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:



## 第二次结果



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。

🍃 ET199 加密锁设置	工具	
初始化	输入数据 用户密码: <sup>★★★★★★★★ 文件路径: \0001</sup>	
下載文件	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(E)
PIN码管理 PN码管理 交组对管理	輸出数据 0000 ❶0 04 00 00 00 00 00 00 00 00 00 00 00 00	返回状态: 成功,返回长度 8
₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩	↓ · · · · · · · · · · · · · · · · · · ·	的长度";如果无

```
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;</pre>
```

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。

🛛 Vse Si	mulator	1	Settings	🛈 Use: ET	199		Settings
V Load A	Application a	t Sta 🔽 Go	till main(	V Load Ap	plication at St	a 🔽 Go	till main
nitial	FT199 Viri	ual File Sy	sten Confi	σ			
				- P			di t
Restor	ET199 VFS:	D:\et199c51\virsy	vs\jd1.etfs			browse	1
₩ B:		new		modify			-
I∼ M	File Path:				-		
1.000	File ID:	0001					-
PV DLL	Real Card:	Real Device [1]		-	Refresh		
S8051.I	Day PIN:	12345678123456	7812345678		Marilu		
	Devrin.				veniy		
lialog			пк	Car	cel		
DP51.DL					loor		

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。

另存为					? 🗙
保存在 (I):	🗀 VirsualSys		•	🗢 🗈 💣 🎟 •	
Recent					
重重					
我的文档					
影響					
変的市間					
网上邻居					
	文件名 (M):	test		•	保存(5)
	保存类型(1):	Virtual Files	(*.Etfs)	•	取消

🖫 test.etfs - ET199虚拟文件系	統管理器				
; 文件 @) 编辑 @) 视图 (Y) 帮助(	<u>H</u> )				
S S U 🕸 🗅	ΒX	🍎 🍯	🗳 🗳	0 ?	
🖃 🚭 ET199 UFS	文件名	大小	ヽ/可用空间(字节)	属性	状态
ROOT DIR(MF)					
<u>=+/4</u>					
<b></b> 就绪					

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

國 test.etfs - EI199虚拟文件系统管理器	_ 🗆 🗙
: 文件 (2) 编辑 (2) 视图 (2) 帮助 (3)	
□ C ET199 VFS 导入	状态
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	
文件ID为14公钥又件 示创建卡内文件的大 小,为0束1 <u>私钥又件</u> 称为空则将根据输入的 文件大小创建一个空文件。	
创建( <u>C</u> ) 取消区)	
就绪	

🖬 test.etfs - ET199虚拟文件系统管理器	
· 文件 (2) 编辑 (2) 视图 (Y) 帮助 (H)	
□ S ET199 UFS 号入 X态 T ROOT D H	
線文件(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 入小 405 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[〕 取消区)	
就绪	

🗟 test.etfs - ET199虚拟文件系	统管理番				
; 文件(E) 编辑(E) 视图(Y) 帮助(	Ð				
S S U 🔹 🗅	θX	🍎 🍯	🗳 🖄	0 ?	
🖃 🥌 ET199 UFS	文件名	大小	ヽ/可用空间(字节)	属性	状态
ROOT DIR(MF)	0005	68		数据文件	
	Ø 0003	408		可执行文件	
就绪					

这时关闭 VfsSet.exe 工具,并保存 test.etfs

醫 test.etfs - ET199虚拟文件系	统管理器				
: 文件(E) 编辑(E) 视图(Y) 帮助(E	D				
664 6	ΒX	🍎 🌾	6	0 ?	
ET199 UFS	文件名 ☐ 0005	大 68 408	小/可用空间 (字节) }	<b>属性</b> 数据文件 可执行文件	状态
	199虚拟文件系 2 是否将改武 <u>是 (1)</u>	先管理器 b保存到 test. 至(2)	etfs? 取消		
就绪					

如果这时正在编译的 C51 为 0x0001 可执行文件

Options f	for Targe	t 'Iarget 1'	X
Device   Ta	arget   Output	t Listing C51   A51   BL51 Locate   BL51 Misc Debug   Utilities	
C Use Si	mulator	Settings 🕫 Use: ET199 💌 Setting	2
🔽 Load	ET199 Vir	tual File System Config 🛛 🚺 📷	a 🛛
Initial			
	ET199 VFS:	D:\et199c51\VirsualSys\test.etfs browse dit	
Restor		new modify	
🔽 Ba	File Path:		
<b>▼</b> *	File ID:	0001	
₩ M.	Real Card:	Real Device [1]	
CPH DII	Dev PIN <sup>.</sup>	123456781234567812345678	
S8051. I	Dorrint.	veiiiy	-
Dialog			
DP51. DLL	-p51	TP51. DLL -p51	-
	1		
		确定 取消 Defaults	

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。

🖫 test.etfs - ET199虚拟文件系	统管理番		
; 文件(E) 编辑(E) 视图(V) 帮助(J	D		
6 6 6 6 🔺 🗅	🖻 X 🆄 📽 🍪 🖉	\$	
🖃 🥌 ET199 UFS	文件名	大小/可用空间(字节) 属性	状态
TOT DIR(MF)	0005	68 数据文件	
	<i>∲</i> 0003	408 可执行文件	
	<b>∅</b> 0001	1968 可执行文件	
	<		
就绪		当前路径:\	乗