AN-1138
应用笔记

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

LINB DLL编程指南

作者：Holger Grothe

简介

本应用笔记说明LINBWSD.dll库提供的库函数。针对集成电池传感器，可以利用这些函数创建一个USB至LIN下载器。LINBWSD.dll采用Protocol 6，通过LIN进行Flash/EE存储器编程。关于Protocol 6的详细说明，请参阅应用笔记AN-946。LINBWSD.dll可以用于下列IBS器件的LIN编程：

- ADuC7032-8L
- ADuC7033
- ADuC7036DCPZ
- ADuC7039

免责声明

LIN下载器

图1. LIN下载器

目录

修订历史

2012年1月—修订版0：初始版

硬件设置

下载器跳线设置

下载器板可以连接两个接口：LIN和I²C。接口的LIN功能由板上跳线连接J5和J6决定，如图2所示。

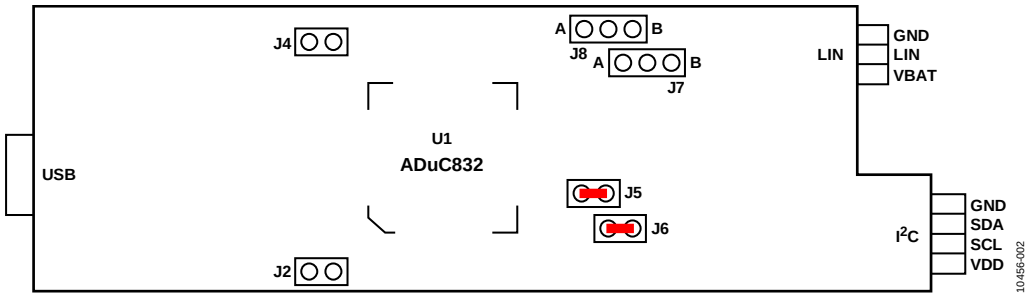

图2. LIN下载器跳线配置

下载序列

LINBWSD.dll包含一个安全序列器，用以防止Flash遭到破坏。使用DLL函数的正常下载序列如下：

- OpenDongle();
- GetDongleStatus();
- ReadDongleIdent();
- AssignNAD();
  - 如无默认设置，应使用NAD
- InitLINkernel();
- ImportHexData();
- GetImportInfo();
- DoPageErase();
  - GetImportInfo()获得的页数
- DoDataWrite();
  - GetImportInfo()获得的字节数
- DoWriteCRC();
- DoResetKernel();
  - 必要时使用
- CloseDongle();
  - 如果没有更多CPU需要编程
- ReadByIdentifier();

GetDLLInfo

获得DLL版本信息。

void **GetDLLInfo** (*char *pcDLLInfo, BYTE *pusStringLen*)

参数

表1.

参数	描述
* *pcDLLInfo*	指向一个字符缓冲器的指针，该字符缓冲器包含DLL提供的版本信息。
**pusStringLen*	指向一个字节型变量的指针，该变量接收ASCII数据的长度。

返回值

无返回值。

备注

关闭应用程序之前，应用程序须先关闭已打开的USB端口。如果发生错误，DLL会打开一个消息框。

示例

```
char *pMessage;
BYTE usStrLen;
BYTE usResult;
CString strMessage;


pMessage = strMessage.GetBuffer(100);
GetDLLInfo(pMessage,&usStrLen);
strMessage.ReleaseBuffer();
strMessage.SetAt(usStrLen,'\0');
MessageBox(strMessage);
```

OpenDongle

打开USB端口并寻找LIN-Dongle。

BYTE **OpenDongle** (char *pcOpenPortError, BYTE *pusStringLen)

参数

表2.

参数	描述
**pcOpenPortError*	指向一个字符缓冲器的指针，该字符缓冲器包含ASCII数据。
**pusStringLen*	指向一个字节型变量的指针，该变量接收ASCII数据的长度。

返回值

OpenDongle的返回值如下：

- 0 = 无错误。
- 1 = 打开USB端口时出错或未连接LIN-Dongle。

备注

如果应用程序试图打开一个已经打开的端口，DLL会忽略该命令。

示例

```
char *pMessage;
BYTE usStrLen;
BYTE usResult;
CString strMessage;

pMessage = strMessage.GetBuffer(100);
usResult = OpenDongle(pMessage,&usStrLen);
strMessage.ReleaseBuffer();
strMessage.SetAt(usStrLen,'\0');
if(usResult != 0)             // Error
{
        MessageBox(strMessage);
        return;
}
```

CloseDongle

关闭USB端口。

BYTE **CloseDongle()**

参数

无参数。

返回值

CloseDongle的返回值如下：

- 0 = 无错误。
- 1 = 关闭USB端口时出错。

备注

关闭应用程序之前，应用程序须先关闭已打开的USB端口。如果发生错误，DLL会打开一个消息框。

示例

```
BYTE usResult;

usResult = closeDongle();
```

GetDongleStatus

启动Dongle固件，测试Dongle的硬件和软件版本。

BYTE **GetDongleStatus** (*char *pcOpenPortError, BYTE *pusStringLen*)

参数

表3.

参数	描述
**pcOpenPortError*	指向一个字符缓冲器的指针，该字符缓冲器包含ASCII数据。
**pusStringLen*	指向一个字节型变量的指针，该变量接收ASCII数据的长度。

返回值

GetDongleStatus的返回值如下：

- 0 = 无错误。
- 1 = Dongle初始化时出错或Dongle不正确。

备注

无备注。

示例

```
char *pMessage;
BYTE usStrLen;
BYTE usResult;
CString strMessage;

pMessage = strMessage.GetBuffer(100);
usResult = GetDongleStatus(pMessage,&usStrLen);
strMessage.ReleaseBuffer();
strMessage.SetAt(usStrLen,'\0');
if(usResult != 0) // Error
{
        MessageBox(strMessage);
        return;
}
```

DoReadDongleIdent

读取Dongle固件信息。

BYTE **DoReadDongleIdent** *(1, char *pcDongleIdent, BYTE *pusStringLen*)

参数

表4.

参数	描述
1	第一个参数必须是1，不得更改。
**pcDongleIdent*	指向一个字符缓冲器的指针，该字符缓冲器包含Dongle提供的ASCII形式硬件和软件信息。
**pusStringLen*	指向一个字节型变量的指针，该变量接收ASCII数据的长度。

返回值

DoReadDongleIdent的返回值如下：

- 0 = 无错误。
- 1 = 获取Dongle信息时出错。

备注

无备注。

示例

```
char *pMessage;
BYTE usStrLen;
BYTE usResult;
CString strMessage;

pMessage = strMessage.GetBuffer(100);
usResult = DoReadDongleIdent(1,pMessage,&usStrLen);
strMessage.ReleaseBuffer();
strMessage.SetAt(usStrLen,'\0');
if(usResult != 0)
{
        MessageBox(strMessage);
        return
}
```

InitLinKernel

连接到内核并获取其识别信息。

BYTE **InitLinKernel** (char **pcLinInitError, BYTE *pusStringLen, BYTE *pusCPU_Ident)*

参数

表5.

参数	描述
**pcLinInitError*	指向一个字符缓冲器的指针，该字符缓冲器包含ASCII数据。
**pusStringLen*	指向一个字节型变量的指针，该变量接收ASCII数据的长度。
**pusCPU_Ident*	指向一个字节型变量的指针，该变量接收CPU识别信息。

返回值

InitLinKernel的返回值如下：

- 0 = 无错误。
- 1 = 初始化时出错或未连接LIN-Dongle。
- 2 = 检测到错误的CPU签名。

备注

如果GetDongleStatus()发生故障，应用程序不应使用此函数。

如果应用程序不使用默认NAD，则在调用InitLinKernel之前，应先调用AssignNAD()。

示例

```
char *pMessage;
BYTE usStrLen;
BYTE usResult;
BYTE usCPU_Ident;
CString strMessage;

pMessage = strMessage.GetBuffer(100);
usResult = InitLinKernel(pMessage,&usStrLen,&usCPU_Ident);
strMessage.ReleaseBuffer();
strMessage.SetAt(usStrLen,'\0');
if(usResult != 0) /           / Error
{
        MessageBox(strMessage);
        return;
}
```

ImportHexData

导入要编程的文件。

BYTE **ImportHexData** (*char *cHexFilePath, char *pcImportError, BYTE *pusStringLen*)

参数

表6.

参数	描述
**cHexFilePath*	十六进制文件的文件路径。
**pcImportError*	指向一个字符缓冲器的指针，该字符缓冲器包含ASCII数据。
**pusStringLen*	指向一个字节型变量的指针，该变量接收ASCII数据的长度。

返回值

ImportHexData的返回值如下：

- 0 = 无错误。
- 1 = 导入时出错。

备注

应用程序应利用GetImportInfo()获得关于导入数据的信息，从而进行擦除、编程和验证。

示例

```
char *pMessage;
char szFileName[1024];
BYTE usStrLen;
BYTE usResult;
CString strMessage;


pMessage = strMessage.GetBuffer(100);
usResult = ImportHexData(szFileName ,pMessage, &usStrLen);
strMessage.ReleaseBuffer();
strMessage.SetAt(usStrLen,'\0');
if(usResult != 0)            // Error
{
       MessageBox(strMessage);
       return;
}
```

GetImportInfo

获得关于导入文件的信息。

BYTE **GetImportInfo** (&*ulEndAdr*, &*ulHexBytes*, &*ucMaxPages*, &*ulAdrOffset*)

参数

表7.

参数	描述
&*ulEndAdr*	表示十六进制文件最高地址的变量
&*ulHexBytes*	表示导入的字节数的变量
&*ucMaxPages*	表示要擦除的页数的变量
&*ulAdrOffset*	用于编程的偏移(目前尚不需要)

返回值

无返回值。

备注

无备注。

示例

```
char *pMessage;
char szFileName[1024];
BYTE usStrLen;
BYTE usResult;
CString strMessage;

unsigned long ulEndAdr, ulHexBytes, ulAdrOffset;
BYTE ucMaxPages;


pMessage = strMessage.GetBuffer(100);
usResult = ImportHexData(szFileName ,pMessage, &usStrLen);
strMessage.ReleaseBuffer();
strMessage.SetAt(usStrLen,'\0');
if(usResult != 0) // Error
{
        MessageBox(strMessage);
        return;
}
else
{
        GetImportInfo(&ulEndAdr, &ulHexBytes, &ucMaxPages, &ulAdrOffset);
}
```

DoPageErase

擦除一页。

BYTE **DoPageErase** (*char *pcPageEraseError, BYTE *pusStringLen, BYTE *pusErasedPageNumber*)

参数

表8.

参数	描述
**pcPageEraseError*	指向一个字符缓冲器的指针，该字符缓冲器包含ASCII数据。
**pusStringLen*	指向一个字节型变量的指针，该变量接收ASCII数据的长度。
**pusErasedPageNumber*	指向一个字节型变量的指针，该变量接收要擦除的页数。

返回值

DoPageErase的返回值如下：

- 0 = 无错误。
- 1 = 擦除页面时出错。
- 2 = 应用程序试图擦除超过需要的页数。

备注

usErasedPageNumber始终为1。

示例

```
char *pMessage;
BYTE usStrLen;
BYTE usResult;
CString strMessage;


BYTE ucPage = 0;


do
{
        pMessage = strMessage.GetBuffer(100);
        usResult = DoPageErase(pMessage, &usStrLen, &usErasedPageNumber);
        strMessage.ReleaseBuffer();
        if(usResult == 1)
        {
                strMessage.SetAt(usStrLen,'\0');
                MessageBox(strMessage);
                return;
        }
        if(usResult == 2)
        {
                MessageBox(„No more pages to erase“);
                ucPage = ucMaxPages;
        }
        ucPage ++;
}
while(ucPage<ucMaxPages);
// ucMaxPage contains the value get by GetImportInfo()
```

DoDataWrite

写入数据到Flash，并验证写入的数据。

BYTE **DoDataWrite** (*char *pcWriteDataError, BYTE *pusStringLen, BYTE *pusDataLen*)

参数

表9.

参数	描述
**pcWriteDataError*	指向一个字符缓冲器的指针，该字符缓冲器包含ASCII数据。
**pusStringLen*	指向一个字节型变量的指针，该变量接收ASCII数据的长度。
**pusDataLen*	指向一个字节型变量的指针，该变量接收要写入的字节数。

返回值

DoDataWrite的返回值如下：

- 0 = 无错误。
- 1 = 写入数据时出错。

备注

如果未擦除全部页面，应用程序将无法写入数据。

示例

```
char *pMessage;
BYTE usStrLen;
BYTE usResult;
BYTE usDataLen;
CString strMessage;

unsigned long ulWrittenHexBytes = 0;

do
{
        pMessage = strMessage.GetBuffer(100);
        usResult = DoDataWrite(pMessage, &usStrLen, &usDataLen);
        strMessage.ReleaseBuffer();
        if(usResult == 1)
        {
                strMessage.SetAt(usStrLen,'\0');
                MessageBox(strMessage);
                return;
        }
        ulWrittenHexBytes += (unsigned long)usDataLen;
}
while(ulWrittenHexBytes < ulHexBytes);
// ulHexBytes contains the value get by GetImportInfo()
```

DoWriteCRC

写入校验和到0x80014。

BYTE **DoWriteCRC** (*char *pcWriteCRCError, BYTE *pusStringLen, usCRC_Val*)

参数

表10.

参数	描述
**pcWriteCRCError*	指向一个字符缓冲器的指针，该字符缓冲器包含ASCII数据。
**pusStringLen*	指向一个字节型变量的指针，该变量接收ASCII数据的长度。
usCRC_Val	0 = 写入十六进制导入的计算值 1 = 写入代码开发的值 2 = 写入已导入十六进制文件的值

返回值

DoWriteCRC的返回值如下：

- 0 = 无错误。
- 1 = 写入校验和时出错。

备注

如果编程和验证不成功，DLL会阻止此函数。

示例

```
char *pMessage;
BYTE usStrLen;
BYTE usResult;
CString strMessage;


pMessage = strMessage.GetBuffer(100);
usResult = DoWriteCRC(pMessage,&usStrLen, 1); // code development
strMessage.ReleaseBuffer();
strMessage.SetAt(usStrLen,'\0');
if(usResult != 0)             // Error
{
       MessageBox(strMessage);
       return;
}
```

DoResetKernel

CPU复位。

BYTE **DoResetKernel** (*char *pcResetKernelError, BYTE *pusStringLen*)

参数

表11.

参数	描述
**pcResetKernelError*	指向一个字符缓冲器的指针，该字符缓冲器包含ASCII数据。
**pusStringLen*	指向一个字节型变量的指针，该变量接收ASCII数据的长度。

返回值

DoResetKernel的返回值如下：

- 0 = 无错误。
- 1 = 内核复位时出错。

备注

无备注。

示例

```
char *pMessage;
BYTE usStrLen;
BYTE usResult;
CString strMessage;

pMessage = strMessage.GetBuffer(100);
usResult = DoResetKernel(pMessage,&usStrLen);
strMessage.ReleaseBuffer();
strMessage.SetAt(usStrLen,'\0');
if(usResult != 0) // Error
{
        MessageBox(strMessage);
        return;
}
```

AssignNAD

给内核指定一个新的NAD。

BYTE **AssignNAD** (*char *pcAssignNADError, BYTE *pusStringLen, BYTE usNewNAD*)

参数

表12.

参数	描述
**pcAssignNADError*	指向一个字符缓冲器的指针，该字符缓冲器包含ASCII数据。
**pusStringLen*	指向一个字节型变量的指针，该变量接收ASCII数据的长度。
usNewNAD	新NAD

返回值

AssignNAD的返回值如下：

- 0 = 无错误。
- 1 = 指定NAD时出错。

备注

无备注。

示例

```
char *pMessage;
BYTE usStrLen;
BYTE usResult;
CString strMessage;


pMessage = strMessage.GetBuffer(100);
usResult = AssignNAD(pMessage,&usStrLen);
strMessage.ReleaseBuffer();
strMessage.SetAt(usStrLen,'\0');
if(usResult != 0)            // Error
{
       MessageBox(strMessage);
       return;
}
```

ReadByIdentifier

读取识别符。

BYTE **ReadByIdentifier** (*char *pcRBIError, BYTE *pusStringLen, BYTE usIdentifier*)

参数

表13.

参数	描述
**pcRBIError*	指向一个字符缓冲器的指针，该字符缓冲器包含ASCII数据。
**pusStringLen*	指向一个字节型变量的指针，该变量接收ASCII数据的长度。
usIdentifier	0x0
	0x32
	0x33
	0x34

返回值

ReadByIdentifier的返回值如下：

- 0 = 无错误。
- 1 = 识别符读取时出错。

备注

无备注。

示例

```
char *pMessage;
BYTE usStrLen;
BYTE usResult;
CString strMessage;


pMessage = strMessage.GetBuffer(100);
usResult = ReadByIdentifier(pMessage,&usStrLen,0); // reading 0x0
strMessage.ReleaseBuffer();
strMessage.SetAt(usStrLen,'\0');
if(usResult != 0)             // Error
{
        MessageBox(strMessage);
        return;
}
```

注释

注释

AN10456sc-0-1/12(0)