AN017 — 使用 C8051FXXX 片内 FLASH 编程接口 DLL

相关器件

本应用笔记适用于下列器件：

C8051F000、C8051F001、C8051F002、C8051F005、C8051F006、C8051F007、C8051F010、C8051F011、C8051F012、C8051F015、C8051F016、C8051F017、C8051F220、C8051F221、C8051F226、C8051F230、C8051F231 和 C8051F236。

1 引言

FLASH编程接口DLL（动态链接库）提供下述功能：下载Intel hex文件到FLASH；与一个C8051Fxxx处理器连接或断开连接；“运行”和“停止”微处理器；读和写内部、外部及程序存储器空间。所有这些功能都是通过PC机的COM端口和Cygnal串行适配器提供的。必须先调用Connect（连接）函数，以建立到C8051Fxxx目标板的通路。

本文所介绍的过程和原则说明如何将编程接口DLL链接到一个用户程序中。DLL不是可独立执行的应用程序，它只是一个在运行时才被链接的输出函数库，由一个“微软视窗”应用程序调用。

Cygnal提供FLASH编程接口DLL，同时还提供了一个使用FLASH编程接口DLL的应用程序（包括程序和源码），用户可以在不写自己的应用程序的情况下用它来测试或使用FLASH编程接口DLL。该FLASH编程接口应用程序采用隐含链接方式，要求将DLL放到一个特定的目录（见4.0节）。

2 文件及兼容性

从http://www.cygnal.com可以得到“CygUtil.dll”和“CygUtil.lib”的最近版本。该DLL是一个Win32 MFC常规DLL，这意味着它使用Microsoft Foundations Classes库；它可以被装入到任何Win32编程环境中，并且只输出‘C’类型函数。有两种版本：一种是MFC“Microsoft Foundations Classes”已被静态链接到DLL内部，另一种是MFC库被动态链接到DLL。

静态链接的MFC版本包含了它所需要的所有MFC库代码的拷贝，因此是自含式的，不需进行外部MFC链接。含有MFC库代码的静态链接型DLL大约为212KB。

动态链接MFC的版本大约为72KB。但是动态链接的DLL要求在目标机器上有文件“MFC42.dll”和“MSVCRT.dll”存在。如果客户程序也被链接到同一版本（4.2版）或更新的MFC库（即用MFC作为共享库），这不会成为问题。所需要的MFC DLL“MFC42.dll”和“MSVCRT.dll”随动态链接的MFC版FLASH编程接口DLL一起提供。如果在目标机器上已有这些文件的相同或更新版本，不要替换它们。

Silicon Laboratories Inc.
4635 Boston Lane
Austin, TX 78735
Email: mcuinfo@silabs.com
Internet: www.silabs.com

新华龙电子有限公司
深圳市福田区华强北路现代之窗大厦 A 座 13F C 室(518013)
电话：0755-83645240 83645242 83645244 83645251
电邮：shenzhen@xhl.com.cn
网址：www.xhl.com.cn

3 从客户程序调用 DLL 的输出函数

该DLL输出12个函数。下面是它们的原型及相关说明。

int Connect(int nComPort=1, BOOL bDisableDialogBoxes=0);
BOOL Connected();
int Disconnect(int nComPort=1);
int Download(char sDownloadFile, BOOL DeviceErase=0, BOOL DisableDialogBoxes=0);*
int SetTargetGo();
BOOL SetTargetHalt();
*int GetRAMMemory(BYTE * ptrMem, DWORD wStartAddress, unsigned int nLength);*
*int SetRAMMemory(BYTE * ptrMem, DWORD wStartAddress, unsigned int nLength);*
*int GetXRAMMemory(BYTE * ptrMem, DWORD wStartAddress, unsigned int nLength);*
*int SetXRAMMemory(BYTE * ptrMem, DWORD wStartAddress, unsigned int nLength);*
*int GetCodeMemory(BYTE * ptrMem, DWORD wStartAddress, unsigned int nLength);*
*int SetCodeMemory(BYTE * ptrMem, DWORD wStartAddress, unsigned int nLength,*
BOOL bDisableDialogs=0);

3.1 通信函数

int Connect(int nComPort=1, BOOL bDisableDialogBoxes=0);
BOOL Connected();
int Disconnect(int nComPort=1);

3.1.1 Connect()

Connect()函数返回一个整型值，该返回值的含义在5.0节介绍。Connect()函数接受两个缺省参数，参数类型为整型和布尔型。输入参数nComPort代表用于建立通信连接的COM端口。输入参数bDisableDialogBoxes是一个布尔值，表示允许（TRUE）或禁止（FALSE）DLL内部的对话框。注意，对于所有的存储器操作，都必须首先建立一个有效通信连接。

当使用C++时，Connect()函数必须被声明为一个导入函数。将下面一行加到调用该函数的源文件（*.cpp）之头文件（*.h）中：

extern "C" __declspec(dllexport) int Connect(int nComPort=1, BOOL bDisableDialogBoxes=0);

在源文件中，按下例所示调用该函数：

int r = Connect(nNewCOMPort, m_bDisableDialogs);

其中，nNewCOMPort和m_bDisableDialogs是在调用“Connect”函数之前已经被声明和初始化的变量。

3.1.2 Connected()

Connected()函数返回一个布尔值（FALSE表示未连接，TRUE表示已连接），该值代表目标C8051Fxxx的连接状态。

当使用C++时，Connected()函数必须被声明为一个导入函数。将下面一行加到调用该函数的

源文件（*.cpp）之头文件（*.h）中：

extern "C" __declspec(dllexport) BOOL Connected();

在源文件中，按下例所示调用该函数：

BOOL r = Connected();

3.1.3 Disconnect()

Disconnect()函数返回一个整型值，该返回值的含义在5.0节介绍。Disconnect()函数接受一个缺省参数（整型）。输入参数nComPort代表要断开通信连接的COM端口。

当使用C++时，Connect()函数必须被声明为一个导入函数。将下面一行加到调用该函数的源文件（*.cpp）之头文件（*.h）中：

extern "C" __declspec(dllexport) int Disconnect(int nComPort=1);

在源文件中，按下例所示调用该函数：

int r = Disconnect(m_nCOMPor);

其中，m_nCOMPort是在调用“Disconnect”函数之前已经被声明和初始化的变量。

3.2 程序接口函数

int Download(char sDownloadFile, BOOL DeviceErase=0, BOOL DisableDialogBoxes=0);*
int SetTargetGo();
BOOL SetTargetHalt();

3.2.1 Download()

Download()函数返回一个整型值，该返回值的含义在5.0节介绍。该函数接受三个参数（两个缺省参数）：char* sDownloadFile、BOOL DeviceErase和BOOL DisableDialogBoxes。输入参数sDownloadFile必须是一个字符指针，该指针指向含有下载文件绝对路径及文件名的字符数组（串）的开始处。输入参数DeviceErase是一个布尔值，在被设置为TRUE时执行器件擦除，如果被设置为FALSE，器件将不被擦除。一个器件擦除操作将擦除器件FLASH的全部内容。注意，在成功地从Download()函数退出后，目标C8051Fxxx将处于“停机”状态。如果器件被置于“停机”状态，它将不执行程序，直到产生一次上电复位或由SetTargetGo()函数调用产生复位。

当使用C++时，Download()函数必须被声明为一个导入函数。将下面一行加到调用该函数的源文件（*.cpp）之头文件（*.h）中：

extern "C" __declspec(dllexport) int Download(char sDownloadFile, BOOL bDeviceErase=0, BOOL bDisableDialogs=0);*

在源文件中，按下例所示调用该函数：

int r = Download(m_sDownloadFile, m_bDeviceErase, m_bDisableDialogs);

其中，m_sDownloadFile、m_bDeviceErase和m_bDisableDialogs是在调用“Download”函数之前已经被声明和初始化的变量。

3.2.2 SetTargetGo()

SetTargetGo()函数返回一个整型值，该返回值的含义在5.0节介绍。注意，在成功地从SetTargetGo()函数退出后，目标C8051Fxxx将处于“运行”状态。

当使用C++时，SetTargetGo()函数必须被声明为一个导入函数。将下面一行加到调用该函数的源文件（*.cpp）之头文件（*.h）中：

extern "C" __declspec(dllexport) int SetTargetGo ();

在源文件中，按下例所示调用该函数：

int r = SetTargetGo ();

3.2.3 SetTargetHalt()

SetTargetHalt()函数返回一个布尔值（FALSE表示目标器件不“停机”，TRUE表示目标器件已处于“停机”状态），该值代表是否成功地对目标C8051Fxxx执行了“停机”命令。

当使用C++时，SetTargetHalt()函数必须被声明为一个导入函数。将下面一行加到调用该函数的源文件（*.cpp）之头文件（*.h）中：

extern "C" __declspec(dllexport) BOOL SetTargetHalt ();

在源文件中，按下例所示调用该函数：

BOOL r = SetTargetHalt ();

3.3 读存储器函数

*int GetRAMMemory(BYTE * ptrMem, DWORD wStartAddress, unsigned int nLength);*
*int GetXRAMMemory(BYTE * ptrMem, DWORD wStartAddress, unsigned int nLength);*
*int GetCodeMemory(BYTE * ptrMem, DWORD wStartAddress, unsigned int nLength);*

GetRAMMemory()、GetXRAMMemory()和 GetCodeMemory()函数均为读存储器函数，所以放在一起讨论。所有读存储器函数都返回整型值，这些返回值的含义在 5.0 节介绍。所有读存储器函数都接受一个字节型指针作为第一个参数，该指针指向一个字节型数组的开始处。如果读存储器函数执行成功，变量 ptrMem 将包含要读的存储器单元。下面给出了一个如何用 C++对一个数组进行初始化的例子：

unsigned char ptrMem;*
ptrMem = new unsigned char[length]; // 假定已在其它地方对 length 进行了声明和赋值
//接下来对用将被写入存储器的字节对该数组进行填充

或者：

BYTE ptrMem[10] = {0x00}; // 必须在将数组传递给 DLL 之前对其初始化

所有读存储器函数都接受一个 DWORD 型的 wStartAddress 作为第二个参数，该参数为待读存储器的起始地址。所有读存储器函数都接受一个整型的 nLength 作为第三个参数，该参数应包含要从存储器中读取的字节数。

3.3.1 GetRAMMemory()

GetRAMMemory()函数读内部数据存储器。要读的RAM存储器必须位于目标器件的内部数据地址空间。当使用C++时，GetRAMMemory ()函数必须被声明为一个导入函数。将下面一行加到调用该函数的源文件（*.cpp）之头文件（*.h）中：

```
extern "C" __declspec(dllexport) int GetRAMMemory(BYTE * ptrMem, DWORD wStartAddress,
unsigned int nLength);
```

在源文件中，按下例所示调用该函数：

```
int r = GetRAMMemory(ptrBuf, m_wStartAt, m_nBytes);
```

其中，ptrBuf、m_wStartAt和m_nBytes是在调用“GetRAMMemory”函数之前已经被声明和初始化的变量。

3.3.2 GetXRAMMemory()

GetXRAMMemory()函数读外部数据存储器。要读的XRAM存储器必须位于目标器件的外部数据地址空间。要特别注意正确选择外部数据地址空间。当使用C++时，GetXRAMMemory()函数必须被声明为一个导入函数。将下面一行加到调用该函数的源文件（*.cpp）之头文件（*.h）中：

```
extern "C" __declspec(dllexport) int GetXRAMMemory(BYTE * ptrMem, DWORD wStartAddress,
unsigned int nLength);
```

在源文件中，按下例所示调用该函数：

```
int r = GetXRAMMemory(ptrBuf, m_wStartAt, m_nBytes);
```

其中，ptrBuf、m_wStartAt和m_nBytes是在调用“GetXRAMMemory”函数之前已经被声明和初始化的变量。

3.3.3 GetCodeMemory()

GetCodeMemory()函数读程序存储器空间。要读的程序存储器必须位于目标器件的程序存储器空间。在读取被设置为读锁定的扇区时要特别注意。在读取被设置为读锁定的扇区时将总是返回0。还要注意，不允许读被保留的空间。读被保留的空间将总是返回错误。当使用C++时，GetCodeMemory()函数必须被声明为一个导入函数。将下面一行加到调用该函数的源文件（*.cpp）之头文件（*.h）中：

```
extern "C" __declspec(dllexport) int GetCodeMemory(BYTE * ptrMem, DWORD wStartAddress,
unsigned int nLength);
```

在源文件中，按下例所示调用该函数：

```
int r = GetCodeMemory(ptrBuf, m_wStartAt, m_nBytes);
```

其中，ptrBuf、m_wStartAt和m_nBytes是在调用“GetCodeMemory”函数之前已经被声明和初始化的变量。

3.4 写存储器函数

*int SetRAMMemory(BYTE * ptrMem, DWORD wStartAddress, unsigned int nLength);*
*int SetXRAMMemory(BYTE * ptrMem, DWORD wStartAddress, unsigned int nLength);*
*int SetCodeMemory(BYTE * ptrMem, DWORD wStartAddress, unsigned int nLength,*
BOOL bDisableDialogs=0);

SetRAMMemory()、SetXRAMMemory()和 SetCodeMemory()函数均为写存储器函数，所以放在一起讨论。所有写存储器函数都返回整型值，这些返回值的含义在 5.0 节介绍。所有写存储器函数都接受一个字节型指针作为第一个参数，该指针指向一个字节型数组的开始处，该数组包含 nLength 个元素并且在调用 DLL 的写存储器函数之前已被赋值。如果写存储器函数执行成功，说明已完成了对存储器的写入。

下面给出了一个如何用 C++对一个数组进行初始化的例子：

unsigned char ptrMem;*
ptrMem = new unsigned char[length]; // 假定已在其它地方对 length 进行了声明和赋值
//接下来对用将被写入存储器的字节对该数组进行填充

或者：
BYTE ptrMem[10] = {0x00}; // 必须在将数组传递给 DLL 之前对其初始化

所有写存储器函数都接受一个 DWORD 型的 wStartAddress 作为第二个参数，该参数为待写存储器的起始地址。所有写存储器函数都接受一个整型的 nLength 作为第三个参数，该参数应包含要向存储器写入的字节数。

3.4.1 SetRAMMemory()

SetRAMMemory()函数写内部数据存储器。目标RAM存储器必须位于目标器件的内部数据地址空间。当使用C++时，SetRAMMemory ()函数必须被声明为一个导入函数。将下面一行加到调用该函数的源文件（*.cpp）之头文件（*.h）中：

*extern "C" __declspec(dllexport) int SetRAMMemory(BYTE * ptrMem, DWORD wStartAddress, unsigned int nLength);*

在源文件中，按下例所示调用该函数：
int r = SetRAMMemory(ptrBuf, m_wStartAt, m_nBytes);

其中，ptrBuf、m_wStartAt和m_nBytes是在调用“SetRAMMemory”函数之前已经被声明和初始化的变量。

3.4.2 SetXRAMMemory()

SetXRAMMemory()函数写外部数据存储器。目标XRAM存储器必须位于目标器件的外部数据地址空间。要特别注意正确选择外部数据地址空间。当使用C++时，SetXRAMMemory ()函数必须被声明为一个导入函数。将下面一行加到调用该函数的源文件（*.cpp）之头文件（*.h）中：

*extern "C" __declspec(dllexport) int SetXRAMMemory(BYTE * ptrMem, DWORD wStartAddress,*

unsigned int nLength);

在源文件中，按下例所示调用该函数：

int r = SetXRAMMemory(ptrBuf, m_wStartAt, m_nBytes);

其中，ptrBuf、m_wStartAt和m_nBytes是在调用“SetXRAMMemory”函数之前已经被声明和初始化的变量。

3.4.3 SetCodeMemory()

SetCodeMemory()函数写程序存储器。该函数增加了一个布尔型参数bDisableDialogs。该参数决定是否显示DLL内部的对话框。BDisableDialogs参数的缺省值为FALSE。注意，如果用户程序写FLASH的保留区，或一次写的数据大于一页（512字节），或对已被设置为写/擦除锁定的扇区写入，则写操作不会成功。如果SetCodeMemory ()函数执行成功，表明指定范围（m_wStartAt + m_nLength）内的存储器单元被正确写入。

当使用C++时，SetCodeMemory ()函数必须被声明为一个导入函数。将下面一行加到调用该函数的源文件（*.cpp）之头文件（*.h）中：

*extern "C" __declspec(dllexport) int SetCodeMemory(BYTE * ptrMem, DWORD wStartAddress, unsigned int nLength, BOOL bDisableDialogs=0);*

在源文件中，按下例所示调用该函数：

int r = SetCodeMemory(ptrBuf, m_wStartAt, m_nBytes, m_bDisableDialogs);

其中，ptrBuf、m_wStartAt、m_nBytes和m_bDisableDialogs是在调用“SetCodeMemory”函数之前已经被声明和初始化的变量。

4 链接

如果使用显式链接，需要在生成客户可执行程序之前向链接器提供“CygUtil.lib”库文件的路径。在 Microsoft Visual C++中，这可以通过从 Project 菜单选择 *Settings...*然后再选择 *link* 标签来完成。然后在 Object/library modules 对话框中输入该库文件的完整路径和文件名。例如，“c:\project\release\ CygUtil.lib”。在客户程序生成后将不再需要该库文件。

如果该 DLL 被隐式链接，则 DLL 必须位于下列目录之一：

1. 含有客户 EXE 文件的目录。
2. 过程的当前目录。
3. Windows 系统目录。
4. Windows 目录。
5. 在环境变量 PATH 中列出的目录。

5 测试结果

在退出时，该 DLL 会返回一个整型值的返回码。如果在 DLL 执行期间出现致命错误，DLL 会显示一个说明错误的消息框（如果允许显示对话框），然后退出。下表给出了返回码及含义说明。

返回码

返回码	错误	状态	可能的原因
-3	FLASH 写错误	失败	写无效页，写 FLASH 保留区等
-2	目标器件状态错误	失败	目标器件不处于停机状态
-1	目标器件状态错误	失败	目标器件未连接
0	无错误	成功	函数调用成功
1	文件名或路径错误	失败	无效路径和/或文件不存在
2	COM 端口错误	失败	不能用所选 COM 端口建立连接
3	下载错误	失败	无效字节数，所要求的存储器操作不存在
4	复位错误	失败	目标器件不能执行复位过程；检查是否仍然保持连接
5	器件擦除错误	失败	目标器件不能执行擦除过程；检查写/擦除锁定字节；检查是否仍然保持连接
7	关闭 COM 端口错误	失败	不能与目标器件建立连接以关闭 COM 端口，检查是否仍然保持连接

6 使用限制

当在一个客户程序的调试（debug）方式调用 DLL 时，消息框可能工作不正确。消息框为客户程序提供了获取瞬时信息的一种手段，而这种信息又不能用其它方法得到。消息框支持一个过程指示器，为客户程序提供与存储器操作过程有关的信息。当从一个客户程序（在调试方式）进入 DLL 时，可能导致 DLL 错误理解用于显示对话框的窗口句柄。建议在调用 DLL 之前将函数中的 bDisableDialogs 参数值设置为 TRUE（布尔值 1），bDisableDialogs 参数的缺省值为 FALSE。该问题不会在客户程序的发行（release）版本中出现。