AN1388

PIC32 自举程序

作者： *Ganapathi Ramachandra*
Microchip Technology Inc.

简介

PIC32 器件的自举程序用于升级目标器件上的固件，无需外部编程器或调试器。

自举程序包括以下应用程序：

- 五个自举程序固件实现：
  - 通用异步收发器 （Universal Asynchronous Receiver Transmitter， UART）
  - 基于人机界面设备 （Human Interface Device，HID）类的通用串行总线 （Universal Serial Bus，USB）设备
  - 基于大容量存储设备 （Mass Storage Device，MSD）类的 USB 主机
  - 以太网
  - 安全数字 （Secure Digital， SD）卡
- 演示应用程序，可通过自举程序下载到目标 PIC32 器件中
- PC 主机应用程序（仅 UART、USB HID 和以太网自举程序需要），可与 PIC32 器件内运行的自举程序固件进行通信。此应用程序用于执行擦除和编程操作。

先决条件

运行自举应用程序的先决条件如下：

- 一台安装了 MPLAB® IDE v8.60 或更高版本，或安装了MPLAB X Beta v7.12或更高版本，并且安装了C32 编译器 v2.01 或更高版本的 PC
- 基于所选项目的开发硬件，如**表 9：“可用的自举程序工作区”**中所列
- 用于 UART 自举程序的 USB 一串口转换器（如果 PC 上没有 COM 端口）
- 用于 USB 大容量存储自举程序的 USB 闪存驱动器
- 用于 SD 卡自举程序的 SD 卡
- 用于以太网自举程序的以太网 （RJ-45）交叉电缆
- 传统编程工具，用于最初将自举程序固件写入 PIC32 器件 （如 MPLAB® REAL ICE™ 在线仿真系统或 MPLAB ICD 3 在线调试器）。PIC32 入门工具包不需要任何编程工具。

在使用 PIC32 自举程序之前，用户应该熟悉以下概念：

- PIC32 器件配置寄存器
- 编译和烧写 PIC32 器件
- PIC32 链接器脚本

自举程序的基本流程

图 3 中的流程图显示了自举应用程序的工作原理。自举程序代码在器件复位时开始执行。如果没有进入固件升级模式的条件，则自举程序开始执行用户应用程序。自举程序在固件升级模式下执行闪存擦除/编程操作。

进入固件升级模式

在器件复位时，如果擦除了用户应用程序复位向量地址的内容，自举程序会强制自己进入固件升级模式。要手动强制自举程序进入固件升级模式，请在上电期间按住 Explorer 16 开发板上的开关 S3。对于 PIC32 入门工具包，则是在上电期间按住开关 SW3。在固件升级模式下， Explorer 16 开发板上标记为 D5 的 LED 和 PIC32 入门工具包上标记为 LED3 的 LED 将闪烁。

退出固件升级模式

对于 USB HID、以太网或 UART 自举程序，可通过对器件应用硬复位或从 PC 发送 “跳转到应用程序” 命令来退出固件升级模式。对于 USB 闪存驱动器或 SD 卡的自举程序，通过硬复位或在固件烧写完成后退出固件升级模式。

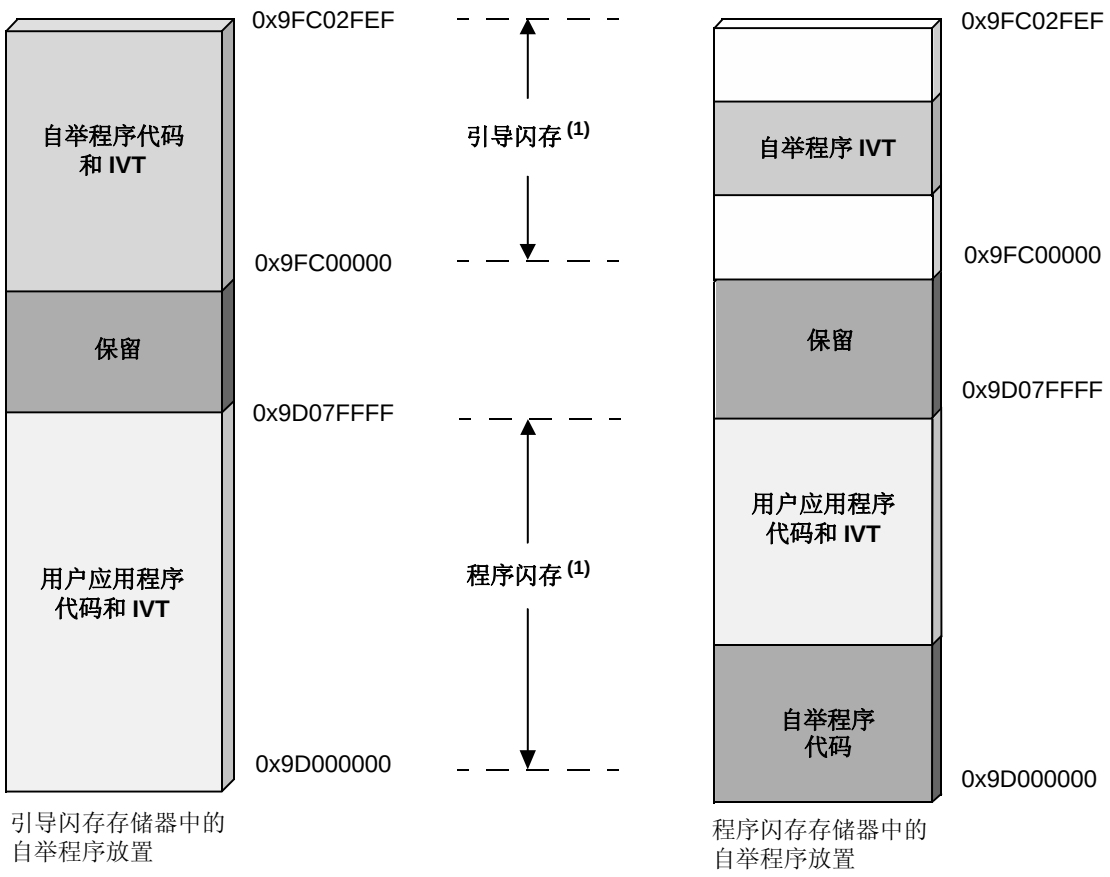
注： 在运行用户应用程序之前，自举程序应禁止和清除任何已允许的中断。自举程序的意外中断可能会干扰用户应用程序并导致应用程序失败。如果可以，应该在用户应用程序中重新初始化中断和外设。

存储器中的自举程序放置

图1给出了两种基于自举程序大小的自举程序放置方案。较小的自举程序放置在 PIC32 引导闪存存储器中。将自举应用程序放到引导闪存存储器中，可为用户应用程序提供完整的程序闪存存储器。

如果自举程序超出 PIC32 引导闪存的大小，则自举程序分为两个部分。中断向量表（Interrupt Vector Table，IVT）和 C 启动代码放置在引导闪存中，自举程序的其余部分放置在程序闪存中。

图 1：　自举程序放置



注 1： 引导闪存存储器和程序闪存存储器的大小取决于具体器件。有关详细信息，请参见具体器件数据手册中的 **“存储器构成”**一章。

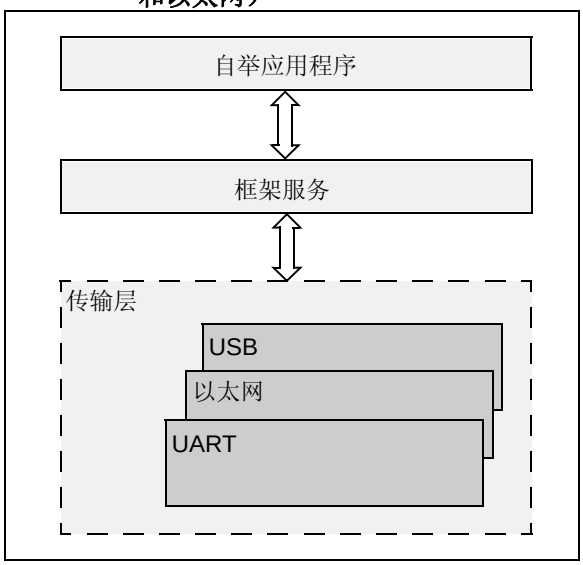
实现概述 （UART、 USB HID 和以太网）

自举应用程序使用框架实现。自举程序固件通过预定义的通信协议与 PC 主机应用程序进行通信。自举程序框架提供应用程序编程接口 （Application Programming Interface，API）函数处理来自 PC 应用程序的协议相关帧。有关通信协议的更多信息，请参见**附录 B：“自举程序通信协议 （UART、USB HID 和以太网）”**。

框架 （UART、 USB HID 和以太网）

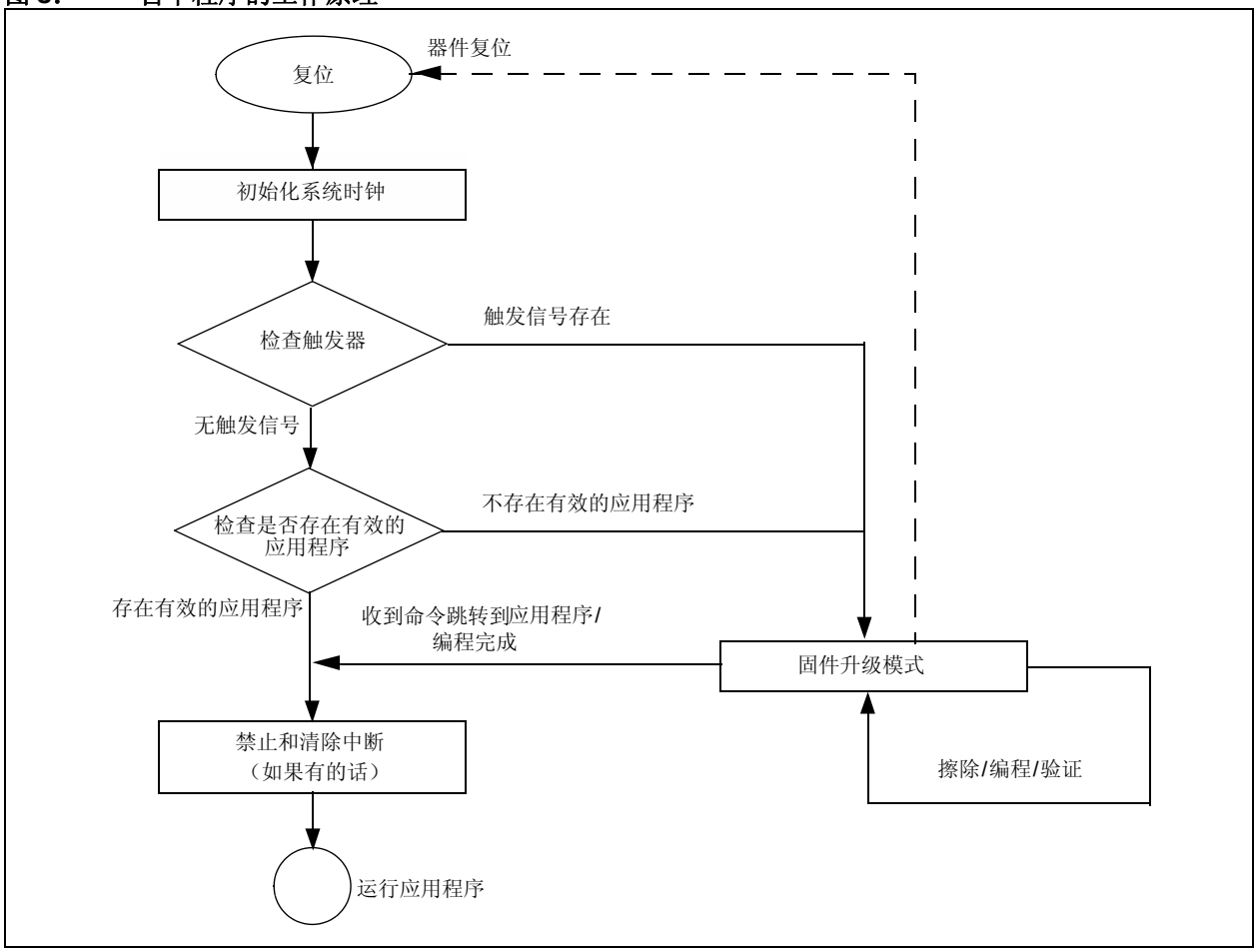
图 2 显示了自举程序架构。自举程序框架提供多个 API 函数，自举应用程序和传输层可调用这些 API 函数。用户借助自举程序框架可轻松修改自举应用程序以适合不同的要求。

图 2： 自举程序架构 （UART、 USB HID 和以太网）



`Bootloader.c`文件包含自举应用程序代码。此文件包含自举程序功能，如图 3 中所示。

图 3： 自举程序的工作原理



Framework.c 文件包含框架函数。框架处理通信协议帧，并执行接收自 PC 主机应用程序的命令。

传输层文件 Uart.c 和 Usb_HID_tasks.c 包含将原始字节发送到 PC 主机应用程序以及从 PC 主机应用程序接收原始字节的功能。

表 1 列出了框架提供的 API 函数。

表 1： 框架 API 说明 （UART、 USB HID 和以太网）

API	说明
void FrameWorkTask(void)	如果存在来自 PC 主机应用程序的有效帧，此函数将执行命令。必须定期在自举应用程序任务中调用该函数。 **输入参数：** 无。 **返回值：** 无。
void BuildRxFrame(UINT8 *RxData, INT16 RxLen)	当传输层接收到来自 PC 主机应用程序的数据时，将调用此函数。 **输入参数：** *RxData—— 指向接收到的数据缓冲区的指针 RxLen—— 接收到的数据字节的长度 **返回值：** 无。
UINT GetTransmitFrame(UINT8* TxData)	如果存在来自框架的有效响应帧，则返回一个非零值。传输层调用此函数以检查是否存在要发送到 PC 主机应用程序的帧。 **输入参数：** *TxData—— 指向将承载响应帧的数据缓冲区的指针 **返回值：** 响应帧的长度。零表示无响应帧。
BOOL ExitFirmwareUpgradeMode(void)	此函数指示自举应用程序退出固件升级模式并执行用户应用程序。当 PC 主机应用程序发出运行应用程序命令时，可能出现这种情况。 **输入参数：** 无。 **返回值：** 如果值为真，则退出固件升级模式。

处理器件配置位

对新应用程序固件进行编程时，自举程序不会擦除或写入器件配置字。这是因为器件配置字设置由自举程序和用户应用程序共享。对器件配置字的任何修改都可能使自举程序失效。因此，强烈建议用户应用程序和自举程序使用共同的器件配置字设置。

演示应用程序

`Demo_Application` 文件夹包含的应用程序示例可控制两个 LED 并使其闪烁。表 2 列出了支持 Explorer 16 开发板和 PIC32 入门工具包的两个工作区。演示应用程序使用定制链接器脚本文件将整个应用程序映射到程序闪存，而不会与自举程序重叠。

使用以下步骤配置并生成项目：

1. 在 MPLAB 或 MPLAB X 中，打开所需工作区。
2. 在 IDE 中选择所需器件的部件编号。
3. 在发布 （Release）模式下生成项目。项目编译并生成 `.hex` 文件。

生成的应用程序十六进制文件可通过自举程序下载到所选开发板中。此演示应用程序控制 Explorer 16 开发板上标记为 D9 和 D10 的两个 LED，或 PIC32 入门工具包上标记为LED1和LED2的LED，并使这些LED闪烁。

表 2： EXPLORER 16 开发板和 PIC32 入门工具包工作区

工作区	硬件资源	兼容器件	操作

对于 MPLAB®：
`Demo_App_Explorer16.mcp`
 	
对于 MPLAB X：
`Demo_App_Explorer16.X`
	Explorer 16 开发板	PIC32MX1XX(1)
		PIC32MX2XX(1)
		PIC32MX3XX
		PIC32MX4XX
		PIC32MX5XX
		PIC32MX6XX
		PIC32MX7XX
			使 LED D9 和 D10 闪烁。

对于 MPLAB：
`Demo_App_PIC32_Starter_Kits.mcp`
 	
对于 MPLAB X：
`Demo_App_PIC32_Starter_Kits.X`
	PIC32 USB 入门工具包 II 或 PIC32 以太网入门工具包	PIC32MX3XX
		PIC32MX4XX
		PIC32MX5XX
		PIC32MX6XX
		PIC32MX7XX
			使 LED LED1 和 LED2 闪烁。

注 1： 演示应用程序要求对固件修改后才能将 LED 映射到正确的 I/O 端口。

使用自举应用程序（UART、 USB HID 和以太网自举程序）

使用以下步骤运行 UART/USB HID/以太网自举程序：

1. 使用表 9 为所选自举程序选择特定硬件设置。
2. 使用调试器或编程器，通过 Release Build（发布编译）将自举程序烧写到器件中。
3. 运行 `PIC32UBL.exe`，该文件位于 `..\PC_Application\` 文件夹中。
4. 根据使用的自举程序工作区，通过 PC 应用程序中的 **Communication Settings**（通信设置）菜单使能 USB、串口或以太网。对于串口自举程序，默认波特率为 115200。对于 USB 自举程序，供应商 ID 编号（VID）和产品 ID 编号（PID）的默认值分别为 0x4D8 和 0x03C。对于以太网自举程序，默认 IP 地址为 192.168.1.11。
5. 如果使用以太网自举程序，请将 PC 的 IP 地址和子网掩码分别设置为 192.168.1.12 和 255.255.255.0。
6. 要进入固件升级模式，请按照 **“进入固件升级模式”** 中所述进行操作。
7. 根据烧写的自举程序的类型，将串行电缆、micro-USB 电缆或以太网（RJ-45）交叉电缆连接到 Explorer 16 开发板。
8. 单击 **Connect**（连接）。这将连接器件并读取自举程序版本信息。
9. 单击 **Load Hex File**（加载十六进制文件），然后浏览并进入 `Demo_Application` 文件夹。
10. 选择特定的演示应用程序文件 `Demo_App_Explorer16.hex/ Demo_App_PIC32_Starter_Kits.hex`，该文件位于 `Firmware\Demo_Application` 文件夹。
11. 单击 **Erase**（擦除）以擦除器件。
12. 单击 **Program**（编程）以将之前加载的 `.hex` 文件烧写到器件闪存。
13. 单击 **Verify**（验证）以验证闪存内容。如果正确写入闪存，控制台中将显示消息“Verification successful”（验证成功）。
14. 单击 **Run Application**（运行应用程序）。应用程序必须运行并执行操作，如表 2 中的 **Remarks**（备注）列所示。还可通过单击 **Erase-Program-Verify**（擦除 - 编程 - 验证）将第 11 步、第 12 步和第 13 步合并为单个操作执行。
15. 器件开始运行烧写的应用程序固件后， PC 应用程序将无法与器件通信。要重新连接到自举程序，请返回到第 6 步。

运行 USB 主机闪存驱动器自举程序

使用以下步骤运行 USB 主机闪存驱动器自举程序：

1. 使用表 9 为 USB 主机闪存驱动器自举程序选择硬件设置。
2. 使用调试器或编程器将 USB 主机 MSD 自举程序烧写到器件中。
3. 烧写器件后，按 **“进入固件升级模式”** 中所述执行操作，以将自举程序置于固件升级模式。
4. 将应用程序 `.hex` 文件复制到 USB 闪存驱动器，然后将该十六进制文件重命名为 `image.hex`。
5. 将 USB 闪存驱动器插入所选的开发硬件。自举程序开始将映像烧写到器件的程序闪存存储器中。如果适用，闪存驱动器上的 LED 在烧写期间将闪烁。烧写完成后，自举程序退出并开始运行应用程序。

注： 如果器件未识别闪存驱动器（或闪存驱动器上的文件），则将闪存驱动器格式化成 FAT32 格式。

运行 SD 卡自举程序

使用以下步骤运行 SD 卡自举程序：

1. 使用表 9 为 SD 卡自举程序选择特定硬件设置。
2. 将应用程序 `.hex` 文件复制到 SD 卡，然后将该十六进制文件重命名为 `image.hex`。将 SD 卡插入所选的硬件设置。
3. 通过 Release Build 将 SD 卡自举程序烧写到器件中。
4. 烧写器件后，按**“进入固件升级模式”**中所述执行操作，以将自举程序置于固件升级模式。
5. 处于固件升级模式下之后，自举程序开始将十六进制映像烧写到程序闪存中。在烧写操作期间，所选开发硬件上的指示灯LED将以较快的速率闪烁。烧写完成后，自举程序退出并开始运行用户应用程序。

注： 如果器件未识别 SD 卡上的文件，则将 SD 卡格式化成 FAT32 格式。

结论

本应用笔记介绍了 PIC32 自举程序、自举程序存储器映射、自举程序框架 API 调用的概念，以及自举程序 PC 应用程序的使用方法。

附录 B：“自举程序通信协议（UART、USB HID 和以太网）”将介绍自举程序使用的通信协议和命令。

附录 C：“移动应用程序映像时的注意事项”将介绍将应用程序映射到程序闪存内不同区域的步骤方法。

附录 D：“自举程序配置”将介绍自举程序源代码中提供的编译时间设置。

附录 E：“自举程序工作区”将列出所有可用于自举程序的工作区，并介绍将自举程序固件烧写到器件中的步骤。

参考资料

可从 Microchip Technology Inc. 获取以下资料

以下文档提供了有关链接器脚本文件中使用的术语的信息：

- 《用于 PIC32 MCU 的 MPLAB® 汇编器、链接器和实用程序用户指南》（DS51833A_CN）
- 《MPLAB® C32 C 编译器用户指南》（DS51686A_CN）

以下文档提供了有关 PIC32 器件及其外设的信息：

- 《PIC32MX1XX/2XX 系列数据手册》（DS61168D_CN）
- 《PIC32MX3XX/4XX 系列数据手册》（DS61143G_CN）
- 《PIC32MX5XX/6XX/7XX 系列数据手册》（DS61156F_CN）

附录 A： 源代码

软件许可协议

Microchip Technology Incorporated （以下简称 “本公司”）在此提供的软件旨在向本公司客户提供专门用于本公司生产的产品的软件。

本软件为本公司和 / 或其供应商所有，并受到适用的版权法保护。版权所有。使用时违反前述约束的用户可能会依法受到刑事制裁，并可能由于违背本许可的条款和条件而承担民事责任。

本软件是按 “现状”提供的。不附有任何形式的保证，无论是明示的、暗示的或法定的，包括 （但不限于）有关适销性和特定用途的暗示保证。对于在任何情况下，因任何原因造成的特殊的、偶然的或间接的损害，本公司概不负责。

本应用笔记中提及的所有软件都以单个 WinZip 归档文件的形式提供。可从 Microchip 公司网站下载此文件：

www.microchip.com

附录 B： 自举程序通信协议 （UART、USB HID 和以太网）

PC 主机应用程序使用通信协议与自举程序固件进行交互。 PC 主机应用程序作为主机，向自举程序固件发送命令以执行特定操作。

帧格式

通信协议遵循例 1 所示的帧格式。帧格式双向（即，从主机应用程序到自举程序以及从自举程序到主机应用程序）相同。

例 1： 帧格式

```
[<SOH>…]<SOH>[<DATA>…]<CRCL><CRCH><EOT>
```

其中：

<...> 表示一个字节

[...] 表示可选或可变数量的字节

帧以一个控制字符 （帧头开始 （Start of Header，SOH））开始，以另一个控制字符 （传输结束 （End of Transmission， EOT））结束。帧的完整性由两个循环冗余校验（Cyclic Redundancy Check，CRC）-16 字节保护，分别表示为 CRCL （低字节）和 CRCH （高字节）。

控制字符

数据字段中的某些字节可能与控制字符 SOH 和 EOT 相似。数据链路转义 （Data Link Escape， DLE）字符用于转义此类会被解释为控制字符的字节。自举程序总是接受 <DLE> 之后的字节作为数据，并且总是在发送任何控制字符之前先发送一个 <DLE>。

表 3： 控制字符说明

控制	十六进制值	说明
<SOH>	0x01	标识帧开始
<EOT>	0x04	标识帧结束
<DLE>	0x10	数据链路转义符

命令

PC 主机应用程序可将表 4 中列出的命令发送到自举程序。数据字段中的第一个字节用于承载命令。

表 4： 命令说明

十六进制的命令值	说明
0x01	读取自举程序版本信息
0x02	擦除闪存
0x03	编程闪存
0x04	读取 CRC
0x05	跳转到应用程序

读取自举程序版本信息

PC 主机应用程序向自举程序请求版本信息，如例 2 所示。

例 2： 请求

```
[<SOH>…]<SOH>[<0x01>]<CRCL><CRCH><EOT>
```

自举程序用两个字节响应 PC 对版本信息的请求，如例 3 所示。

例 3： 响应

```
[<SOH>…]<SOH><0x01><MAJOR_VER><MINOR_VER>
<CRCL><CRCH><EOT>
```

其中：

MAJOR_VER—— 自举程序的主要版本

MINOR_VER—— 自举程序的次要版本

擦除闪存

接收到来自 PC 主机应用程序的擦除闪存命令后，自举程序将擦除分配给用户应用程序的那部分程序闪存。PC 主机应用程序发送到自举程序的请求帧如例 4 所示。

例 4： 请求

```
[<SOH>…]<SOH><0x02><CRCL><CRCH><EOT>
```

自举程序发送到 PC 主机应用程序的响应帧如例 5 所示。

例 5： 响应

```
[<SOH>…]<SOH><0x02><CRCL><CRCH><EOT>
```

编程闪存

PC 主机应用程序发送一个或多个 Intel Hex 格式的十六进制记录以及编程闪存命令。 MPLAB C32 编译器将生成 Intel Hex 格式的映像。 Intel 十六进制文件中的每一行都表示一条十六进制记录。每条十六进制记录以一个冒号（:）开头并采用 ASCII 格式。PC 主机应用程序会丢弃冒号并将其余数据从 ASCII 格式转换为十六进制格式，然后将数据发送到自举程序。自举程序从十六进制记录中提取目标地址和数据，并将数据写入程序闪存。

PC 主机应用程序发送到自举程序的请求帧如例 6 所示。

例 6： 请求

```
[<SOH>…]<SOH><0x03>[<HEX_RECORD>…]<CRCL>
<CRCH><EOT>
```

其中：

HEX_RECORD 是十六进制格式的 Intel Hex 记录

自举程序发送到 PC 主机应用程序的响应如例 7 所示。

例 7： 响应

```
[<SOH>…]<SOH><0x03><CRCL><CRCH><EOT>
```

读取 CRC

读取 CRC 命令用于在编程后校验程序闪存的内容。PC 主机应用程序发送到自举程序的请求帧如例 8 所示。

例 8： 请求

```
[<SOH>…]<SOH><0x04><ADRS_LB><ADRS_HB>
<ADRS_UB><ADRS_MB><NUMBYTES_LB><NUMBYTES_HB>
<NUMBYTES_UB><NUMBYTES_MB><CRCL><CRCH><EOT>
```

ADRS_LB、ADRS_HB、ADRS_UB 和 ADRS_MB（如例 8 所示）表示 CRC 计算开始处的 32 位闪存地址。

NUMBYTES_LB、NUMBYTES_HB、NUMBYTES_UB 和 NUMBYTES_MB（如例 8 所示）表示将计算其 CRC 的字节总数 （32 位格式）。

自举程序发送到 PC 主机应用程序的响应如例 9 所示。

例 9： 响应

```
[<SOH>…]<SOH><0x04><FLASH_CRCL><FLASH_CRCH>
<CRCL><CRCH><EOT>
```

跳转到应用程序

PC 主机应用程序的跳转到应用程序命令用于命令自举程序执行应用程序。 PC 主机应用程序发送到自举程序的请求帧如例 10 所示。

例 10： 请求

```
[<SOH>…]<SOH><0x05><CRCL><CRCH><EOT>
```

自举程序不会响应此命令，因为自举程序会立即退出固件升级模式并开始执行应用程序。

附录 C： 移动应用程序映像时的注意事项

本节介绍将用户应用程序置于所需程序闪存存储区中的步骤。必须确保用户应用程序的存储区不会与预留给自举程序的存储区重叠。

1. 创建一个新的文本文件，然后保存为扩展名为 `.ld` 的文件。
2. 将新建的 `*.ld` 文件添加到项目中。新建的 `*.ld` 文件即显示在项目树中。
3. 从默认链接器脚本开始比从临时脚本开始简单。使用文本编辑器将 `\pic32mx\lib\ldscripts\elf32pic32mx.x` 默认链接器脚本的内容复制到新创建的 `*.ld` 文件。`INCLUDE procdefs.ld` 指令应替换为链接器脚本的器件特定 `\pic32mx\lib\proc\device\procdefs.ld` 部分的内容。路径 `\pic32mx\lib` 位于安装 C32 编译器工具的文件夹内。
4. 编辑新建的 `*.ld` 文件，以将链接器脚本存储区 `exception_mem`、`kseg0_boot_mem`、`kseg1_boot_mem` 和 `kseg0_program_mem` 重新映射到**预留给用户应用程序的程序闪存中**。`exception_mem` 必须在程序闪存的 4K 地址边界上对齐，如例 11 所示。

   有关链接器脚本存储区的更多信息，请参见《用于 PIC32 MCU 的 MPLAB® 汇编器、链接器和实用程序用户指南》（DS51833A_CN）和《MPLAB® C32 C 编译器用户指南》（DS51686A_CN）。
5. 将 `_ebase_address` 的值设置为 `exception_mem` 的 ORIGIN 值。
6. 更改存储器地址 `_RESET_ADDR`、`_BEV_EXCPT_ADDR` 和 `_DBG_EXCPT_ADDR` 的值，以便所有这些地址均位于 `kseg1_boot_mem` 范围内。
7. 清除应用程序项目，然后再生成，以获取重映射的应用程序映像。

注： 修改后的应用程序链接器脚本文件不适合在调试或独立（在没有自举程序的情况下使用）模式下构建应用程序。

应用程序重新映射后的下一个步骤是将用户应用程序在程序闪存中的新位置和复位地址通知给自举程序。

自举程序代码为此提供了编译时间选项。宏 `APP_FLASH_BASE_ADDRESS` 和 `APP_FLASH_END_ADDRESS` 定义了预留给用户应用程序的程序闪存的起始地址和结束地址。仅当闪存的目标地址处于这些地址范围内时，自举程序才执行擦除或编程操作。因此，在应用程序重映射后，用户必须将新的起始地址和结束地址值设置为这些宏。设置地址，以便用户应用程序的 `procdefs.ld` 文件中定义的 `exception_mem`、`kseg0_boot_mem`、`kseg1_boot_mem` 和 `kseg0_program_mem` 在这些地址的范围内，如例 12 中所示。

宏 `USER_APP_RESET_ADDRESS` 指定了用户应用程序的复位地址。当必须运行用户应用程序时，自举程序会转移到此地址。该宏的值必须更改为用户应用程序项目的 `procdefs.ld` 文件中定义的 `_RESET_ADDR`。在修改这些宏后，必须重新编译自举程序项目并烧写到 PIC32 器件中。

例 11：　应用程序链接器脚本中要修改的行

```
/*****************************************************************************
 * 处理器特定的对象文件。包含 SFR 定义。
 *****************************************************************************/
INPUT("processor.o")

/*****************************************************************************
 * 用于中断向量处理
 *****************************************************************************/
PROVIDE(_vector_spacing = 0x00000001);
/* _ebase_address 的值必须与 exception_mem 的 ORIGIN 值相同 （见下文） */
_ebase_address = 0x9D000000;

/*****************************************************************************
 * 使存储器地址相同
 *****************************************************************************/
 /* 使 _RESET_ADDR 等于 kseg1_boot_mem 的 ORIGIN 值 （见下文） */
_RESET_ADDR              = (0x9D000000 + 0x1000 + 0x970);

/* 将 _BEV_EXCPT_ADDR 和 _DBG_EXCPT_ADDR 映射到 kseg1_boot_mem 中 （见下文） */
/* 将 _BEV_EXCPT_ADDR 放置在与 _RESET_ADDR 偏移 0x380 的位置 */
/* 将 _DBG_EXCPT_ADDR 放置在与 _RESET_ADDR 偏移 0x480 的位置 */

_BEV_EXCPT_ADDR          = (0x9D000000 + 0x1000 + 0x970 + 0x380);
_DBG_EXCPT_ADDR          = (0x9D000000 + 0x1000 + 0x970 + 0x480);

_DBG_CODE_ADDR           = 0xBFC02000;
_DBG_CODE_SIZE           = 0xFF0     ;
_GEN_EXCPT_ADDR          = _ebase_address + 0x180;

/*****************************************************************************
 * 存储区
 *
 * 不带属性的存储区不可用于孤立部分。
 * 只有专门指定给这些区域的部分
 * 可以分配到这些区域中。
 *****************************************************************************/
MEMORY
{
  /* IVT 映射到 exception_mem 中。 exception_mem 的 ORIGIN 值必须与 4K 地址边界对齐。保持长度的默认值 */

  exception_mem       : ORIGIN = 0x9D000000, LENGTH = 0x1000

  /* 将 kseg0_boot_mem 置于与 exception_mem 相邻的位置。保持长度的默认值 */
  kseg0_boot_mem      : ORIGIN = (0x9D000000 + 0x1000), LENGTH = 0x970

  /* C 启动代码映射到 kseg1_boot_mem 中。将 kseg1_boot_mem 置于与 kseg0_boot_mem 相邻的位置。
     保持长度的默认值 */
  kseg1_boot_mem      : ORIGIN = (0x9D000000 + 0x1000 + 0x970), LENGTH = 0x490

  /* 所有 C 文件 （文本和数据）均映射到 kseg0_program_mem 中。将 kseg0_program_mem 置于与 kseg1_boot_mem
     相邻的位置。根据需要更改 kseg0_program_mem 的长度。在本例中，对 512 KB 闪存进行如下缩减： */

  kseg0_program_mem (rx):ORIGIN = (0x9D000000 + 0x1000 + 0x970 + 0x490),
    LENGTH = (0x80000 - (0x1000 + 0x970 + 0x490))


  debug_exec_mem             : ORIGIN = 0xBFC02000, LENGTH = 0xFF0
  config3                    : ORIGIN = 0xBFC02FF0, LENGTH = 0x4
  config2                    : ORIGIN = 0xBFC02FF4, LENGTH = 0x4
  config1                    : ORIGIN = 0xBFC02FF8, LENGTH = 0x4
  config0                    : ORIGIN = 0xBFC02FFC, LENGTH = 0x4
  kseg1_data_mem       (w!x) : ORIGIN = 0xA0000000, LENGTH = 0x20000
  sfrs                       : ORIGIN = 0xBF800000, LENGTH = 0x100000
  configsfrs                 : ORIGIN = 0xBFC02FF0, LENGTH = 0x10
}
```

例 12：　自举程序代码中要修改的行

```
/* APP_FLASH_BASE_ADDRESS 和 APP_FLASH_END_ADDRESS 为预留给应用程序的程序闪存 */
/* 规则：
        1) 应用程序链接器脚本的存储区 kseg0_program_mem、 kseg0_boot_mem、 exception_mem 和
        kseg1_boot_mem 必须在 APP_FLASH_BASE_ADDRESS 和 APP_FLASH_END_ADDRESS 的范围内

        2) 基址和结束地址必须在 4K 地址边界上对齐
*/

#define APP_FLASH_BASE_ADDRESS    0x9D000000
#define APP_FLASH_END_ADDRESS     0x9D07FFFF

/* 应用程序开始执行处的闪存地址 */
/* 规则：将 APP_FLASH_BASE_ADDRESS 设置为应用程序链接器脚本的 _RESET_ADDR 值 */

#define USER_APP_RESET_ADDRESS (0x9D000000 + 0x1000 + 0x970)
```

附录 D： 自举程序配置

自举程序代码提供了几个宏，用于在编译时进行配置设置。表 5 到表 8 列出了这些宏及其用途。根据用户要求，可能需要更改这些宏中的值。

表 5： 通用宏

宏	用途
APP_FLASH_BASE_ADDRESS	预留给用户应用程序的程序闪存的基址。地址值必须指向 4K 闪存页的开始位置。
APP_FLASH_END_ADDRESS	预留给用户应用程序的程序闪存的结束地址。地址值必须指向 4K 闪存页的结束位置。
USER_APP_RESET_ADDRESS	用户复位向量的地址。当必须运行用户应用程序时，自举程序会转移到此地址。
MAJOR_VERSION	自举程序固件的主要版本。
MINOR_VERSION	自举程序固件的次要版本。

表 6： UART 自举程序宏

宏	用途
DEFAULT_BAUDRATE	设置 UART 波特率。

表 7： USB HID 自举程序宏

宏	用途
USB_VENDOR_ID	设置供应商 ID。
USB_PRODUCT_ID	设置产品 ID。

表 8： 以太网自举程序宏

宏	用途
MY_DEFAULT_MAC_BYTE1 MY_DEFAULT_MAC_BYTE2 MY_DEFAULT_MAC_BYTE3 MY_DEFAULT_MAC_BYTE4	设置 MAC 地址。
MY_DEFAULT_IP_ADDR_BYTE1 MY_DEFAULT_IP_ADDR_BYTE2 MY_DEFAULT_IP_ADDR_BYTE3 MY_DEFAULT_IP_ADDR_BYTE4	设置 IP 地址。

附录 E： 自举程序工作区

Bootloader 文件夹包含自举程序固件源代码。表 9 列出了可用的工作区。

烧写自举程序固件

根据硬件配置和所选自举程序类型，使用以下步骤选择合适的工作区。

1. 使用 MPLAB 或 MPLAB X 打开特定自举程序工作区。
2. 在 IDE 中选择特定器件的部件编号，然后重新编译项目。
3. 编译项目后，将自举程序烧写到器件中。
4. 使用 REAL ICE 或 ICD 3 烧写 Explorer 16 开发板上的器件。 PIC32 入门工具包不需要编程器。

表 9： 可用的自举程序工作区

项目名称	开发硬件	兼容器件	自举程序类型	自举程序映射
对于 MPLAB®： UART_Btl_Explorer16.mcp 对于 MPLAB X： UART_Btl_Explorer16.X	Explorer 16 开发板（DM24001）和 PIC32 接插模块（PIM）	PIC32MX1XX(1,2) PIC32MX2XX(1,2) PIC32MX3XX PIC32MX4XX PIC32MX5XX PIC32MX6XX PIC32MX7XX	UART	见**注 3**
对于 MPLAB： UART_HID_Btl_StarterKit.mcp 对于 MPLAB X： UART_HID_Btl_StarterKit.X	PIC32 以太网入门工具包 （DM320004）*或* PIC32 USB 入门工具包 II （DM320003-2）	PIC32MX2XX(1,2) PIC32MX3XX PIC32MX4XX PIC32MX5XX PIC32MX6XX PIC32MX7XX	USB HID	见**注 3**
对于 MPLAB： UART_MSD_Btl_StarterKit.mcp 对于 MPLAB X： UART_MSD_Btl_StarterKit.X	PIC32 以太网入门工具包 （DM320004）*或* PIC32 USB 入门工具包 II （DM320003-2）	PIC32MX2XX(1,2) PIC32MX3XX PIC32MX4XX PIC32MX5XX PIC32MX6XX PIC32MX7XX	USB MSD	见**注 3**
对于 MPLAB： ETH_Btl_ETH_StarterKit.mcp 对于 MPLAB X： ETH_Btl_ETH_StarterKit.X	PIC32 以太网入门工具包 （DM320004）	PIC32MX6XX PIC32MX7XX	以太网 （具有内部 MAC）	见**注 3**

注 1： 需要根据这些器件的 I/O 端口映射修改自举程序代码。

2： Bootloader\linker_scripts\PIC32MX_1XX_2XX 文件夹中提供了这些器件的自举程序链接器脚本示例。用户必须使用特定的链接器脚本文件编译自举程序。

3： 请参见相应链接器脚本文件中的 “映射说明”。

表 9： 可用的自举程序工作区（续）

项目名称	开发硬件	兼容器件	自举程序类型	自举程序映射
对于 MPLAB： ETH_Btl_ETH_Explorer16_ENC28.mcp 对于 MPLAB X： ETH_Btl_ETH_Explorer16_ENC28.X	Explorer 16 开发板（DM24001）、PIC32 PIM 和以太网 PICtail™ Plus 子板（AC164123）	PIC32MX1XX(1,2) PIC32MX2XX(1,2) PIC32MX3XX PIC32MX4XX PIC32MX5XX PIC32MX6XX PIC32MX7XX	以太网（具有外部以太网控制器 ENC28J60）	见**注 3**
对于 MPLAB： ETH_Btl_Explorer16_ENC624.mcp 对于 MPLAB X： ETH_Btl_Explorer16_ENC624.X	Explorer 16 开发板（DM24001）、PIC32 PIM 和快速 100 Mbps 以太网 PICtail™ Plus 子板（AC164132）	PIC32MX1XX(1,2) PIC32MX2XX(1,2) PIC32MX3XX PIC32MX4XX PIC32MX5XX PIC32MX6XX PIC32MX7XX	以太网（具有外部以太网控制器 ENC624J600）	见**注 3**
对于 MPLAB： SD_Card_Btl_Explorer16.mcp 对于 MPLAB X： SD_Card_Btl_Explorer16.X	Explorer 16 开发板（DM24001）、PIC32 PIM 以及用于 SD™ 和 MMC 卡的 PICtail™ 子板（AC164122）	PIC32MX1XX(1,2) PIC32MX2XX(1,2) PIC32MX3XX PIC32MX4XX PIC32MX5XX PIC32MX6XX PIC32MX7XX	SD 卡	见**注 3**

注 1： 需要根据这些器件的 I/O 端口映射修改自举程序代码。

2： Bootloader\linker_scripts\PIC32MX_1XX_2XX 文件夹中提供了这些器件的自举程序链接器脚本示例。用户必须使用特定的链接器脚本文件编译自举程序。

3： 请参见相应链接器脚本文件中的 “映射说明”。

附录 F： 版本历史

版本 A （2011 年 6 月）

本文档的初始版本。

版本 B （2012 年 1 月）

此版本包括以下更新：

- 更新了 **“简介”** 中的自举应用程序列表
- 更新了 **“先决条件”**
- 更新了 **“自举程序的基本流程”** 中的所有内容
- 删除了 **“基本设置”** 一节
- 更新了图 1
- 更新了 **“实现概述（UART、USB HID 和以太网）”** 和 **“框架（UART、USB HID 和以太网）”** 中的标题
- 更新了图 2 中的标题
- 更新了图 3
- 更新了表 1 中的标题
- 增加了 **“演示应用程序”**
- 更新了 **“处理器件配置位”** 中的所有内容
- 更新了 **“使用自举应用程序 （UART、 USB HID 和以太网自举程序）”** 中的标题和步骤
- 删除了图 5：通信设置
- 删除了图 6：自举程序成功连接
- 删除了图 7：PC 应用程序消息
- 增加了 **“运行 USB 主机闪存驱动器自举程序”**
- 增加了 **“运行 SD 卡自举程序”**
- 更新了 **“结论”**
- 更新了 **附录 B：“自举程序通信协议（UART、USB HID 和以太网）”** 中的标题
- 更新了 **附录 C：“移动应用程序映像时的注意事项”** 中的所有内容
- 增加了 **附录 D：“自举程序配置”**
- 增加了 **附录 E：“自举程序工作区”**
- 对整篇文档的文本和格式进行了少量更改

注：

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 “牢不可破”的。

代码保护功能处于持续发展中。 Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

ISBN：978-1-62076-255-4

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2009 认证。 Microchip 的 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品严格遵守公司的质量体系流程。此外， Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

全球销售及服务网点

美洲

公司总部 Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://www.microchip.com/support
网址：www.microchip.com

亚特兰大 Atlanta
Duluth, GA
Tel: 1-678-957-9614
Fax:1-678-957-1455

波士顿 Boston
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 Chicago
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

克里夫兰 Cleveland
Independence, OH
Tel: 1-216-447-0464
Fax: 1-216-447-0643

达拉斯 Dallas
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 Detroit
Farmington Hills, MI
Tel: 1-248-538-2250
Fax: 1-248-538-2260

印第安纳波利斯 Indianapolis
Noblesville, IN
Tel: 1-317-773-8323
Fax: 1-317-773-5453

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

圣克拉拉 Santa Clara
Santa Clara, CA
Tel: 1-408-961-6444
Fax: 1-408-961-6445

加拿大多伦多 Toronto
Mississauga, Ontario, Canada
Tel: 1-905-673-0699
Fax: 1-905-673-6509

亚太地区

亚太总部 Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 – 北京
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

中国 – 成都
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

中国 – 重庆
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

中国 – 杭州
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

中国 – 香港特别行政区
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 – 南京
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

中国 – 青岛
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 – 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 – 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 – 深圳
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

中国 – 武汉
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

中国 – 西安
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

中国 – 厦门
Tel: 86-592-238-8138
Fax: 86-592-238-8130

中国 – 珠海
Tel: 86-756-321-0040
Fax: 86-756-321-0049

亚太地区

台湾地区 – 高雄
Tel: 886-7-536-4818
Fax: 886-7-330-9305

台湾地区 – 台北
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

台湾地区 – 新竹
Tel: 886-3-5778-366
Fax: 886-3-5770-955

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

印度 India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

印度 India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

印度 India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

日本 Japan - Osaka
Tel: 81-66-152-7160
Fax: 81-66-152-9310

日本 Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

韩国 Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

韩国 Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

马来西亚 Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

马来西亚 Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

菲律宾 Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

新加坡 Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

泰国 Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

欧洲

奥地利 Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

丹麦 Denmark-Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

法国 France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

德国 Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

意大利 Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

荷兰 Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

西班牙 Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

英国 UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

11/29/11