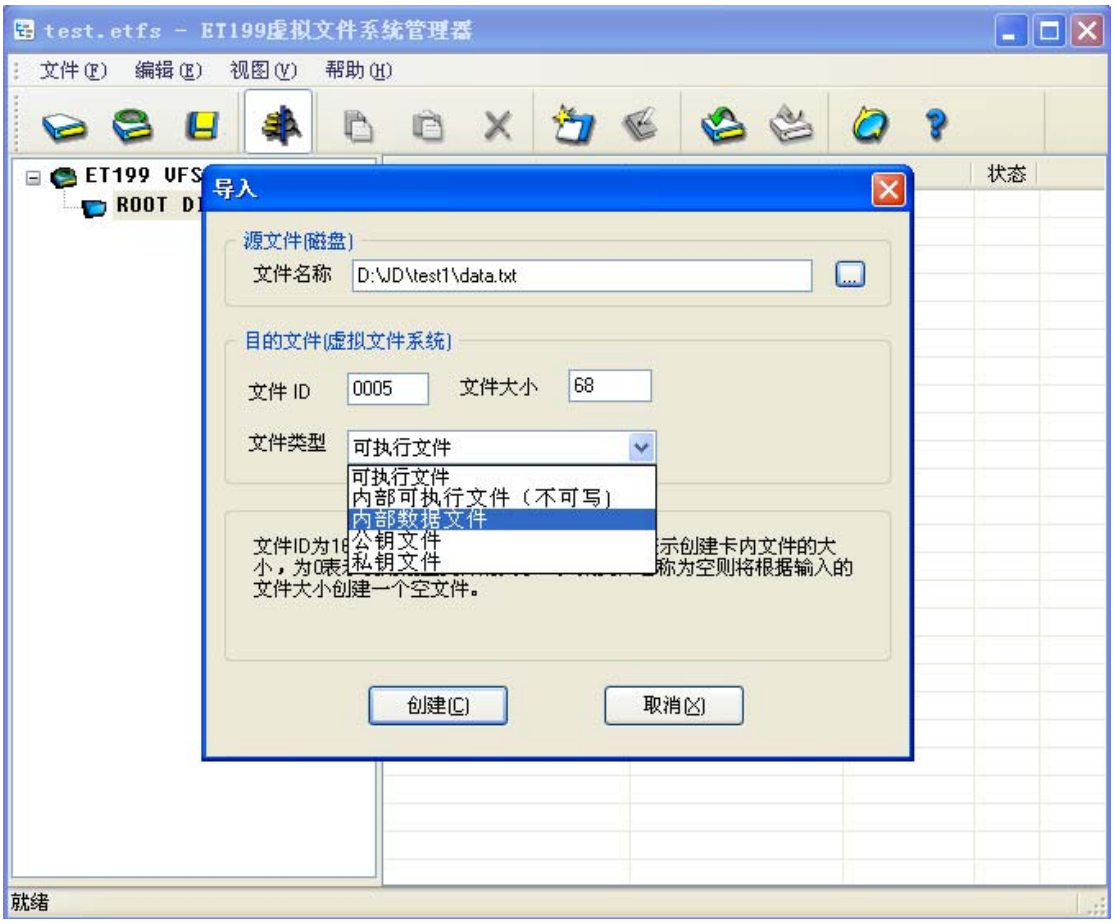
在工程的 option 中选择 Debug，然后选择 ET199，按“Settings”按钮（如果没有 ET199，见说明书中的 KEIL 环境配置章节）。这里点击“new”按钮建立一个新的虚拟系统，test.etfs。

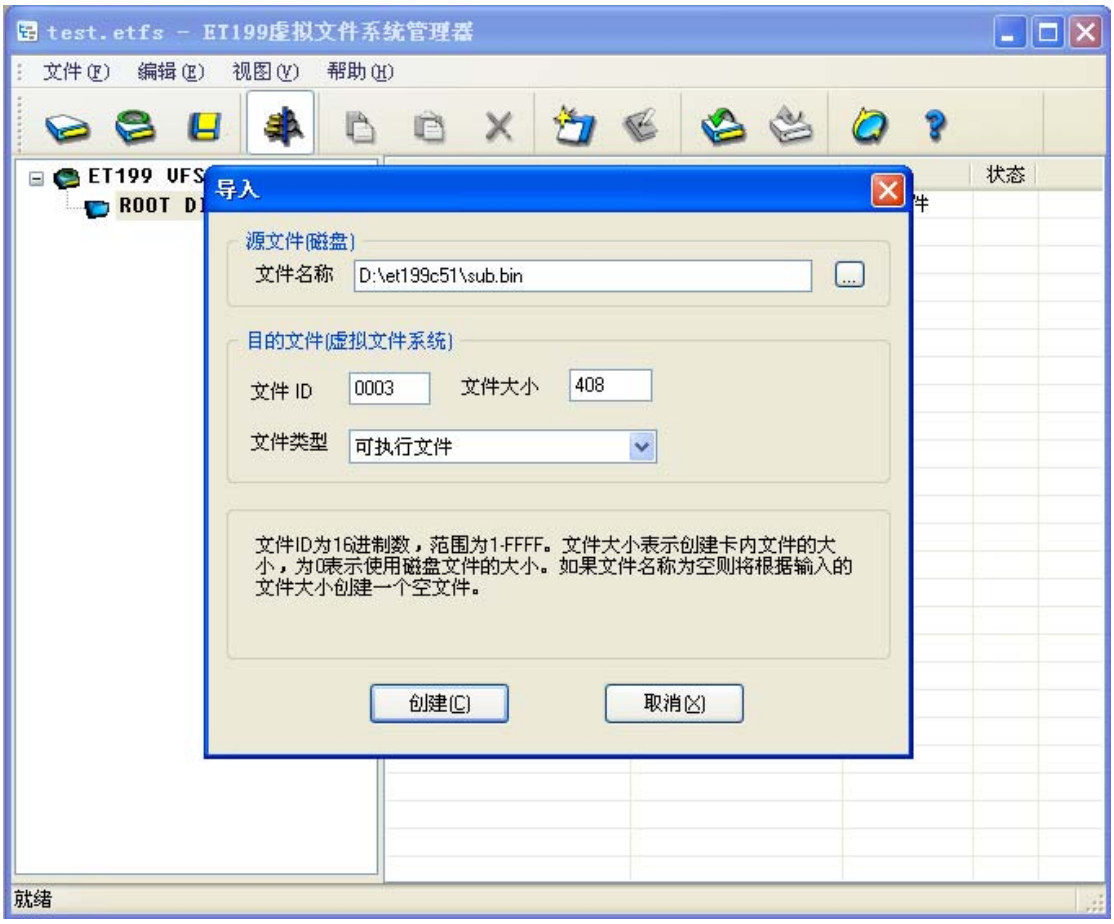


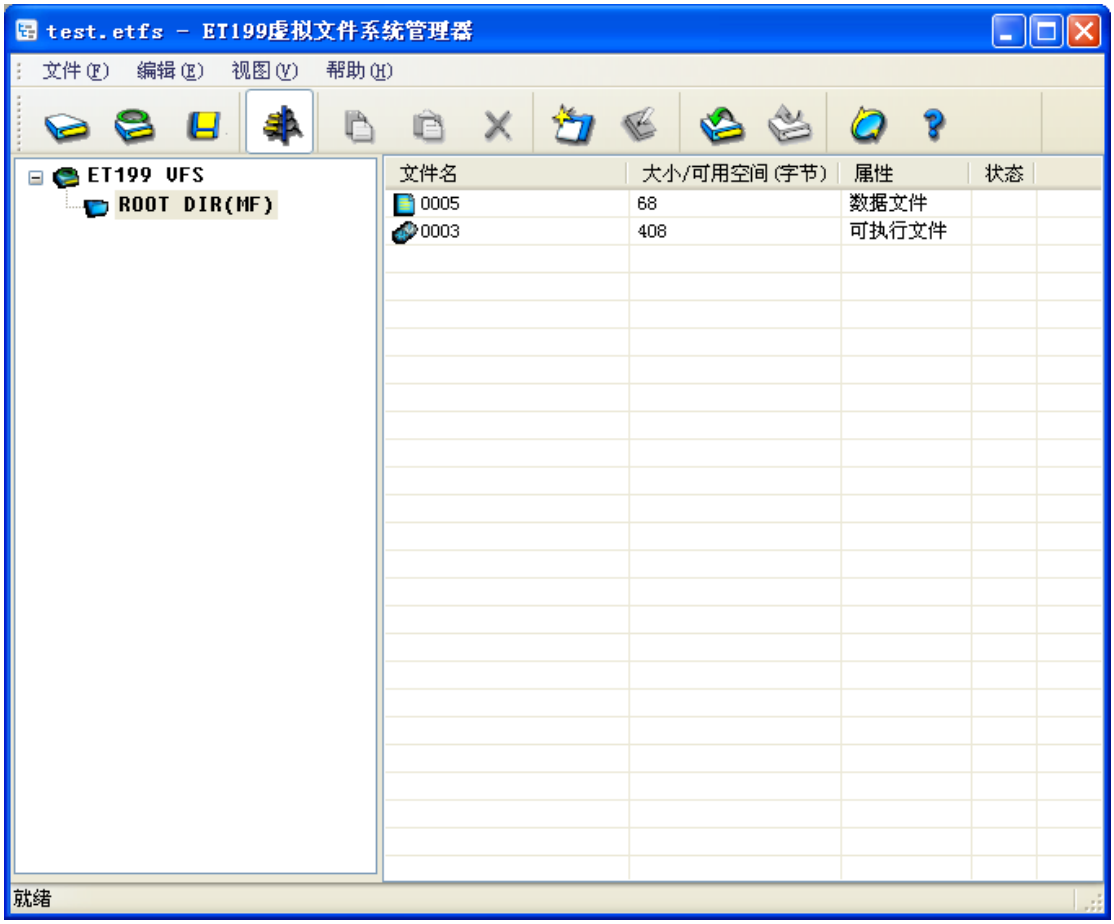
使用 VfsSet.exe 工具打开 test.etfs，这时里面是空的



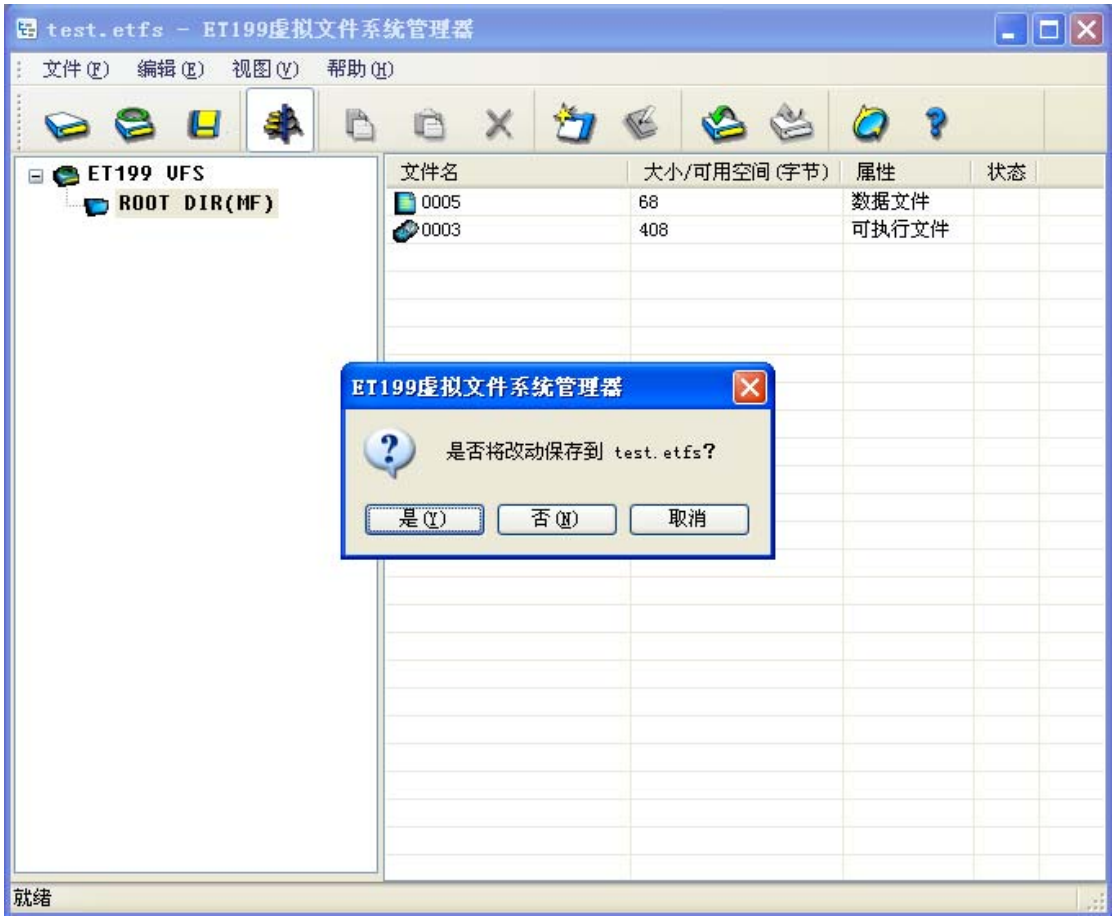
可以在这里导入文件，如导入数据文件 0x0005，和可执行文件 0x0003


test.etfs - ET199虚拟文件系统管理器
文件(F) 编辑(E) 视图(V) 帮助(H)
状态
导入
源文件(磁盘)
文件名称
D:\JD\test1\data.txt
目的文件(虚拟文件系统)
文件 ID
0005
文件大小
68
文件类型
可执行文件
可执行文件
内部可执行文件（不可写）
内部数据文件
公钥文件
私钥文件
文件ID为16
示创建卡内文件的大
小，为0表
称为空则将根据输入的
文件大小创建一个空文件。
创建(C)
取消(X)
就绪

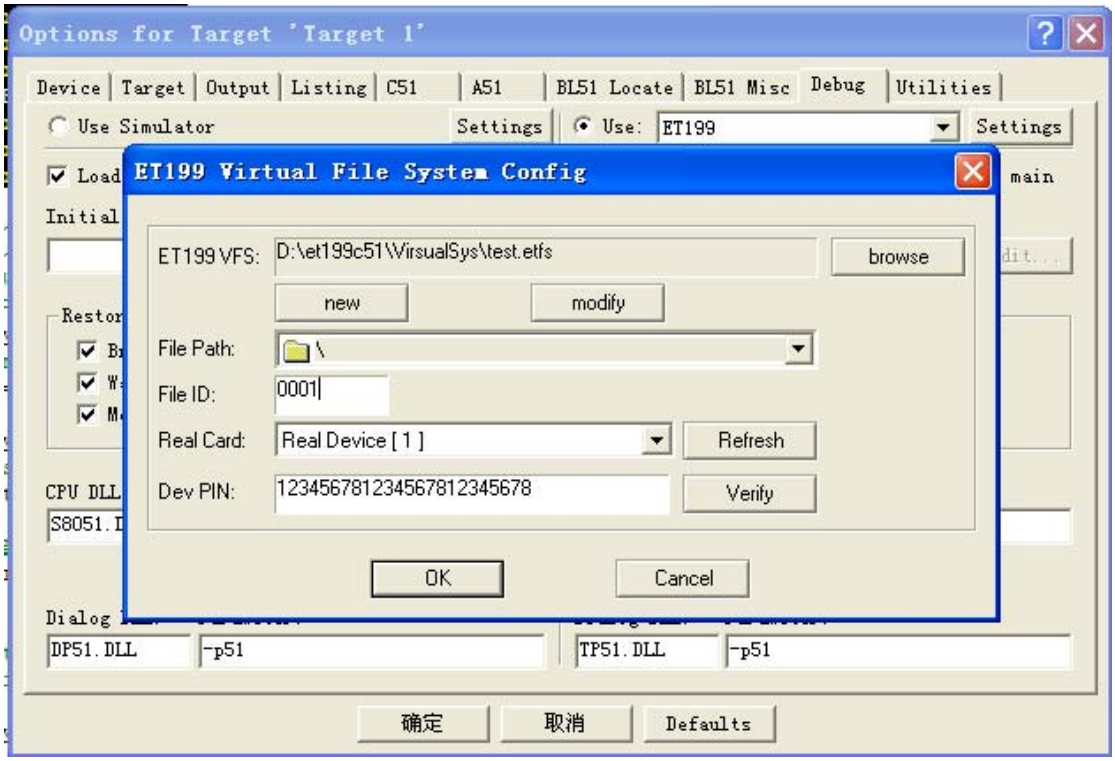

test.etfs - ET199虚拟文件系统管理器
文件(F) 编辑(E) 视图(V) 帮助(H)
ET199 UFS
ROOT DI
状态
导入
源文件(磁盘)
文件名称
D:\et199c51\sub.bin
...
目的文件(虚拟文件系统)
文件 ID
0003
文件大小
408
文件类型
可执行文件
文件ID为16进制数，范围为1-FFFF。文件大小表示创建卡内文件的大小，为0表示使用磁盘文件的大小。如果文件名称为空则将根据输入的文件大小创建一个空文件。
创建(C)
取消(X)
就绪



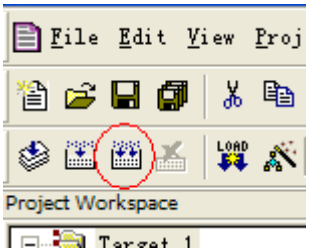
这时关闭 VfsSet.exe 工具，并保存 test.etfs



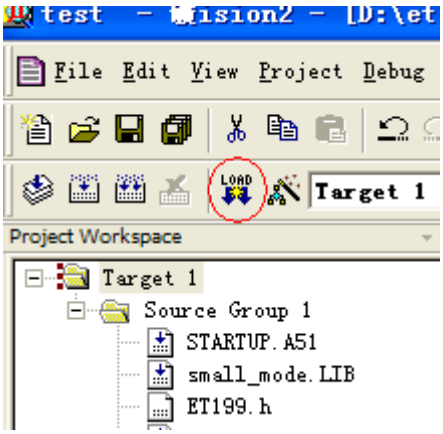
如果这时正在编译的 C51 为 0x0001 可执行文件



那么这时编译成功后，



下载到硬件中



这时会先将锁内所有数据清空，然后将 0x0001 下载到 test.etfs 中，然后再将 test.etfs 下载到锁内。那么锁内就有 0x0001，0x0003 可执行文件和 0x0005 数据文件，这时在打开 test.etfs 虚拟系统，与锁内是一致的。