AN10905

LPC1300的USB大容量存储类的片内驱动

Rev. 1 – 2010年1月18日 应用笔记

文档信息

信息	内容
关键字	LPC1300、USB、MSC、片内驱动、ROM、Cortex-M3、LPC-LINK、LPCXpresso、IAR LPC1343-SK、Keil MCB1000
摘要	该应用笔记介绍了如何使用基于Cortex-M3的微控制器LPC1300的片内USB驱动，实现一个简单的USB大容量存储类（USB Mass Storage Class，MSC）的设备。

版本历史记录

版本	日期	描述
1	20100118	初始版本

联络信息

更多信息，请访问：http://www.nxp.com

如欲了解NXP销售办公室地址，请发送电子邮件至：saleaddresses@nxp.com

法律免责声明适用于本文档所提供的所有信息
© NXP B.V. 2010. 版权所有

1. 简介

LPC1300微控制器家族基于ARM Cortex-M3架构，针对嵌入式应用，具有高级的集成外设和低功耗。LPC1300系列的外设包括高达32K的闪存，最大到8K的数据存储器，USB Device接口，一个UART，一个SSP控制器，SPI接口，一个I2C接口，八通道的10位ADC，四个通用计时器Timer/PWMs，和多达40个通用GPIO引脚。

另外还具有16KB的ROM。这个片内ROM包含了一个bootloader，作为闪存的用户固件API，可支持UART和USB闪存编程。闪存API通过一个简单的接口，实现了板级闪存编程的功能。这个USB API支持开发人际交互接口设备（Human Interface Devices，HID）和MSC设备。

该应用笔记包括了如下的内容：

- USB概述
- 片内USB驱动功能
- 片内USB驱动设置
- 使用usbmemrom示例

2. 片内USB驱动功能

片内USB驱动程序被固化在LPC1300系列上的片内ROM中。其有利于建立简单的USB设备，同时节省闪存空间。LPC1300上的片内USB驱动实现了HID和MSC设备。ROM驱动的功能使得使用更简化和容易。

HID类驱动是用于和USB主机的数据量适中的通讯（小于64KB每秒）。它可支持中断方式传输，允许PC主机轮询从设备。而MSC类驱动则是实现磁盘驱动器的文件读取和来自USB主机的写入。

Feature	ROM HID Driver	ROM MSC Driver
Interrupt Transfers (Data “Pushed” to PC)	Yes	No
Endpoints	Control, 1 in, 1 out	Control, 1 in, 1 out
Real-time Data Transfers	Yes	No
File read/write	No	Yes
Supported clock	12 MHz external crystal[1]	12 MHz external crystal[1]
RAM Usage	First 384 bytes	First 384 bytes + Storage

表1. 片内USB驱动功能

[1] 采用12 MHz的外部晶振或者高精度的USB陶瓷谐振器是为了满足USB2.0规定的频率精度，即12Mbps，12.000Mb/s ±0.25% (2,500ppm)。

参见LPC1300用户手册中“USB驱动功能”一节。其详细的描述了USB驱动的功能，包括

法律免责声明适用于本文档所提供的所有信息

© NXP B.V. 2010. 版权所有

时钟和管脚的初始化，USB外设的初始化，USB接口和USB中断处理程序等。

该应用笔记详细描述了MSC驱动的。而应用笔记AN10904则是描述HID驱动。

3. 片内USB驱动设置

使用片内USB驱动需要几个步骤。以下的信息不太全面，可以参见用户手册第10章中的4.1节“USB大容量存储驱动”作为补充。

3.1 内存分配

片内USB驱动需要使用RAM空间——0x10000050到0x10000180，作为USB帧缓冲区。分配RAM的方法依赖于特定的开发环境，但是通常通过修改链接脚本和改变数据放置的地址范围来实现。由于通常链接器未被设计为具备智能分配算法，可管理极小的存储器区块。我们建议也保留从0x1000000到0x10000050的未分配RAM。如图1所示，以下各节描述了在Keil、IAR和LPCXpresso的环境下设置链接器的步骤。

法律免责声明适用于本文档所提供的所有信息
© NXP B.V. 2010. 版权所有

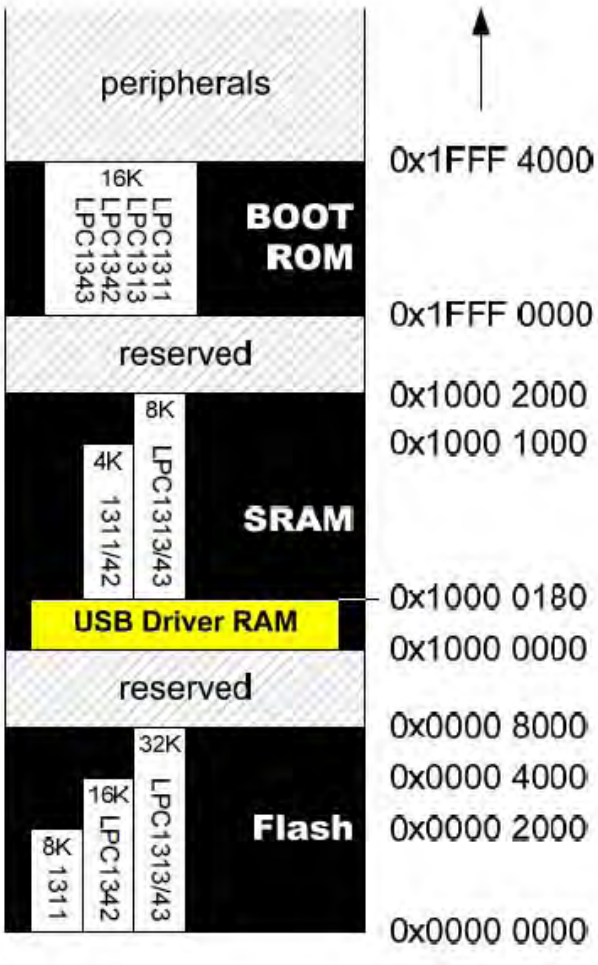

图1. LPC1300内存映射

3.1.1 Code Red的LPCXpresso

LPCXpresso通常生成链接脚本，自动根据当前选定的LPC微控制器匹配存储器映射。为了自定义链接脚本，LPCXpresso必须被配置为不能再次生成链接脚本。

1. 首先，创建并编译一个项目。这会创建一个匹配其选择的LPC1300芯片存储器配置的标准的链接脚本。这个链接脚本会有一个.ld扩展名的文件。它们将会生成到“project\build configuration”的目录下，或者“usbmemrom\Debug”目录下。
2. 通过重命名保存这个标准的链接脚本（我们选择命名为lpc1343_romusb_buffer），并移动到项目子目录下，以区别LPCXpresso工具自动生成的脚本。我们选择把他们放到lpcxpresso_too的子目录下，以帮助区分Keil和IAR开发工具的文件。
3. 修改lpc1343_romusb_buffer_mem.ld脚本，将存储器映射中的执行起点从0x1000000改到0x10000180。下面突出显示的改变区域——RamLoc8line即是为了LPC1343的具体示例。

法律免责声明适用于本文档所提供的所有信息
© NXP B.V. 2010. 版权所有

```
MEMORY
{
  /* Define each memory region */
  MFlash32 (rx) : ORIGIN = 0x0, LENGTH = 0x8000 /* 32k */
  RamLoc8 (rwx) : ORIGIN = 0x10000180, LENGTH = 0x1E80 /*
8k */
}
```

4. 修改主链接脚本的包含路径。通常有三个脚本，usb_buffer_lid.ld、usb_buffer_mem.ld和usb_buffer.ld。修改lpc1343_romusb_buffer.ld脚本的INCLUDE行，改为正确的路径，指向lpc1343_romusb_buffer_lib.ld和lpc1343_romusb_buffer_mem.ld。如下所示。

```
 * (created from nxp_lpc13_c.ld (v3.0.6 (200911181345)) on
Fri Nov 20 17:14:35 PST 2009)
*/

INCLUDE "../lpcxpreso tool/lpc1343 romusb buffer lib.ld"
INCLUDE "../lpcxpreso_tool/lpc1343_romusb_buffer_mem.ld"

ENTRY(ResetISR)

SECTIONS
{
...
```

5. 配置LPCXpresso，使用修改后的链接脚本。这是通过C/C++ Builed Setting下的Project Properties对话框设置。现在确保Tool Setting选项页被选中。然后选择MCU Linker Target。取消选中“Manager linker script”，并把链接脚本路径放入链接脚本文本字段中。这路径应该和Project Debug或者Release output目录相关，指向主链接脚本，其中一个不带有_lib或者_mem的名字。这个示例里面是lpc1343_romusb_buffer_mem.ld。

法律免责声明适用于本文档所提供的所有信息
© NXP B.V. 2010. 版权所有

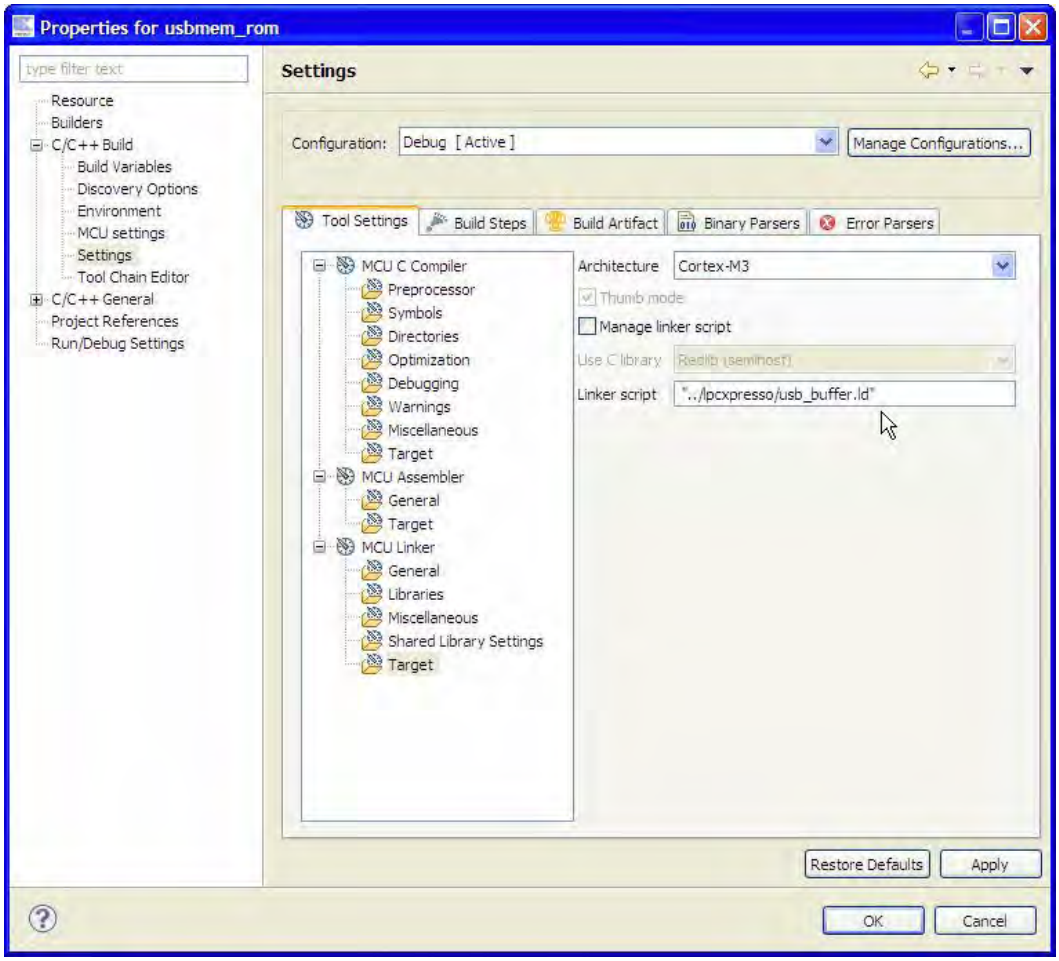

图2. LPCXpresso项目属性链接脚本设置

3.1.2 IAR Embedded Workbench IDE 5.4

IAR的项目通常已经引用了部分指定的链接脚本。这个链接脚本的路径可以从Linker category中Config选项卡下的Project Options对话框中找到。要分配为了片内USB驱动程序的RAM区域，需单击Project Options对话框的Edit按钮。使用Linker configuration file editor中的Memory Regions选项，修改RAM区域的起始地址为0x10000180。

法律免责声明适用于本文档所提供的所有信息
© NXP B.V. 2010. 版权所有

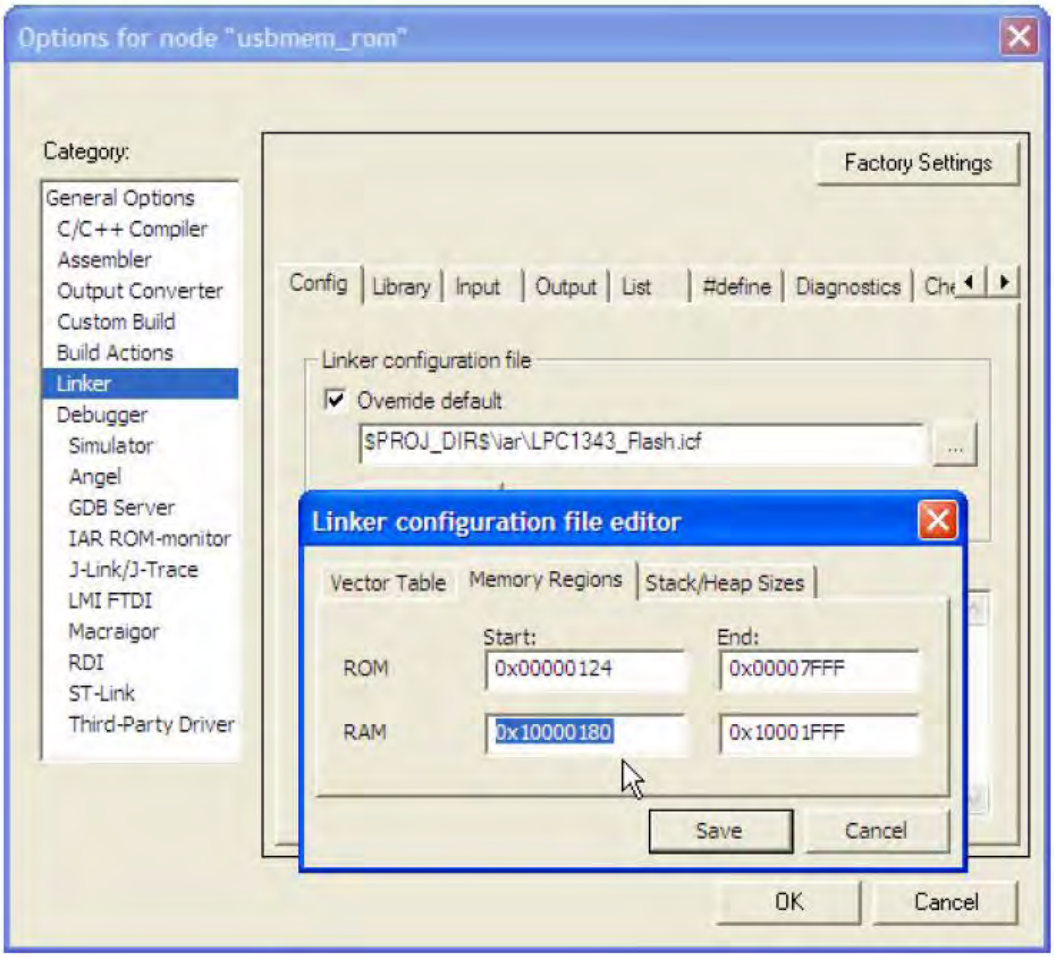

图3. IAR Embedded Workbench项目选项链接脚本设置

3.1.3 Keil uVision4 RealView MDK-ARM

在Keil uVision4中，使用Project Options对话框中的Target选项，改变RAM区域。设置起始地址为0x10000180，并把大小调整到0x180。

法律免责声明适用于本文档所提供的所有信息
© NXP B.V. 2010. 版权所有

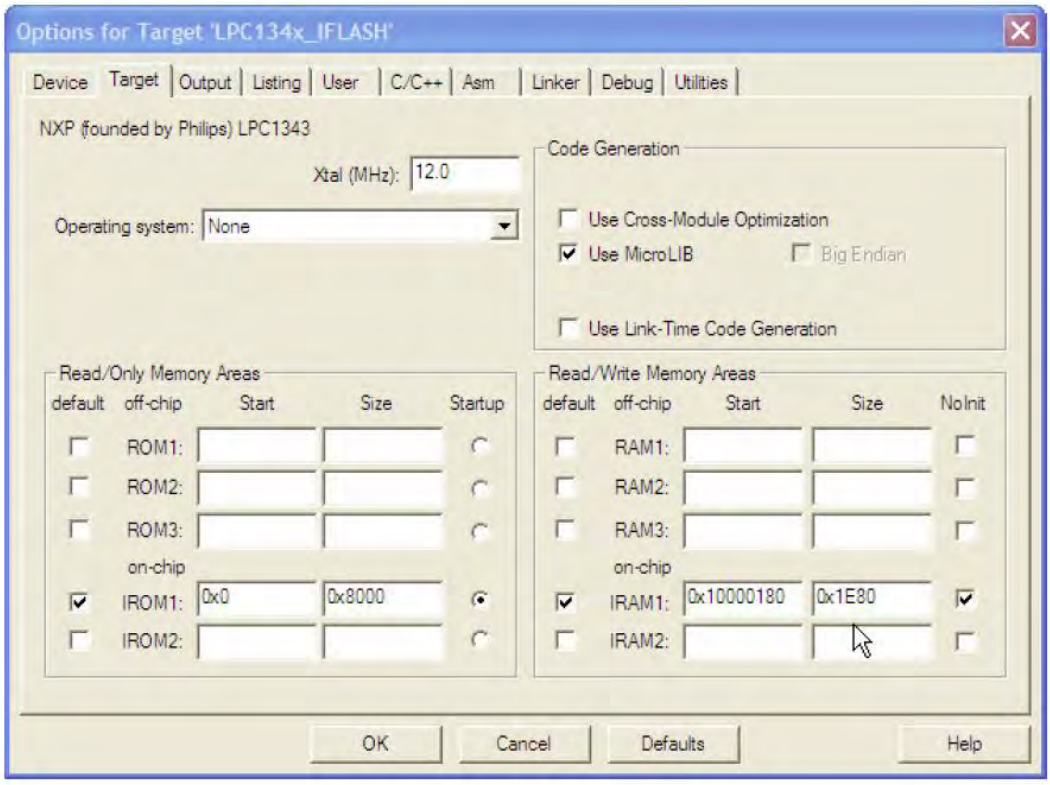

图4. Keil uVision4 RealView MDK-ARM目标选项链接脚本设置

3.2 ROM初始化

LPC1300片内ROM总是在复位后执行。这是在一个已知状态下芯片启动的关键。但你在开发调试代码时，重要的是确保片内ROM在用户代码前被执行。通常，使用调试宏或者脚本文件可以确保这一情况。多数情况下，LPC1300开发工具带有兼容的调试宏，可确保ROM在启动时运行。如果对此你有任何问题，请询问你开发工具提供商。

3.3 调用片内USB驱动程序函数

USB驱动程序的API函数包含三个函数和一个中断处理程序。它们通过ROM中的一个跳转表来被调用。这个跳转表的位置可能因为新产品ROM改进而被改变，因此，它是通过一个固定地址指针访问。为了访问这些函数，必须要声明跳转表项和指向跳转表指针的指针。图5显示的是指针的设置。

法律免责声明适用于本文档所提供的所有信息
© NXP B.V. 2010. 版权所有

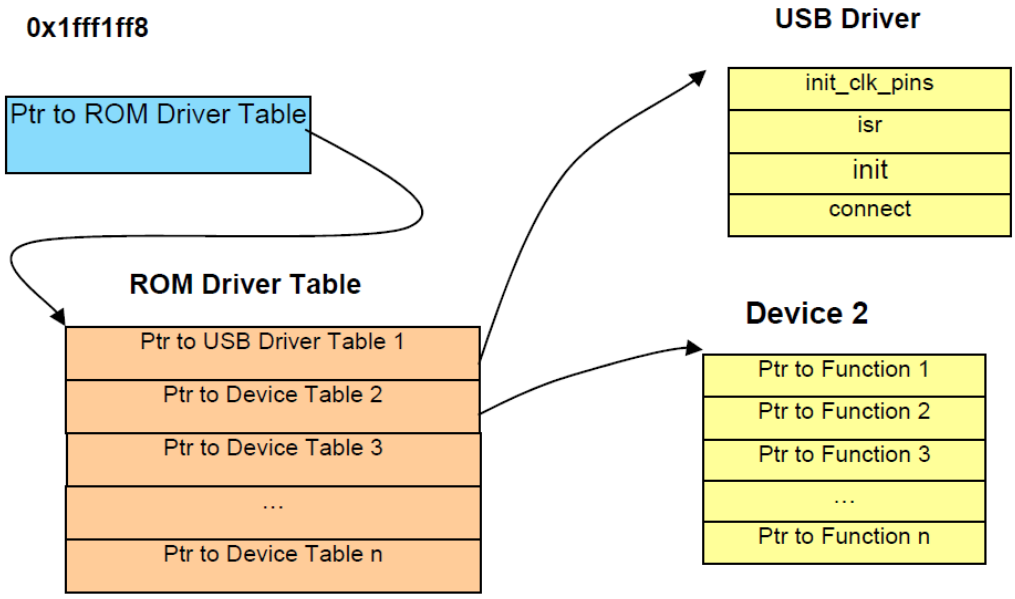

图5. 调用API的ROM布局

以下代码可以被采用：

```
typedef  struct _USBD {
  void       (*init_clk_pins)(void);
  void       (*isr)(void);
  void       (*init)( USB_DEV_INFO * DevInfoPtr );
  void       (*connect)(uint32_t  con);
}  USBD;

typedef  struct _ROM {
   const    USBD * pUSBD;
}  ROM;

ROM ** rom = (ROM **)0x1fff1ff8;
```

要调用其中的一个函数，可以采用以下的语法。这个代码取消引用位于0x1fff1ff8的为USB设备驱动的跳转表，然后使用跳转表调入USB驱动init_clk_pins()函数。

```
(*rom)->pUSBD->init_clk_pins();
```

3.4 配置USB驱动程序中断

片内USB驱动使用中断来响应LPC1300的USB控制器产生的事件。USB中断是在Cortex中断向量表中，其起始地址位于0x00000000。当应用程序收到中断时，它必须通过ROM中的USB程序的中断服务程序来处理。

在LPC1300上，用户代码是从Flash的0x00000000开始被编译执行。复位后，缺省情况下，片内ROM会替代Flash映射到0x00000000。当微控制器ROM中的初始化代码执行完毕，存储器会重新映射，Flash映射到0x00000000。ROM包含USB驱动的程序是在0x1FFF1000

法律免责声明适用于本文档所提供的所有信息
© NXP B.V. 2010. 版权所有

开始的。在这一点上，但0x00000000的内存被重新映射时，中断的控制会从ROM转移到用户闪存上。

为了确保ROM的USB中断处理程序被调用，在用户代码中必须声明中断会调用ROM中断服务程序。当使用Cortex相关的工业标准CMSIS头文件时，无论采用那种开发工具，中断处理程序应该看起来像下面的代码。

```
USB_IRQHandler(void)
{
  (*rom)->pUSBD->isr();
}
```

3.5 配置USB驱动数据结构

为了使用片内USB驱动的大容量存储类模式，必须建立几个结构体。首先必须声明USB_DEV_INFO，其成员DevType必须被初始化为USB_DEVICE_CLASS_STORAGE，而其成员DevDetailPtr需要被指向MSC_DEVICE_INFO结构体。在MSC_DEVICE_INFO结构体中，StrDescPtr必须被初始化为指向你的设备的USB字符串描述符。最重要的是必须定义读取和写入到本地存储的数据块的函数，初始化MSC_DEVICE_INFO并指向它们。在这个简单的示例中，我们将要读取和写入片内RAM的数据。另外，读取和写入外部SPI Flash存储器也可以实现。关于这些结构体的更多细节，请参阅用户手册的USB on-chip driver这一章节。

3.5.1 USB_DEVICE_INFO初始化

要指定我们打算初始化的大容量存储类模式的驱动程序，我们要设置DevType，并使DevDetailPtr指针指向带有MSC配置域的MSC_DEVICE_INFO结构体。

```
DeviceInfo.DevType = USB_DEVICE_CLASS_STORAGE;
DeviceInfo.DevDetailPtr = (uint32_t)&MscDevInfo;
```

3.5.2 MSC_DEVICE_INFO初始化

MSC_DEVICE_INFO结构体包含了需要配置的片内驱动的信息，实现大容量存储类USB外设。一些关键领域需要初始化，如供应商ID，产品ID，字符串描述符，块大小和数量。磁盘驱动器的区块的标准尺寸是512字节。而区块的数量则应该被设置为你的设备的块数量。MSC_MemorySize应该等于MSC_BlockSize *MSC_BlockCount。

```
MscDevInfo.idVendor = USB_VENDOR_ID;
MscDevInfo.idProduct = USB_PROD_ID;
MscDevInfo.StrDescPtr = (uint32_t)&USB_StringDescriptor[0];
MscDevInfo.BlockSize = MSC_BlockSize;
MscDevInfo.BlockCount = MSC_BlockCount;
MscDevInfo.MemorySize = MSC_MemorySize;
```

读取和写入数据到存储设备，需要使用两个函数。它们会传递字节偏移地址、一个小的RAM缓存和长度值。HID_DEVICE_INFO结构体中的MSC_Read和MSC_Write函数指针必须被初始化，指向这些函数。当PC要求读取和写入MSC设备时，片内USB驱动程序能调用这些函数，传输数据。

法律免责声明适用于本文档所提供的所有信息
© NXP B.V. 2010. 版权所有

```
MscDevInfo.MSC_Read = MSC_MemoryRead;
MscDevInfo.MSC_Write = MSC_MemoryWrite;
```

3.5.3 USB字符串描述符初始化

一个字符串描述符是一个双字节构成的字符数组，提供可读性的文本，以方便设备识别和安装过程。字符串描述符使用Unicode格式的双字节字符，运行表述全球多种语言。关于字符串描述符格式的更多细节，请参看USB规格书。

```
/* USB String Descriptor (optional) */
const uint8_t USB_StringDescriptor[] = {
  /* Index 0x00: LANGID Codes */
  0x04,                              /* bLength */
  USB_STRING_DESCRIPTOR_TYPE,        /* bDescriptorType */
  WBVAL(0x0409), /* US English */    /* wLANGID */
  /* Index 0x04: Manufacturer */
  0x1C,                              /* bLength */
  USB_STRING_DESCRIPTOR_TYPE,        /* bDescriptorType */
  'N',0,
  'X',0,
  'P',0,
  ' ',0,
  'S',0,
  'E',0,
  'M',0,
  'I',0,
  'C',0,
  'O',0,
  'N',0,
  'D',0,
  ' ',0,
…
```

3.6 调用设置函数

除了以上的设置，这一章节列出了初始化片内USB驱动，需要实际调用的函数。更多细节，请检查示例项目。

1. 使能32位Timer1
2. ROM驱动程序会使用32位Timer1作为内部时钟，其不能被应用程序使用。
3. 调用init_clk_pins()；
4. 调用init()；
5. 调用connect()；

4. 使用usbmemrom示例

此应用笔记介绍了如何使用软硬件，运行HID应用实例程序。请注意，NXP的LPCXpresso、IAR和Keil工具下的三种示例软件都会被提供。虽然C代码是相同的，针对特定的IDE环境，每种工作区都被定制化，可以轻松打开。下面列出的指定的开发板都经过测试。但是由于

法律免责声明适用于本文档所提供的所有信息 © NXP B.V. 2010. 版权所有

usbmemrom不使用任何外围设备功能，如LED，应该使用USB连接LPC1343开发板运行。

Software suite	JTAG probe	Target board
LPCXpresso by Code Red	LPC-LINK integrated on LPCXpresso board	LPCXpresso board and Embedded Artists LPCXpresso baseboard
LPCXpresso by Code Red	LPC-LINK integrated on LPCXpresso board	LPCXpresso board
IAR Embedded Workbench IDE 5.4	J-Link integrated on IAR LPC1343-SK board	IAR LPC1343-SK board
Keil µVision4 RealView MDK-ARM	ULINK2 by Keil	Keil MCB1000 board

表2. 工具组合

4.1 使用LPCXpresso IDE在LPCXpresso LPC1343开发板上运行usbmemrom程序

LPCXpresso开发板包含一个板上的LPC_LINK(USB的JTAG/SWO调试头)和一个LPC1343，所以不需要外部JTAG调试头。

图6. LPCXpresso LPC1343开发板

法律免责声明适用于本文档所提供的所有信息
© NXP B.V. 2010. 版权所有

4.1.1 使用带有Embedded Artists底板的LPCXpresso开发板

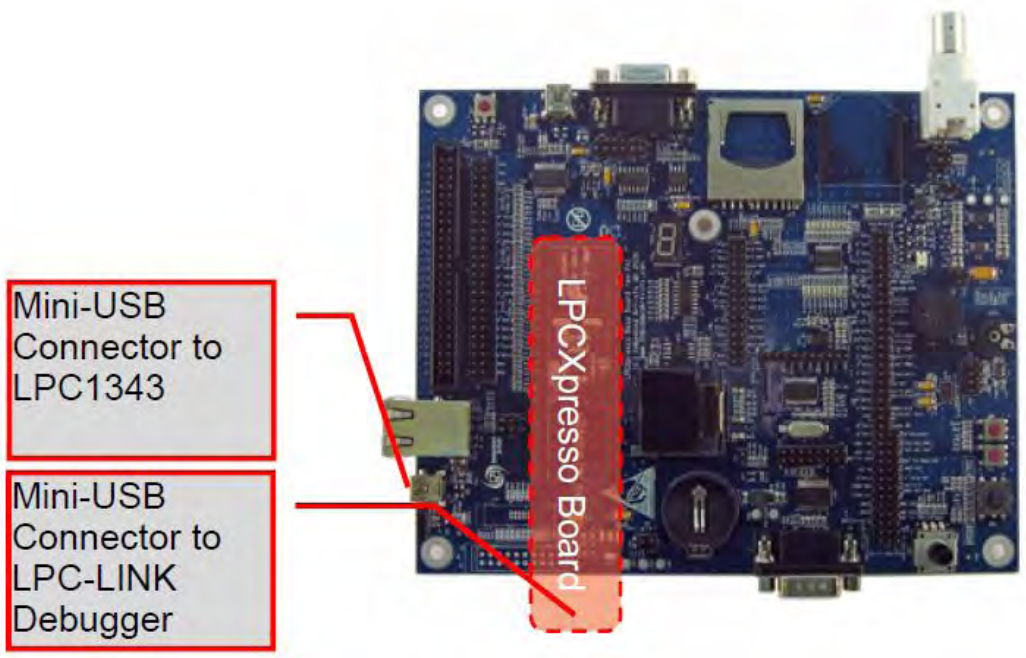

图7. Embedded Artists底板的USB连接

将0.1"头焊接到LPCXpresso开发板上，然后插入到EA的底板上。现在你可以使用mini-USB线缆把PC连接到底板上。通过设置EA底板上的一些跳线，可以使能USB。在写这篇文章时，需要连接跳线J14上的1和2针脚。J14是在底板的以太网接口附近。J12和J14都需要使引脚1和2连接。这些跳线位于电位器附近。EA底板上有很多的跳线设置，如果你有问题，可以参阅完整的Embedded Artists跳线的文档。

4.1.2 修改LPCXpresso开发板

使用Embedded Artists的LPCXpresso底板的另一个方法是修改LPCXpresso LPC1343开发板和增加USB线缆。由于LPC1343有一个USB PHY芯片，只需要一个上拉电阻连接到微控制器的USB端口。

注意：这个简单的连接不能实现NXP Soft-Connect，允许软断开和连接，也不能实现USB供电。正因为如此，必须在USB外设初始化以后，USB连接才能被插入到PC中。如果USB端口在LPC USB外设初始化以前连接，上拉电阻将会通知PC，USB设备存在，但由于没有初始化，微控制器USB外设不会响应。这将触发Windows提示USB设备产生一个错误。拔下并重新插入设备，这个错误会消失。

法律免责声明适用于本文档所提供的所有信息
© NXP B.V. 2010. 版权所有

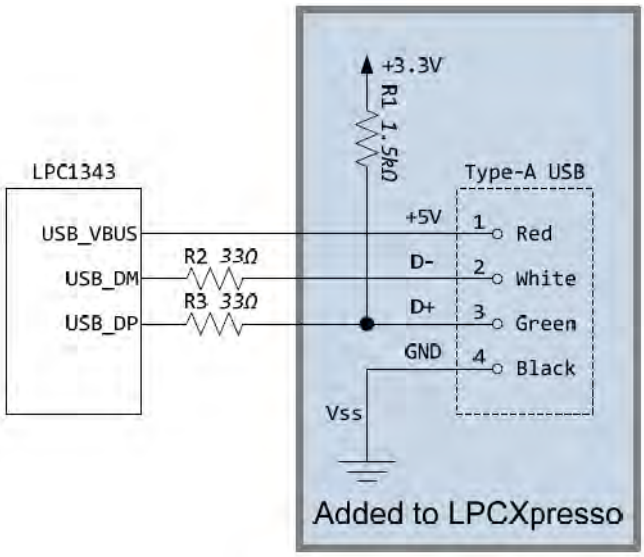

图8. LPCXpresso LPC1343 USB线缆修改原理图

注：不用制作USB Type-A接口的线缆或者电线，你可以利用现有A-B USB线缆，去掉它的B接口。然后，线缆的A接口可以剥离下来并焊接到LPCXpresso开发板上。

4.1.3 在LPCXpresso IDE上开始

解压本应用笔记中的示例项目。启动LPCXpresso IDE，使用在快速启动面板中的Import Example Projects链接，选择lpcxpresso_usbmemrom.zip。如果它不是在你的工作区中，请确保usbmemrom项目和CMSIS项目都同时导入。

使用mini USB线缆将LPCXpresso开发板上的LPC-LINK调试器连接到你开发的PC上。

如果你使用的是修改后的LPCXpresso USB开发板，则不用讲LPC1300的USB端口连接到PC上。因为电线上有1.5K的上拉电阻，PC会认为设备故障，其连接但无响应。在这个情形下，最好的方法是代码在初始化LPC1343 USB外设以后，才连接USB线缆。如果板上有NXP SoftConnet二极管，这会有问题。在Embedded Artists底板，Keil和IAR开发板上，都有SoftConnect二极管，因此一旦USB外设初始化完成，USB物理上连接以后，USB只会被PC枚举。

确保在LPCXpresso的当前项目中选择usbmemrom，然后在LPCXpresso快速启动面板中选择调试usbmemrom(Debug)。LPCXpresso应该编译项目，下载到目标板上，然后从main()的第一行开始运行。采用执行或者单步代码，直至调用函数Connnet(TRUE)。然后插上USB线缆，将修改的LPC1300开发板连接到PC上。如果声音开着，你应该可以听到PC去枚举LPC1300。

4.2 使用IAR Embedded Workbench在IAR LPC1343-SK开发板上运行usbmemrom

LPC1343-SK开发板包含一个板上的JLINK(USB的JTAG/SWO调试头)和一个LPC1343，所以不需要外部JTAG调试头。

法律免责声明适用于本文档所提供的所有信息
© NXP B.V. 2010. 版权所有

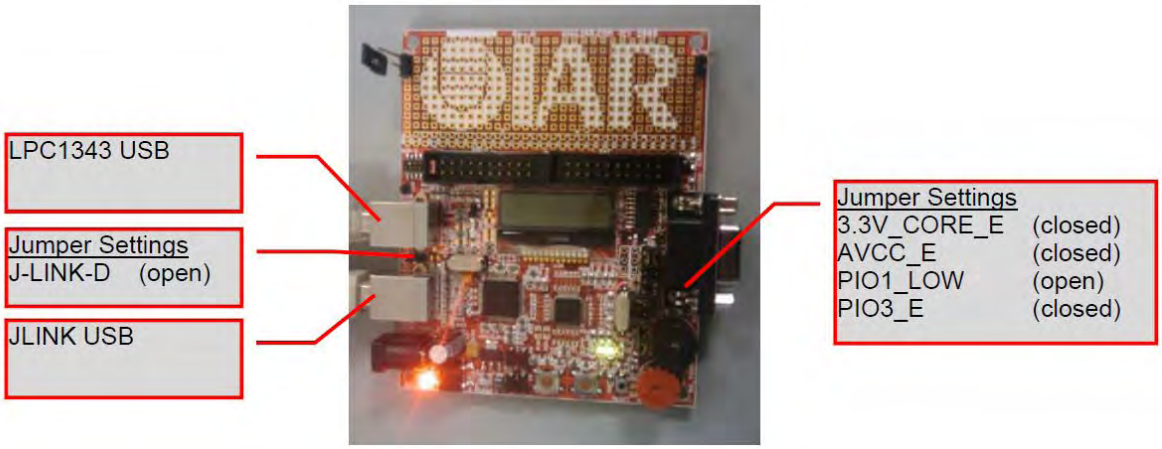

图9. IAR LPC1343-SK开发板

4.2.1 设置IAR LPC1343-SK开发板

LPC1343-SK开发板上有少数跳线很重要。首先，位于两个USB连接之间的跳线J_LINK_D，必须开发来使能板上的JLINK调试器。3.3V_CORE_E和AVCC_E跳线必须连接。PIO1_LOW必须开发。如果这个跳线连接了，那么LPC1343会在重启以后进入ISP模式，而不是执行Flash中的代码。PIO3_E必须被连接。PIO3_E允许连接USB的 VBUS。

4.2.2 在IAR Embedded Workbench IDE 5.4上开始usbmemrom

IAR LPC1343-SK开发板上有两个USB接口。一个标记为JLINK连接到集成的JTAG调试器。另一个标记为USB，是USB接口，用来连接LPC1343目标板。使用标准的USB电缆把它们两个都连接到你的PC。

现在，解压本应用笔记包括的示例项目。启动IAR Embedded Workbench IDE 5.4。打开File菜单中的Workspace选项，打开IAR版本的usbmemrom项目。从Project菜单中选择编译，然后在Project菜单中选择下载和调试。通过单步执行代码，直到调用connet(TRUE)函数被执行。如果声音打开，你应该可以听到PC枚举LPC1300 MSC设备。在此，你可以略过这一节“练习 usbmemrom”。

4.3 使用Keil uVision4 IDE在Keil MCB1000开发板上运行usbmemrom

为了调试MCB1000开发板，你需要将JTAG/SWO调试器如Keil ULINK2连接到开发板上。

法律免责声明适用于本文档所提供的所有信息
© NXP B.V. 2010. 版权所有

4.3.1 设置MCB1000开发板和ULINK2调试器

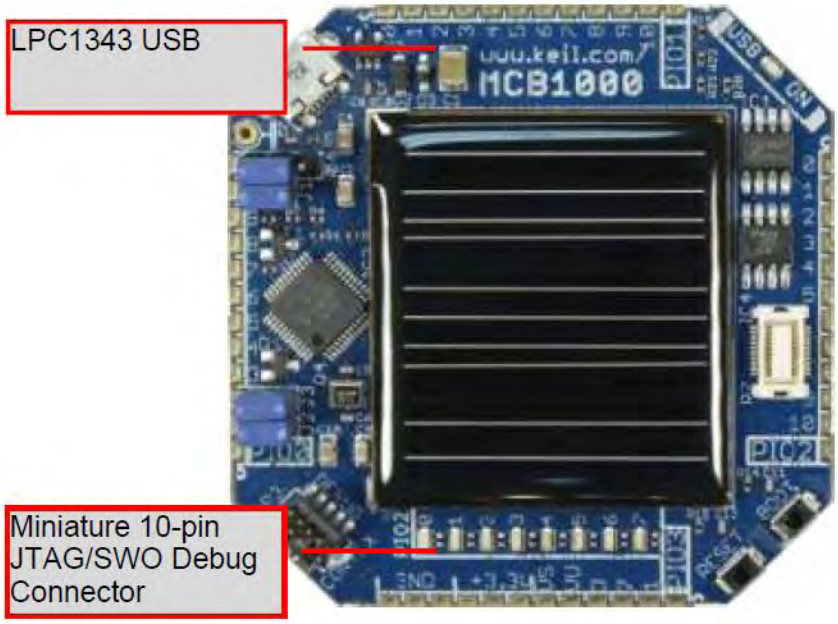

图10. Keil MCB1000开发板

使用标准USB线缆将ULINK连接你的PC，然后使用一个mini 10-pin的调试线缆把ULINK和Keil MCB1000目标板连接起来。在撰写本文时，出品的ULINK2接口只有一个大的20-pin的调试线缆。而适用于MCB1000开发板的小的10-pin线缆需要单独订购。要想连接到ULINK2，需要卸下底部的螺丝，打开塑料外壳。在ULINK2里面，有五种不同的类型的调试接口。插上mini 10-pin的线缆，就可以把它连接到MCB1000目标板上了。

图11. Keil ULINK2 JTAG/SWO调试器接口，去除外壳时

连接MCB1000到你的PC是使用一个标准的micro-USB的线缆。

4.3.2 在Keil uVision4 IDE上开始usbmemrom

现在，解压本应用笔记包括的示例项目。启动Keil uVision4 IDE，使用Project菜单中的Open

法律免责声明适用于本文档所提供的所有信息
© NXP B.V. 2010. 版权所有

Project选项，打开Keil usbmemrom项目。选择Project菜单中的Build Target，编译整个项目。然后，选择Flash菜单下的Download，对LPC1300微控制器进行编程。再选择Debug菜单下的Start/Stop Debug Session选项，进入调试模式。

通过单步执行代码，直到调用connet(TRUE)函数被执行。如果声音打开，你应该可以听到PC枚举LPC1300 MSC设备。在此，你可以略过这一节“练习 usbmemrom”。

4.4 练习usbmemrom

现在，LPC1343的usbmemrom示例应该运行在你的开发板上，并连接到PC。在Windows的资源浏览器中应该出现一个标记为LPC134x USB的磁盘驱动器。在驱动器上，有一个readme.txt的文件。

如果电脑正在运行Windows，它会显示一个关于“USB设备故障”的消息。这里会有一些故障排除提示，来解决问题。首先，打开设备管理器，你可以在Windows XP的控制面板中，硬件选项卡的系统对话框中找到它。在其他版本的Windows或者其他操作系统中，你不得不搜索该功能。确保在磁盘驱动器这一节下面，此设备被检测到。如果在设备管理器中的“人机接口设备（Human Interface Device）”下设备被检测到，这可能是因为其他示例项目已经在这台PC上运行过，而带有USB供应商和设备ID信息的HID驱动已经被安装到这个端口上。右键单击该设备，选择卸载。然后从电脑上拔下它并重新连接。Windows应该会自动重新安装正确的MSC USB设备驱动。

法律免责声明适用于本文档所提供的所有信息
© NXP B.V. 2010. 版权所有

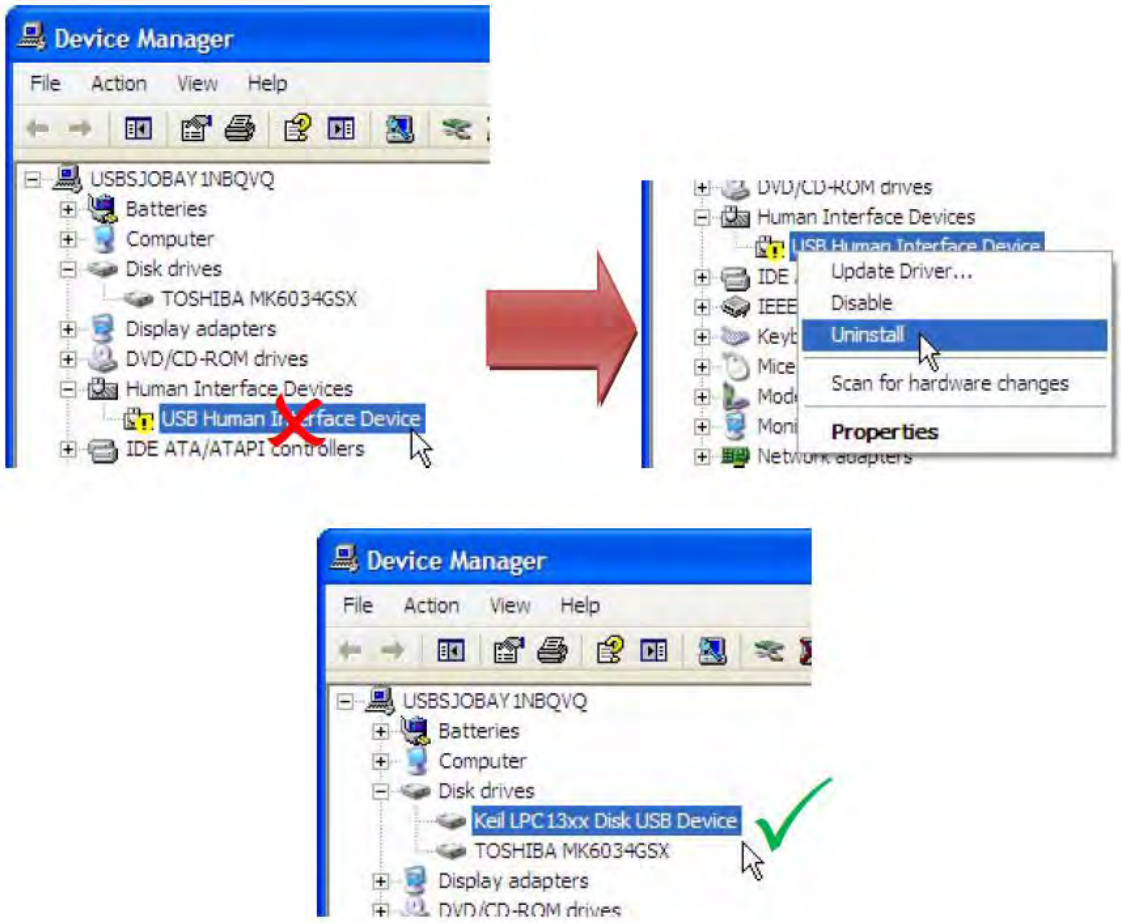

图12. Windows设备管理器 – 解决不正确的检测USB Device

法律免责声明适用于本文档所提供的所有信息

© NXP B.V. 2010. 版权所有

5. 结论

总之，在微控制器LPC1300系列中，片内USB驱动可以帮助轻松的建立一个简单的MSC设备。虽然内置的驱动程序非常简单，但它是十分有用的，能够在占用Flash内存最小的情况下，快速的建立一个MSC设备。

6. USB概述

6.1 什么是USB

USB代表通用串行总线。这是一个标准，设计为一个连接到PC的外设接口。该标准的管理由USB Inplementers Forum, Inc.公司负责，它和USB.org存在一些联系。设计USB驱动程序的关键是低成本、可热插拔，互操作性和易于使用。USB标准定义了布线的特点、标准化的链接器、总线供电的电器规范、硬件信号标准、通讯协议和应用配置文件。

USB已经变得无所不在。在2008年，已经安装了80亿的USB端口，并且销售量达到20亿。

6.2 USB总线的拓扑结构

在USB总线上，有一个USB主机，通常是PC。另外有一个或者多个USB设备，有时候还有一个或者多个USB集线器。每个USB连接都是点对点的，所有的通信都是由USB主机发起。在USB连接的上行端，是一个USB主机，或者USB主机的下行端口。而在USB连接到下行端，则是一个USB设备，或者一个USB集线器的上行端口。USB总线总共可以支持最多127个设备。这使得这个网络看起来像一棵树，因此被称为“星型”拓扑结构。

法律免责声明适用于本文档所提供的所有信息
© NXP B.V. 2010. 版权所有

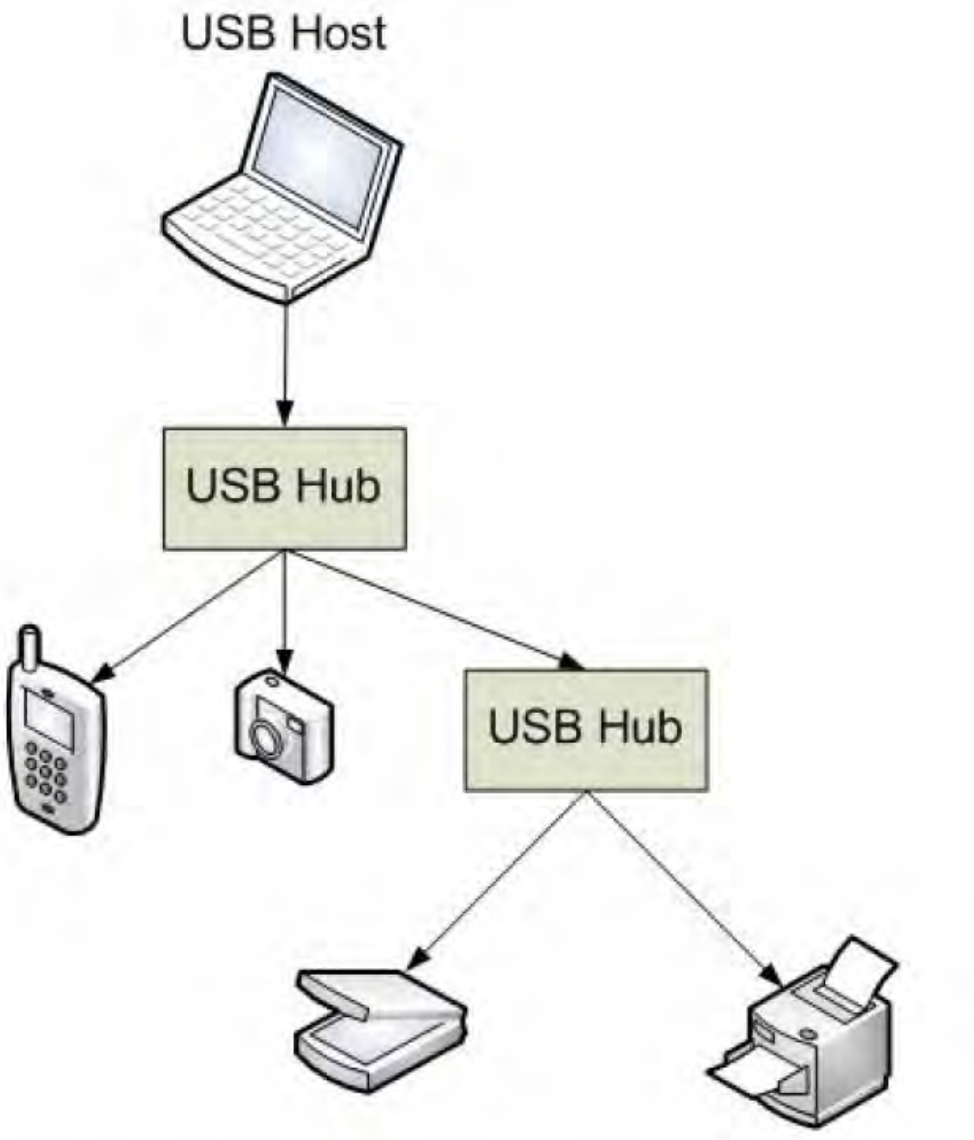

图13. USB总线拓扑结构 – 星型结构

6.3 USB总线的术语

USB接口由自己的术语。了解这些术语可以使设计USB产品更加容易。

6.3.1 设备类 Device Class

USB设备类是一个预先定义的配置文件，可简化产品开发。如果你能够在你的产品中使用标准的设备类，你就能够减少或者消除PC驱动程序和应用开发，并促进不同平台的兼容性，如Linux或Windows的未来版本。通用设备类包括HID（用于键盘和鼠标），MSC（用于USB磁盘驱动器和记忆棒）。

总之，在微控制器LPC1300系列中，片内USB驱动可以帮助轻松的建立一个简单的MSC设备。虽然内置的驱动程序非常简单，但它是十分有用的，能够在占用Flash内存最小的情况

法律免责声明适用于本文档所提供的所有信息
© NXP B.V. 2010. 版权所有

下，快速的建立一个MSC设备。

6.3.2 端点 Endpoint

USB端点就是一个缓冲器。它会被分配为一个从0到15的数值，和方向。IN端点是用于传输到USB主机的数据，而OUT端点是传输从主机出来的数据。IN和OUT端点的名称与主机的传输是相关的。一个USB设备上的IN端点实际将数据发送到主机，而不是从设备接受数据。两个相连的端点具有同样的数值和方向，被称为管道，作为一个单向的数据通道。例如，在USB设备上的EP 5 IN端点和USB主机上的EP 5 IN端点组合，就作为管道，传输设备到主机的数据。

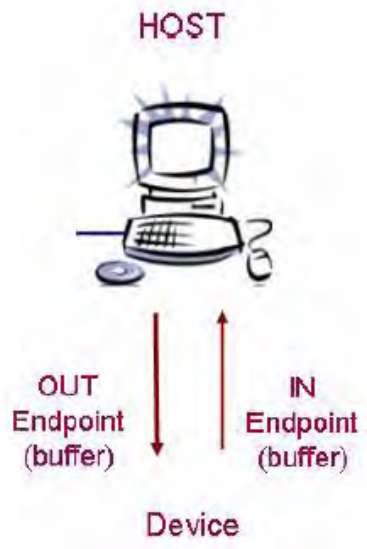

图14. USB IN和OUT端点的数据传输方向

6.3.3 描述符 Descriptor

USB描述符是一个静态的数据结构体，定义了一个USB的能力。当USB设备第一次连接上USB总线时，主机会读取设备的描述符。它描述了设备的制造商、产品类型、产品名称、端点数量和类型，以及设备类。

6.3.4 枚举 Enumeration

枚举是指在USB总线上发现USB设备，并读取其描述符的流程。之后，USB主机会启动一个流程去安装并实例化正确的USB设备驱动程序。

6.3.5 供应商ID (VID)和产品ID (PID)

VID和PID都是16位整数。每个USB产品必须靠一个唯一的VID和PID组合来识别，并通过USB认证。VID由USB实施者论坛USB-IF分配，在2009年12月2日的费用是2000美金。

法律免责声明适用于本文档所提供的所有信息
© NXP B.V. 2010. 版权所有

7. 免责声明

有限保修和责任－本文档中的信息被认为是准确和可靠的。然而，对于信息的准确性和完整性，恩智浦半导体公司不给予任何陈述或担保，明示或暗示，对于此类信息的使用后果不负任何责任。

在任何情况下，恩智浦半导体不会承担任何间接、意外发生、惩罚性、特别或相关性的损害赔偿（包括单不限于利润损失、储蓄损失、业务中断、有关去除或更换任何产品的费用或返工费用），不管这些损害赔偿是基于侵权（包括疏忽）、保修、违约合同或其他法律理论。

对于客户无论任何理由可能招致的任何损害，恩智浦半导体为在这里所提到的产品的汇总和累积责任应限制在恩智浦半导体商业销售的条款及条件里面。

变更的权利－恩智浦半导体有权在任何时间对此文件发布的信息(包括单不限于规格和产品说明）做出任何改动。本文件将取代所有之前所公布的信息。

适用性－恩智浦半导体产品并非为那些用于对生命和安全有重大关系的系统和设备而设计、授权或提供保证，也不用于那些可以合理预见到的因恩智浦半导体的产品的故障会造成人身伤害、甚至死亡、或是严重的财产或环境损害的应用程序中。恩智浦半导体的产品如果应用在此类的设备或应用程序中，恩智浦半导体对所此造成的风险将不承担任何责任，因此这些风险有客户自行承担。

应用－在这里所描述有关产品的任何应用程序仅用于说明的目的。在没有进一步的测试或修改的情况下，恩智浦半导体对该应用程序对指定用途是否合适不作任何表示或保证。

客户应对其使用恩智浦半导体产品的应用以及产品的设计和运行自行负责，恩智浦半导体不负责协助应用程序或客户的产品设计。同时，客户应自行负责决定恩智浦产品是否适合客户应用、计划产品、计划的应用程序以及第三方客户使用。客户应提供适当的设计和运行的保障措施以尽量减少其产品与应用的相关风险。

因客户的应用或产品的弱点或缺陷所产生的，或因使用其第三方客户的产品而产生的任何缺陷、损失、费用支出和问题，恩智浦半导体不承担任何责任。客户应负责为其使用恩智浦半导体芯片的产品或应用以及其第三方客户使用产品或应用做必要的测试，以避免使用不当而造成不必要的损失。恩智浦对在此方面不承担任何责任。

限制值－超过一个或多个限制值（如在IEC60134的绝对值最大额定义）的施压会对设备造成永久的损害。限制值只强调额定功率，这个设备的操作除了应用在此文件中所提到的“推荐工作条件”和“特征”部分之外，恩智浦半导体不担保超过上述要求的操作。恒定或反复超出限制值将永久地和不可逆转地影响设备的质量和可靠性。

商业销售条件－恩智浦半导体产品的销售适用公布于http://www.nxp.com/profile/terms网站上的通用商业销售条款，除非另存一个单独有效的书面协议，在此种情况下，将适用该单独有效的书面协议之条款和条件。关于客户采购恩智浦半导体产品，恩智浦半导体在此明确拒绝适用客户的通用条款和条件。

不构成任何出售要约或许可－本文中任何部分都不可被翻译或解释成可以开放接受或授予、转让或任何暗示许可版权、专利或其它工业或知识产权的销售产品要约。

出口控制－本文件以及其项目描述可能受出口管制条例限制。出口可能需事先获得国家机关许可。

非车规级产品－除非数据手册明确标出此恩智浦半导体产品为车规级，否则该产品不适合于汽车应用。该产品未在汽车产品测试和应用条件下经测试和质量认证。恩智浦半导体对客户将非车规产品运用在汽车设备和应用中不承担任何责任。

当客户使用该产品设计并使用在需要车规级规格和标准的汽车应用时，(1) 客户在该汽车应用、使用和规格中使用恩智浦半导体产品时，不在恩智浦半导体对该产品的保证范围内；(2) 当在汽车应用中使用超出恩智浦半导体规格的产品，客户应该自行承担风险；(3) 因客户超标准和产品规格使用恩智浦半导体产品导致的影响、损坏和失效产品索赔，客户不能要求恩智浦半导体进行赔偿。

法律免责声明适用于本文档所提供的所有信息

© NXP B.V. 2010. 版权所有

8. 目录

This translated version is for reference only, and the English version shall prevail in case of any discrepancy between the translated and English versions.

版权所有 2012恩智浦有限公司 未经许可，禁止转载