Document Number: AN4368
Rev 1, 12/2012

USB 大容量存储设备主机引导加载程序

作者: Derek Lau

内容

1 简介

引导加载程序是放在设备中的一个小程序，可用于在设备中进行用户应用代码编程。我们已针对飞思卡尔 32 位 ColdFire 和 Kinetis MCU 系列生成了使用 USB 大容量存储设备(MSD)的 USB 主机引导加载程序。

将 USB 记忆棒插入系统后，就可以将用户的应用程序代码烧录到 MCU 中。本应用笔记将使用 MCF51JM128、MCF52259 和 MK60N512VMD100 MCU 来演示引导加载程序在 ColdFire 和 Kinetis MCU 中的工作方式。

2 引导加载程序概述

引导加载程序是一个小型应用，用于擦除闪存以及将用户应用载入设备。USB 主机 MSD 引导加载程序可让您轻松、可靠地将用户应用载入设备。

将包含有效 s-record 或二进制文件的 USB 记忆棒插入系统后，引导加载程序将会加载用户应用代码，并将其烧录到设备中。然后，新的用户应用便可以在设备中运行。本应用笔记将会帮助读者深入了解引导加载程序，以及培养使用引导加载程序自行开发应用的能力。

下图显示了 MCF51JM128、MCF52259 和 MK60N512VMD100 引导加载程序系统的存储器映射。

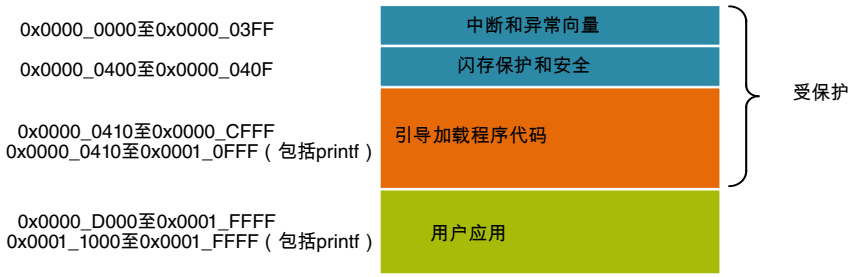

图 1. MCF51JM128 引导加载程序存储器映射

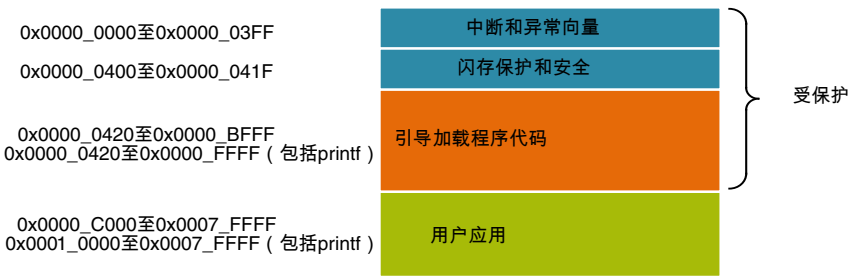

图 2. MCF52259 引导加载程序存储器映射

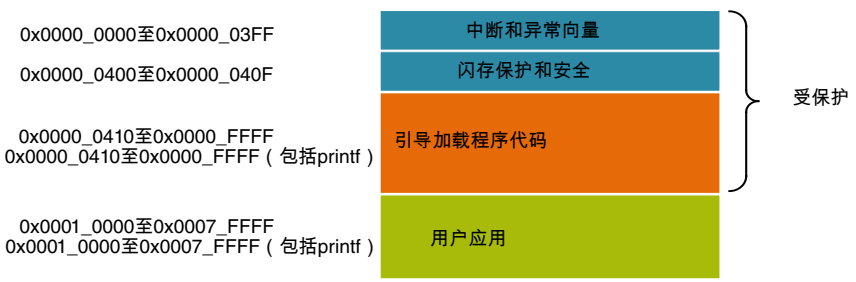

图 3. MK60N512VMD100 引导加载程序存储器映射

默认的中断和异常向量放置在闪存区域的起始地址中，此地址由引导加载程序使用，您不得更改此地址。必须将用户应用中断和异常向量放置在应用闪存区域中，并在应用启动例程中将其复制到 RAM 存储器。中断和异常向量可重定向到 RAM 区域。

引导加载程序可以擦除应用闪存，分析用户应用映像，以及在用户应用区域的闪存（将引导加载程序放入闪存后可用的闪存）中为该映像编程。使用 printf 函数显示调试消息时，引导加载程序的代码会增多。引导加载程序闪存区域必须受到保护，而可用闪存必须进行块对齐，因此，可用闪存可能会变小。

对于 MCF52259 而言，不支持 printf 的引导加载程序将占用闪存区域 0x0000 至 0x9FFF (40 KB)。由于闪存保护块大小为 16 KB，因此需要保护闪存区 0x0000 到 0xBFFF (48 KB)，以防止引导加载程序损坏。保护后，引导加载程序将占用 48 KB。从 0xC000 到 0x7FFFF (464 KB)的余下闪存供用户应用使用。

无论引导加载程序使用多少 RAM，用户应用都可以使用整个 RAM 存储器。

3 引导加载程序架构

引导加载程序包含引导加载程序应用、文件分配表(FAT)文件系统支持模块、引导加载程序驱动程序、闪存驱动程序、USB MSD 主机类、USB 主机协议栈和 USB 主机控制器。

下图显示了引导加载程序系统的架构:

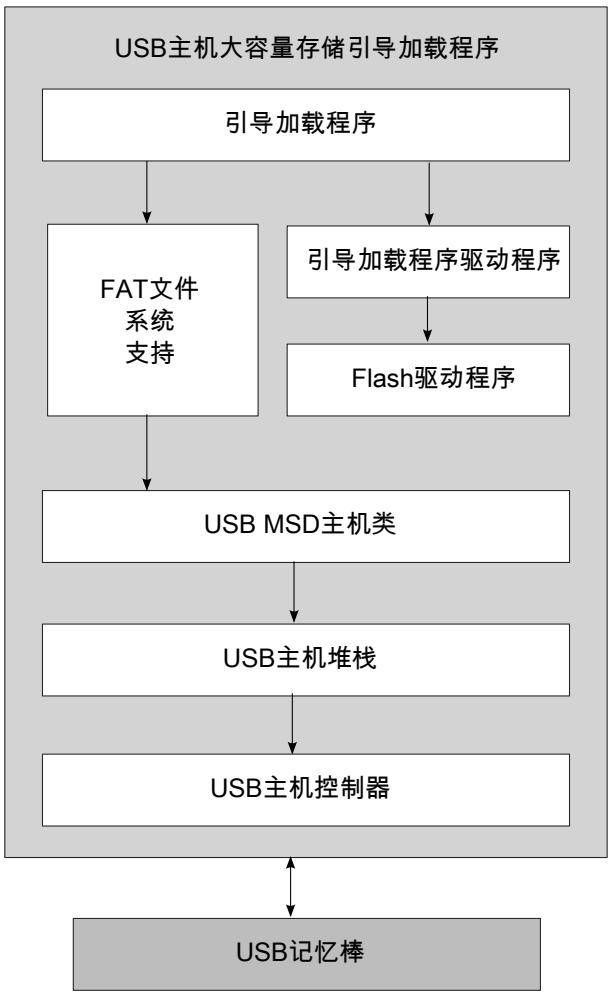

图 4. USB 主机大容量存储引导加载程序架构

- 引导加载程序控制加载过程。它使用 FAT 文件系统支持模块读取映像文件，然后使用引导加载程序驱动程序将该映像文件烧录到设备闪存中。
- 借助 FAT 文件系统支持模块，引导加载程序应用可以从 FAT32 格式的 USB 记忆棒中读取文件。
- 引导加载程序驱动程序将会分析映像文件，并将该文件烧录到闪存中。它支持分析 CodeWarrior 二进制、S19 和原始二进制文件格式的映像文件。
- 闪存驱动程序支持擦除、读取和写入闪存。
- USB MSD 主机类为 USB MSD 类中指定的应用程序接口提供服务。
- USB 主机协议栈和 USB 主机控制器通过 USB MSD 协议来与 USB 记忆棒通信。
- USB 记忆棒用于存储需要在闪存中编程的映像文件。映像文件必须采用 CodeWarrior 二进制、S19 或原始二进制文件格式。

3.1 引导加载程序软件流程

引导加载程序系统中集成了一个用于用户程序升级的引导加载程序，以及一个用于执行产品主要功能的用户应用。在复位和初始化后，该系统将会确定启动用户应用程序或引导加载程序模式。如果未提供有效的用户应用程序，设备将自动以引导加载程序模式启动。如果提供了有效的应用，则按下特定的键时，设备将运行引导加载程序，否则将运行用户应用。下表显示了适用于不同开发板的引导加载程序示例，以及示例中使用的指定键。

表 1. 用于进入引导加载程序模式的指定键

开发板	指定键
M51JM128EVB	PTG1
M52259EVB	PDD5 (SW1)
TWR-K60-N512-KIT	PTA19 (SW1)

系统进入引导加载程序模式后，将不断地检查是否已连接 USB 记忆棒。如果连接了 USB 记忆棒，它将会搜索 image.s19 和 image.bin 文件。如果存在有效的 S19 或二进制文件，它将开始分析该文件，并将该文件编程到应用区域中。以下是引导加载程序的流程图：

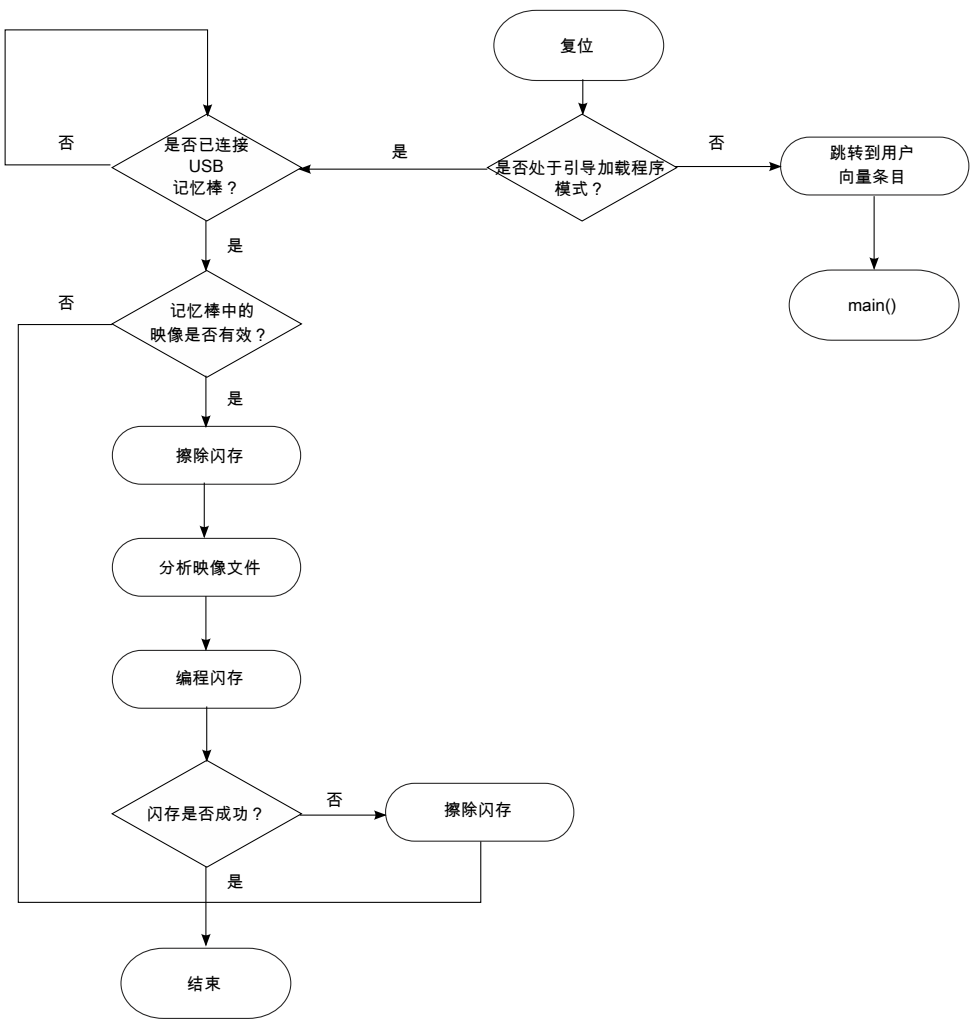

图 5. 引导加载程序软件流程

4 开发新的引导加载程序

本部分概述引导加载程序的文件结构，并说明如何在其他平台中开发新的引导加载程序。

4.1 引导加载程序文件结构

下图显示了给定源代码的文件格式：

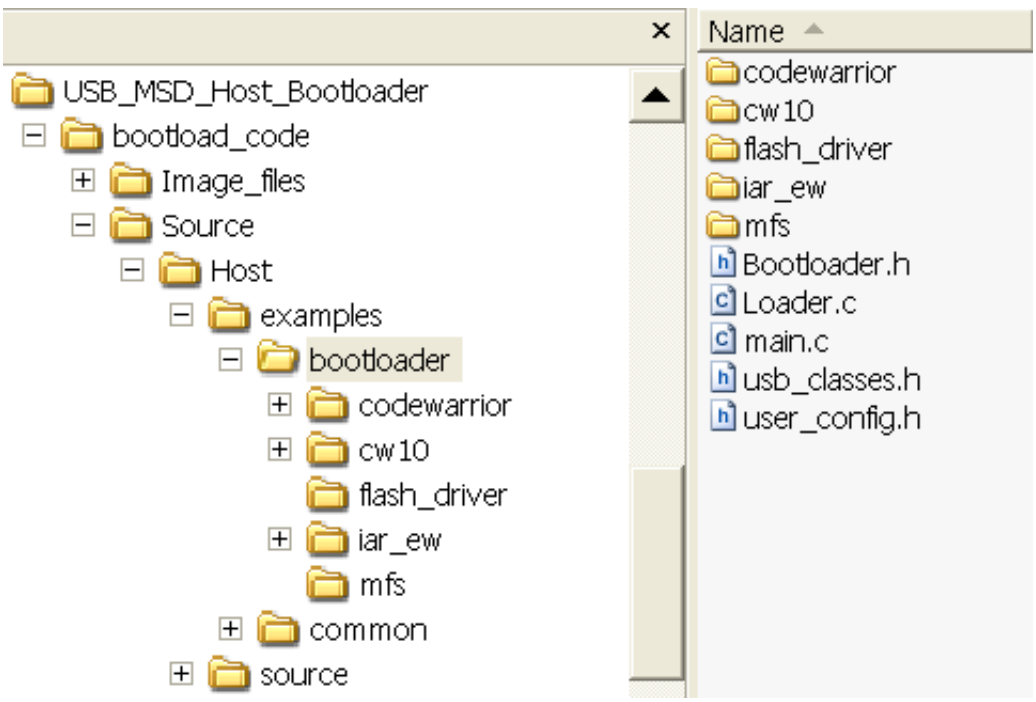

图 6. 引导加载程序文件结构

文件夹 *USB_MSD_Host_Bootloader\bootload_code\Source\Host\examples\bootloader* 包含以下子文件夹：

- ***CodeWarrior***：包含 CodeWarrior 版本 6.3 中 MCF51JM128 的项目，以及 CodeWarrior 版本 7.2 中 MCF52259 的项目。
- ***cw10***：包含 CodeWarrior 10.1 中 MCF51JM128、MCF52259 和 MK60N512VMD100 的项目。
- ***flash_driver***：包含闪存驱动程序。
- ***iar_ew***：包含 IAR 中 MK60N512VMD100 的项目。
- ***mfs***：包含文件系统源代码：
  - ***bootloader.h***：包含设备存储器映射和引导加载程序例程的定义。
  - ***user_config.h***：包含用户配置的定义。
  - ***usb_classes.h***：定义 USBCLASS_INC_MASS 和 USBCLASS_INC_HUB 的标识符，用于指示引导加载程序使用 MSD 类和 HUB 类。
  - ***load.c***：包含用于分析映像文件以及将该文件编程到设备闪存中的源代码。
  - ***main.c***：包含用于处理 USB 事件，以及检查系统是否已进入引导加载程序模式的源代码。

.

4.2 将引导加载程序移植到其他平台

本部分介绍有关将引导加载程序移植到其他平台的步骤，这些平台符合以下假设条件：

- 平台支持 USB MSD 类。
- 平台支持 FAT32 文件系统。
- 提供了闪存驱动程序。

用户可以参考以下步骤将引导加载程序移植到其他平台：

1. 在 *USB_MS_Host_Bootloader\bootload_code\Source\Host\examples\bootloader\CodeWarrior* 或 *USB_MS_Host_Bootloader\bootload_code\Source\Host\examples\bootloader\cw10 directory* 中创建新项目。

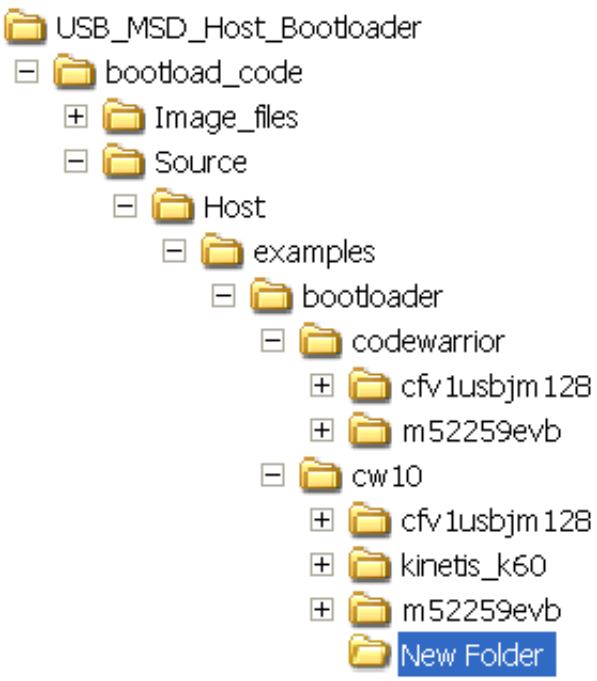

图 7. 创建新的项目文件夹

2. 创建一个其文件结构与 cfv1usbjm128、m52259evb 或 kinetis_k60 的项目类似的项目。

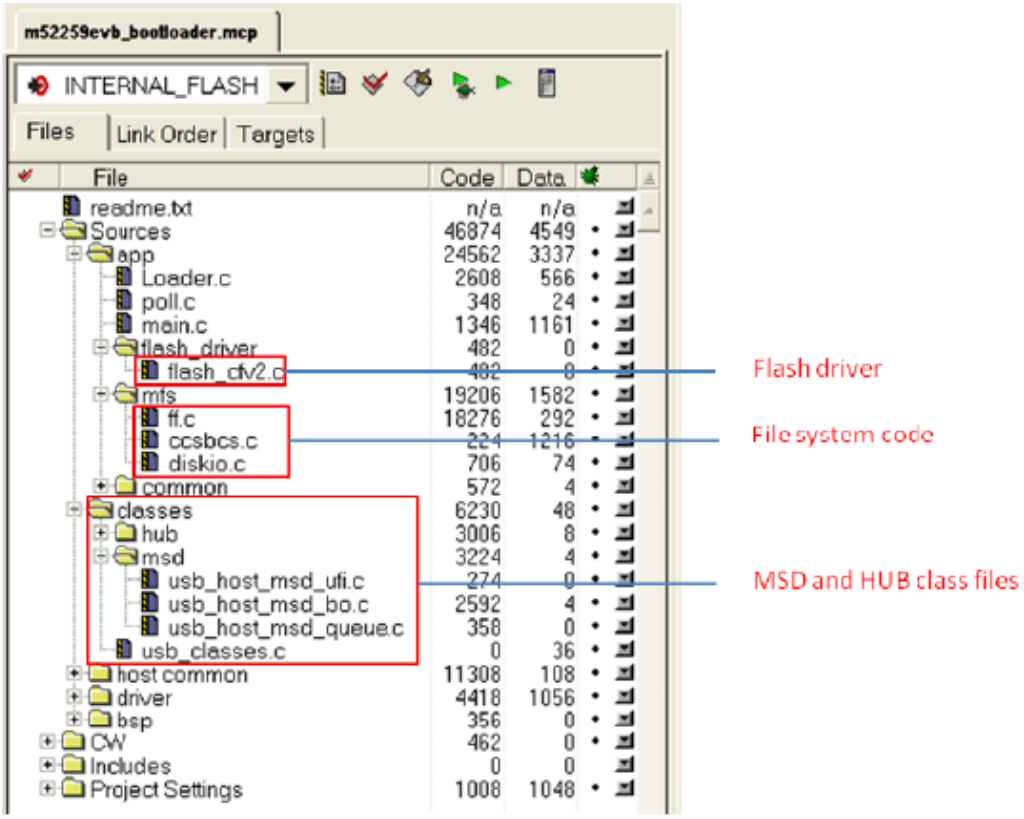

图 8. M52259 引导加载程序项目

3. 将以下文件添加到该项目：
   - USB MSD 类源代码
   - USB HUB 类源代码(可选)
   - 文件系统源代码
   - 闪存驱动程序源代码
   - *Bootloader.h*、*Loader.c, main.c*、*usb_classes.h*、*user_config.h*
4. 在文件 Bootloader.h 中添加新的存储器映射，用于指示平台的应用区。以下代码显示了 MCF52259、MCF51JM128 和 MK60N512VMD100 的存储器映射。

```
/* Define for MM52259 */
#if (defined __MCF52259_H__)
#define MIN_RAM1_ADDRESS 0x20000000
#define MAX_RAM1_ADDRESS 0x2000FFFF
#define MIN_FLASH1_ADDRESS 0x00000000
#define MAX_FLASH1_ADDRESS 0x0007FFFF
#if (defined __DEBUG__)
#define IMAGE_ADDR ((uint_32_ptr)0x10000)
#else
#define IMAGE_ADDR ((uint_32_ptr)0xC000)
#endif
#define ERASE_SECTOR_SIZE (0x1000) /* 4K bytes*/
/* Define for JM128 */
#elif (defined _MCF51JM128_H)
#define MIN_RAM1_ADDRESS 0x00800000
#define MAX_RAM1_ADDRESS 0x00803FFF
#define MIN_FLASH1_ADDRESS 0x00000000
#define MAX_FLASH1_ADDRESS 0x0001FFFF
#if (defined __DEBUG__)
#define IMAGE_ADDR ((uint_32_ptr)0x11000)
#else
#define IMAGE_ADDR ((uint_32_ptr)0xD000)
#endif
#define ERASE_SECTOR_SIZE (0x0400) /* 4K bytes*/
```

```
/* Define for K60 */
#elif (defined MCU_MK60N512VMD100)
#define MIN_RAM1_ADDRESS 0x1FFF0000
#define MAX_RAM1_ADDRESS 0x20010000
#define MIN_FLASH1_ADDRESS 0x00000000
#define MAX_FLASH1_ADDRESS 0x0007FFFF
#define IMAGE_ADDR ((uint_32_ptr)0x10000)
#define ERASE_SECTOR_SIZE (0x800) /* 2K bytes*/
#endif
```

5 开发新的用户应用

本部分介绍如何修改普通应用来使用引导加载程序。

5.1 修改链接器文件

对于使用 ColdFire 和 Kinetis MCU 的普通应用，中断向量表位于闪存区域的起始位置，应用代码可以放置在剩余的任何闪存区域中。在引导加载程序系统中，中断向量表和引导加载程序程序放置在闪存的起始位置，用户应用放置在剩余的闪存区域中。必须修改链接器文件以指示链接器将应用中断向量表和应用代码放入某些指定的内存区中。

修改 **CFV1** 链接器文件

下面显示了 MCF51JM128 的普通应用链接器文件 project.lcf 的代码，它通知链接器可以将代码放置在闪存区域 0x410 到 0x1FFFF 中。

```
MEMORY {
 code (RX) : ORIGIN = 0x00000410, LENGTH = 0x0001FBF0
 userram (RWX) : ORIGIN = 0x0080000, LENGTH = 0x00004000
}
```

要使用 MCF51JM128 的引导加载程序系统，必须将用户应用定位在从地址 0xD000 (不支持 printf)或 0x11000 (支持 printf)开始的闪存区域中。可按如下所示修改支持 printf 的链接器文件:

```
MEMORY {
 code (RX) : ORIGIN = 0x00011410, LENGTH = 0x0000EBF0
 userram (RWX) : ORIGIN = 0x00800400, LENGTH = 0x00003C00
}
```

修改 **CFV2** 链接器文件

下面显示了 MCF52259 的普通应用链接器文件 MCF52259_INTERNAL_FLASH.lcf 的代码，它通知链接器可以将代码放置在闪存区域 0x420 到 0x7FFFF 中。

```
MEMORY {
 vectorrom (RX) : ORIGIN = 0x00000000, LENGTH = 0x00000400
 cfmprotrom (RX) : ORIGIN = 0x00000400, LENGTH = 0x00000020
 code (RX) : ORIGIN = 0x00000420, LENGTH = 0x0007FB00
}
```

要使用 MCF52259 的引导加载程序系统，必须将用户应用定位在从地址 0xC000 (不支持 printf)或 0x10000 (支持 printf)开始的闪存区域中。可按如下所示修改支持 printf 的链接器文件:

```
MEMORY {
 vectorrom (RX) : ORIGIN = 0x00010000, LENGTH = 0x00000400
 cfmprotrom (RX) : ORIGIN = 0x00010400, LENGTH = 0x00000020
 code (RX) : ORIGIN = 0x00010420, LENGTH = 0x0006FBE0
 vectorram (RWX) : ORIGIN = 0x20000000, LENGTH = 0x00000400
}
```

修改 **Kinetis** 链接器文件

下面显示了 MK60N512VMD100 的普通应用链接器文件 MK60N512VMD100_flash.lcf 的代码，它通知链接器可以将代码放置在闪存区域 0x410 到 0x7FFFF 中。

```
MEMORY
{
 interrupts (RX) : ORIGIN = 0x00000000, LENGTH = 0x00000400
 cfmprotrom (RX) : ORIGIN = 0x00000400, LENGTH = 0x00000010
 code (RX) : ORIGIN = 0x00000410, LENGTH = 0x0007FBF0
}
```

要使用 MK60N512VMD100 的引导加载程序系统，必须将用户应用定位在从地址 0x10000（支持或不支持 printf）开始的闪存区域中。可按如下所示修改链接器文件:

```
MEMORY
{
 interrupts (RX) : ORIGIN = 0x00010000, LENGTH = 0x00000400
 cfmprotrom (RX) : ORIGIN = 0x00010400, LENGTH = 0x0000010
 code (RX) : ORIGIN = 0x00010410, LENGTH = 0x0006FBF0
}
```

5.2 重定向中断和异常向量

默认的中断和异常向量放置在闪存区域的起始地址中，此地址由引导加载程序使用，您不得更改此地址。 如果用户应用使用中断，则必须将用户应用中断和异常向量放入 RAM。重定向中断向量以及使用 RAM 中的中断向量的方式可能根据不同的 MCU 而有所不同。本部分介绍如何使用飞思卡尔 MQX™ USB 协议栈和其他 USB 协议栈来完成向量重定向。

MQX USB 协议栈

飞思卡尔 MQX USB 协议栈使用配置文件 userconfig.h 中的标识符 MQX_ROM_VECTORS，将应用配置为在 RAM 或 ROM (闪存)中放置中断向量。用户可以将 MQX_ROM_VECTORS 定义为 0，以将中断向量放入 RAM。

注

如果配置文件发生了更改，请记得重新编译库。

```
/* userconfig.h */
#define MQX_ROM_VECTORS 0 //DES 1=ROM, 0=RAM vector table...used with bootloaders
```

其他 **USB** 协议栈

重定向向量表的方式可能根据不同的平台而有所不同。以下部分介绍如何将向量表重定向到 CFV1、CFV2 和 Kinetis MCU 系列的 RAM。

CFV1 ColdFire

由于从地址 0x0000 开始的向量表由引导加载程序使用，因此，必须将重定向的向量表放入 RAM。CFV1 ColdFire 微控制器(例如 MCF51JM128)的向量基寄存器(VBR)包含异常向量表的 1MB 对齐基地址，可以使用该地址将向量表从它在闪存地址 0x0000 中的默认位置，重新定位到 RAM 的基地址，例如 0x0080_0000。

以下代码用于将 MCU 配置为使用位于 RAM 地址 0x0080_0000 处的向量表。

```
/* startcf.c */
      asm (move.l #0x00800000,d0);
      asm (movec d0,vbr);
```

以下代码是用于将中断例程地址放入原始闪存向量表的链接器的原始代码。

```
/* exceptions.c */
__declspec(weak) vectorTableEntryType vector_0 @INITSP = (vectorTableEntryType)&_SP_INIT;
__declspec(weak) vectorTableEntryType vector_1 @INITPC = (vectorTableEntryType)&_startup;
__declspec(weak) vectorTableEntryType vector_2 @Vaccerr = asm_exception_handler;
```

应用向量表将被放入应用闪存区域，然后复制到 RAM。以下代码可以声明引导加载程序框架中的新向量表:

```
/* exceptions.c */
#define APP_FLASH 0x00011000;
__declspec(weak) vectorTableEntryType vector_0 @(APP_FLASH+INITSP) =
(vectorTableEntryType)&_SP_INIT;
__declspec(weak) vectorTableEntryType vector_1 @(APP_FLASH+INITPC) =
(vectorTableEntryType)&_startup;
__declspec(weak) vectorTableEntryType vector_2 @(APP_FLASH+Vaccerr) = asm_exception_handler;}
```

以下代码可将上述向量表从应用闪存区域复制到 RAM 中的向量表区。

```
/* main.c */
    #define APP_FLASH 0x00011000;
     dword *pdst;
     dword *psrc;
     pdst=(dword*)0x00800000;
     psrc=(dword*)APP_FLASH;
     for (i=0;i<111;i++){
           *pdst++=*psrc++;}
```

CFV2 微控制器

对于 CFV2 版本(例如 MCF52259), 装有 PHDC 版本 3.0 的飞思卡尔 USB 协议栈支持使用一个函数将中断向量表复制到 RAM 中的指定区域。使用其他协议栈的用户可将此视作参考。

```
void initialize_exceptions(void)
{
   uint32 n;
   /* Copy the vector table to RAM */
   if (__VECTOR_RAM != (unsigned long*)_vect)
   {
         for (n = 0; n < 256; n++)
                __VECTOR_RAM[n] = (unsigned long)_vect[n];
   }
   mcf5xxx_wr_vbr((unsigned long)__VECTOR_RAM);
}
```

上面所示的 initialize_exceptions 函数已将中断向量表复制到 RAM 区域中的*__VECTOR_RAM* 地址。需要在链接器文件(*.lcf)中定义此地址(ADDR)。

```
MEMORY {
vectorram (RWX) : ORIGIN = 0x20000000, LENGTH = 0x00000400
}
___VECTOR_RAM = ADDR(.vectorram);
```

启动时，将按默认调用上面所示的函数，因此，如果使用装有 PHDC 版本 3.0 的 USB 协议栈，则用户程序不需要调用此函数。

Kinetis 微控制器

在 Kinetis MCU 中，SCB_VTOR 寄存器包含异常向量表的基地址。要重定向向量表，请将该向量表复制到 RAM，然后将 SCB_VTOR 设置为 RAM 地址。发行的大多数示例代码都已实现向量重定向。

以下步骤说明了重定向 Kinetis MCU 向量表的方法:

1. 在链接器文件中声明一个用于存储向量表的 ROM 区域，以及向量表要复制到的 RAM 区域。

```
KEEP_SECTION { .vectortable }
MEMORY {
  interrupts (RX) : ORIGIN = 0x00010000, LENGTH = 0x00000400
  vectorram (RWX) : ORIGIN = 0x1FFF0000, LENGTH = 0x00000400
  data (RW) : ORIGIN = 0x1FFF0400, LENGTH = 0x0001FC00
}


.interrupts :
 {
```

```
___VECTOR_ROM = .;
* (.vectortable)
. = ALIGN (0x4);
} > interrupts

.vectorram : {
___VECTOR_RAM = .;
} > vectorram
```

2. 执行此代码，将中断向量表复制到 RAM 并从 RAM 中运行。

```
extern uint_32 ___VECTOR_RAM[];
extern uint_32 ___VECTOR_ROM[];      //Get vector table in ROM

uint_32 i,n;
/* Copy the vector table to RAM */
if (___VECTOR_RAM != ___VECTOR_ROM)
{
for (n = 0; n < 0x400; n++)
            ___VECTOR_RAM[n] = ___VECTOR_ROM[n];
}
/* Point the VTOR to the new copy of the vector table */
SCB_VTOR = (uint_32)___VECTOR_RAM;
```

6 MCF52259EVB 的 MQX 引导

本部分通过 MCF52259EVB 板上的 MQX 引导示例，介绍如何使用引导加载程序。

本部分介绍以下步骤：

- 准备设置
- 准备映像文件
- 编译应用
- 运行应用

6.1 准备软件和硬件

所需软件：

- CodeWarrior 版本 7.2
- Freescale MQX 3.7
- 超级终端

所需硬件：

- 个人计算机(PC)
- MCF52259EVB 板，带电源
- USB 记忆棒
- USB 线缆
- USB 转 RS232 转换器

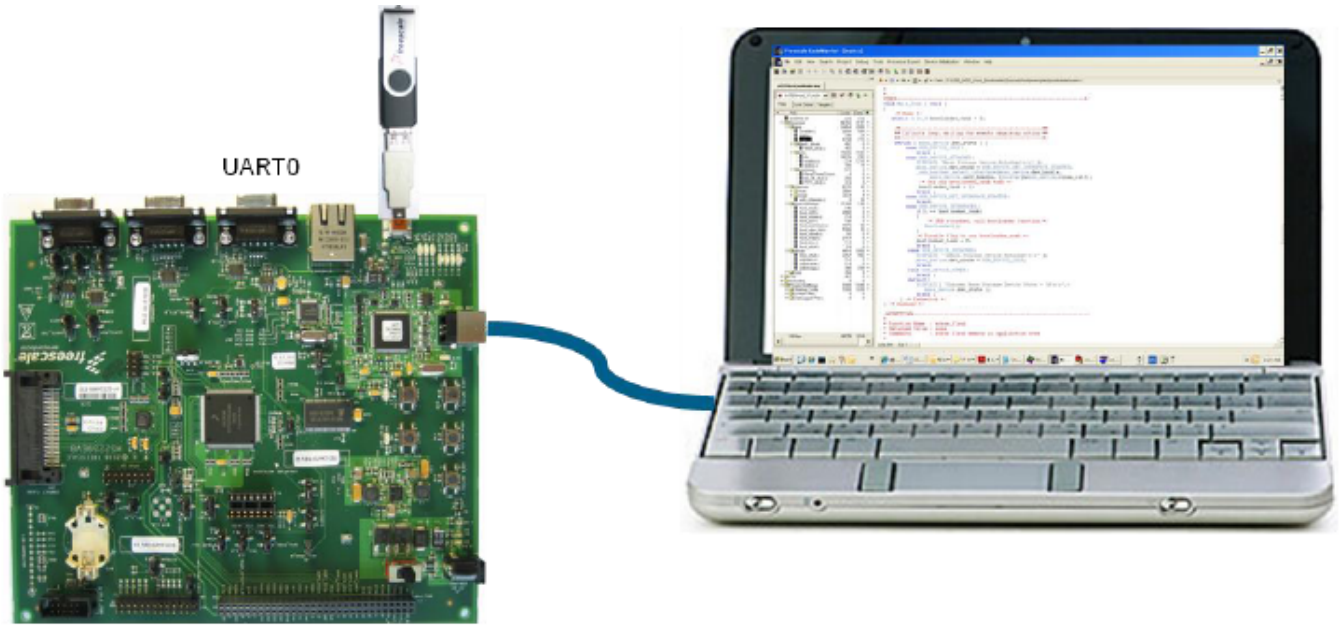

图 9. 硬件设置

硬件设置:

1. 连接 MCF52259EVB 板 J29 端口的引脚 2-3。
2. 将电源连接到 MCF52259EVB 板。
3. 使用 USB 线缆将 PC 连接到板的 USB BDM 端口。
4. 使用 USB 转 RS232 转换器将 PC 连接到板的 UART0 端口。
5. 打开 SW4 为板通电。

6.2 准备映像文件

本部分介绍有关创建引导加载程序将要加载的 MQX 映像的步骤。如果用户不想要编译 MQX 映像文件，则可以直接转到步骤 6，并使用提供的示例映像。

1. 设置 MQX，以便将向量表放入 RAM:
   - 打开文件 Freescale MQX 3.7\config\m52259evb\user_config.h。
   - 添加“#define MQX_ROM_VECTORS to 0”，表示在 RAM 中使用向量表。

```
#define MQX_ROM_VECTORS        0
```

2. 编译 MQX 库:
   - 打开项目 Freescale MQX 3.7\config\m52259evb\cwcf72\build_m52259evb_libs.mcp。
   - 按 F7 以编译库。

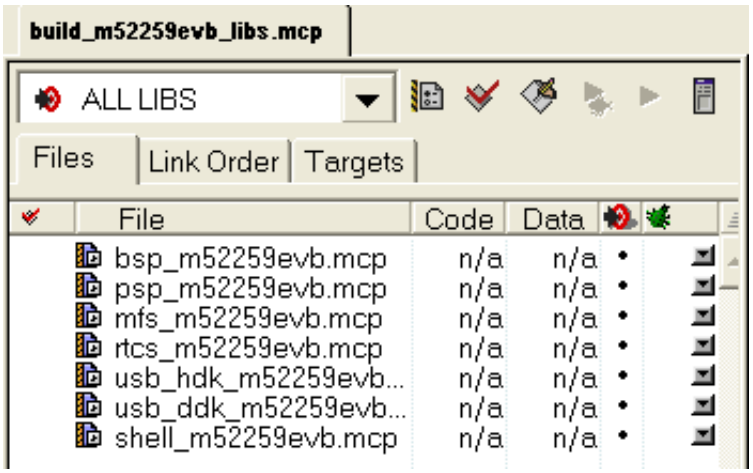

图 10. 编译 MQX 库

3. 打开 MQX 应用演示:
   - 打开项目 Freescale MQX 3.7\mfs\examples\mfs_usb\cwcf72\mfs_usb_m5329evb.mcp。
   - 选择“Project”->“Set Default Target”，并选择“Flash Release”或“Flash Debug”作为目标。

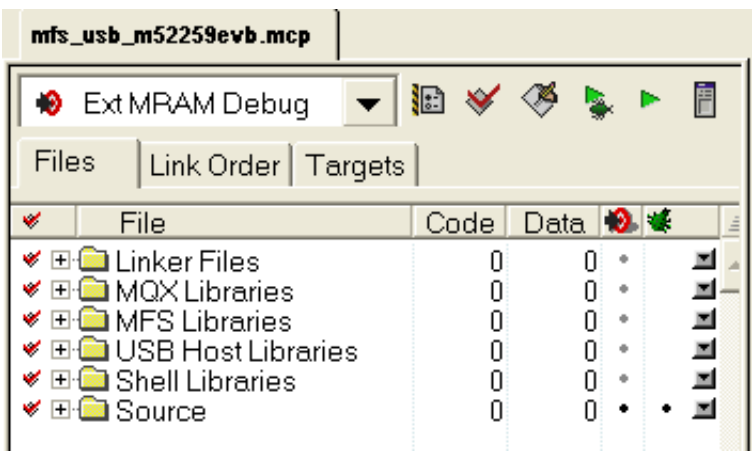

图 11. MFS USB 示例

4. 修改 intflash.lcf 链接器文件，以将此应用的 ROM 段移到应用闪存区。

```
vectorrom    (RX): ORIGIN = 0x00010000, LENGTH = 0x00000400
cfmprotrom   (RX): ORIGIN = 0x00010400, LENGTH = 0x00000020
rom          (RX): ORIGIN = 0x00010420, LENGTH = 0x0006FBE0 # Code+Const data
```

5. 选择生成 S-Record:
   - 选择“Setting”->“Int Flash Release Setting”或“Int Flash Debug Setting”。
   - 将弹出如下所示的窗口。
   - 选择“Linker”->“ColdFire Linker”。
   - 如果尚未选中“Generate S-Record File”复选框，请将它选中。

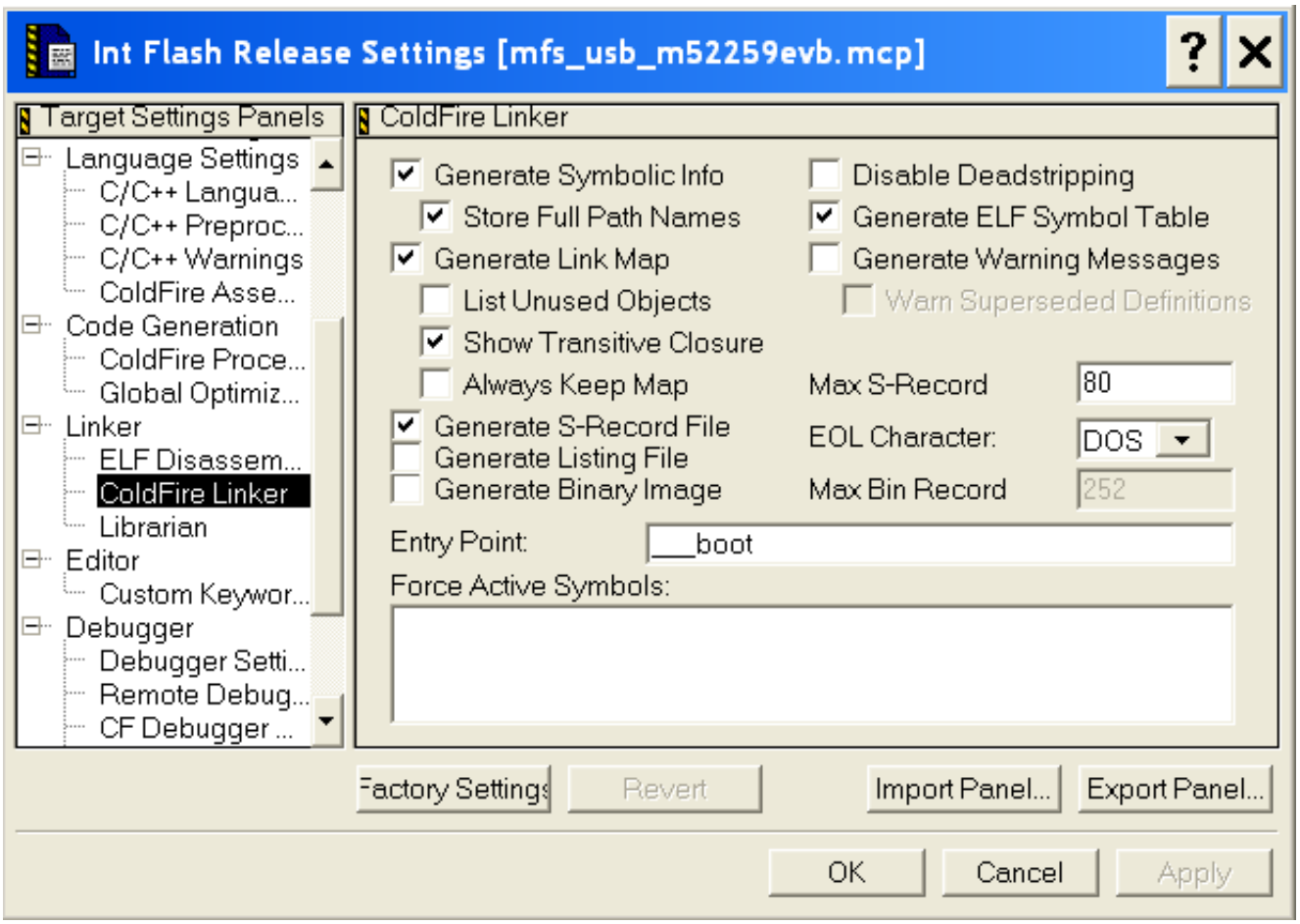

图 12. 生成 S_Record 和二进制映像

6. 编译项目：
   - 选择“Project”->“Make”。
   - 将在文件夹 Freescale MQX 3.7\mfs\examples\mfs_usb\cwcf72\m52259evb 中生成文件 intflash.elf.s19 或 intflash_d_elf.s19。
   - 文件夹 USB_MSD_Host_Bootloader\bootload_code\Image_files\support_printf\MCF52259\MQX_MFS_USB_Shell 包含这两个用于测试的示例映像文件。
   - S19 文件显示此映像的起始地址为 0x10000。
7. 将文件 intflash.elf.s19 或 intflash_d_elf.s19 重命名为 image.s19，并将它复制到 USB 记忆棒。

6.3 引导加载程序编程

以下步骤说明如何将引导加载程序烧录到设备中：

1. 打开引导加载程序项目 USB_MSD_Host_Bootloader\bootload_code\Source\Host\examples\bootloader\CodeWarrior \ m52259evb\m52259evb_bootloader.mcp。
2. 打开闪存编程器：
   - 选择“Tools”->“Flash Programmer”。
   - 将弹出“Flash Programmer”窗口。
   -

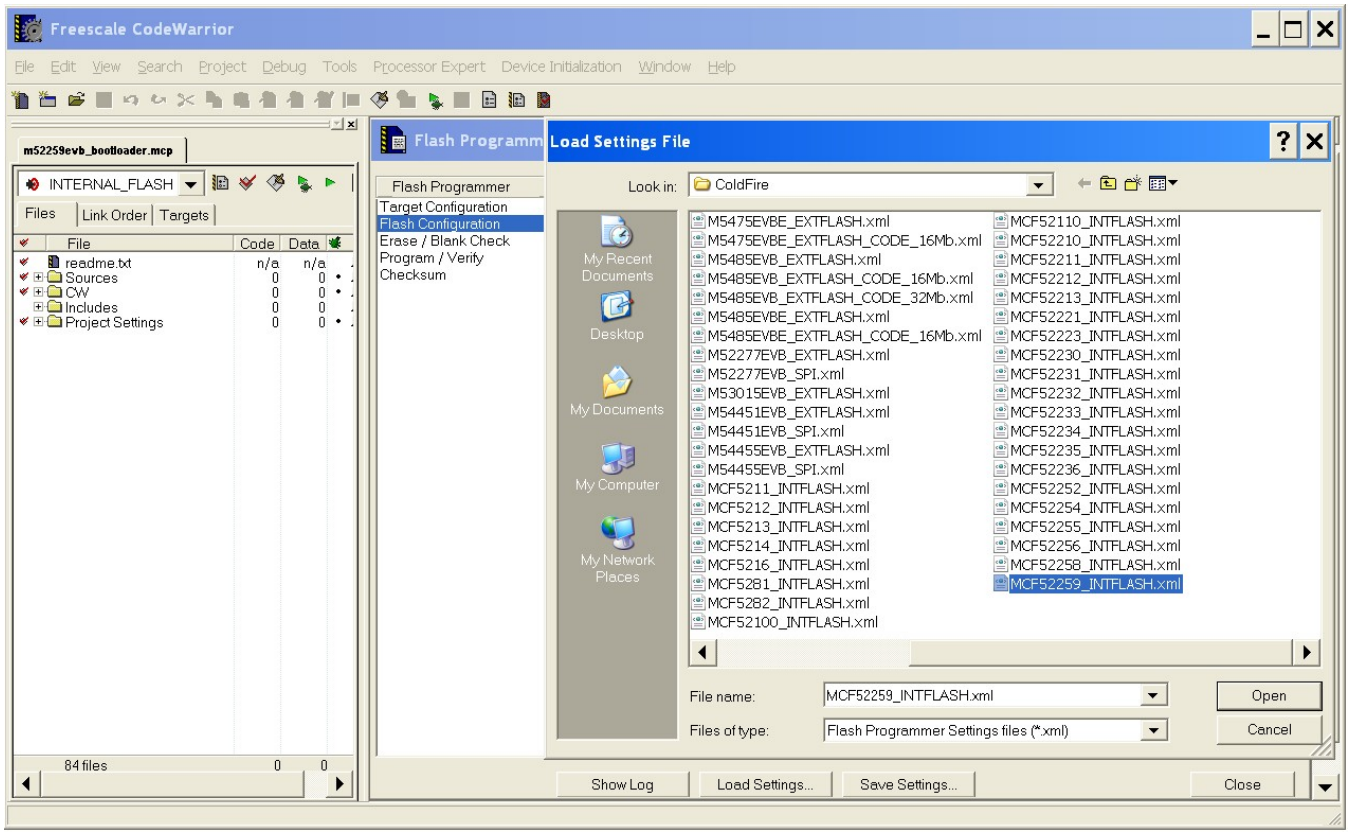

图 13. 闪存编程器

3. 加载设置:
   - 在弹出窗口的底部选择“Load Settings”。
   - 选择 M52259_INTELFLASH.xml。
   - 单击“Open”按钮。
4. 擦除闪存:
   - 单击“Erase / Blank Check”。
   - 选择“All Sectors” (如果未选择)。
   - 单击“Erase”擦除整个闪存区域。
   - 将显示状态“Erase Command Succeeded”。
5. 为引导加载程序编程:
   - 单击“Program / Verify”。
   - 单击“Use Selected File”。
   - 单击“Browse”并选择文件 USB_MSD_Host_Bootloader\bootload_code\Source\Host\examples\bootloader \CodeWarrior\ m52259evb\bin\MCF52259_INTERNAL_FLASH.elf.S19。
   - 单击“Program”按钮。
   - 将显示状态“Program Command Succeeded”。
6. 关闭 CodeWarrior。

6.4 打开 HyperTerminal

HyperTerminal 可以从运行引导加载程序或应用的设备中获取事件。 执行以下步骤可以配置 HyperTerminal 程序:

1. 打开 HyperTerminal 应用程序。

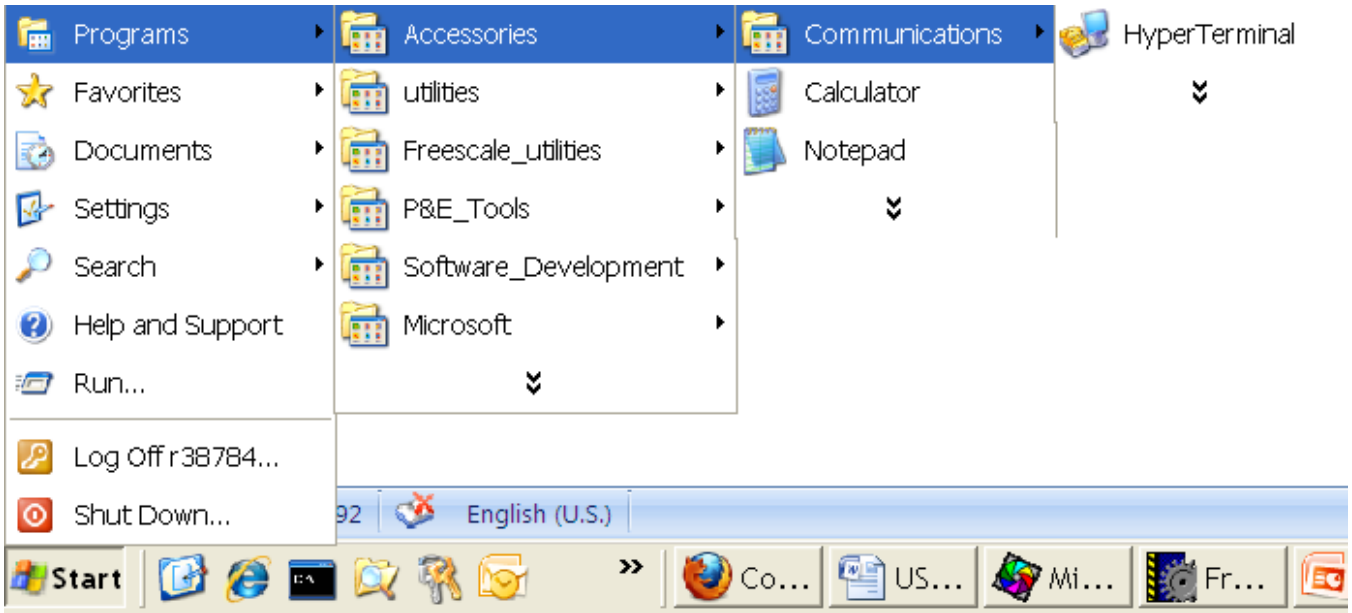

图 14. 打 HyperTerminal 应用程序

2. 已打开 HyperTerminal，如下图中所示：
   - 输入连接名称，例如 com8_115200_n_8_1。
   - 单击“确定”按钮。

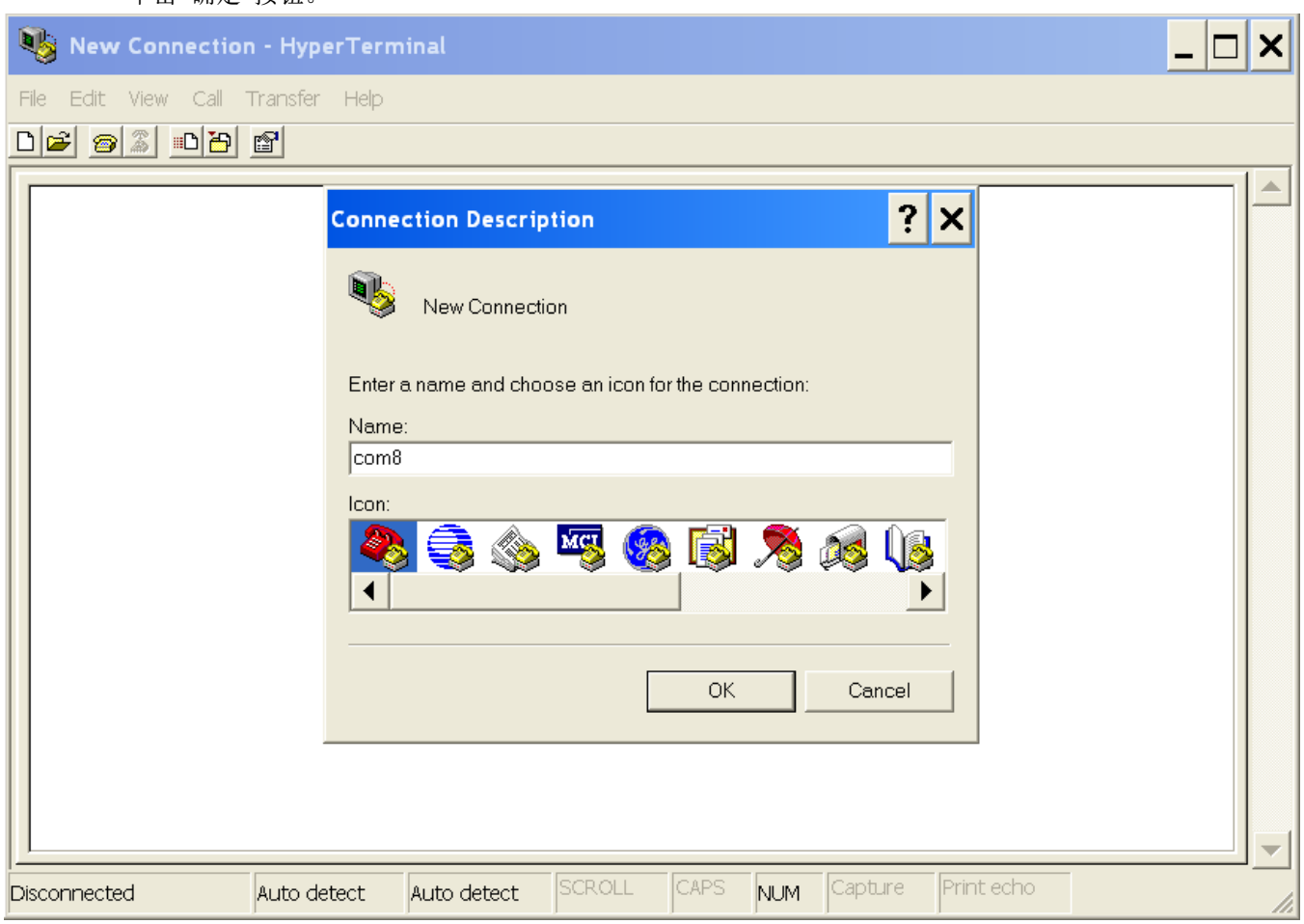

图 15. HyperTerminal GUI

3. 将显示如下图所示的窗口。选择适合与 USB-RS232 COM 端口进行连接的 COM 端口。

图 16. 使用 USB-RS232 COM 端口建立连接

4. 配置以下 COM 属性：
   - 每秒位数：115,200
   - 数据长度：8
   - 奇偶校验：无
   - 停止位：1
   - 单击“确定”按钮。

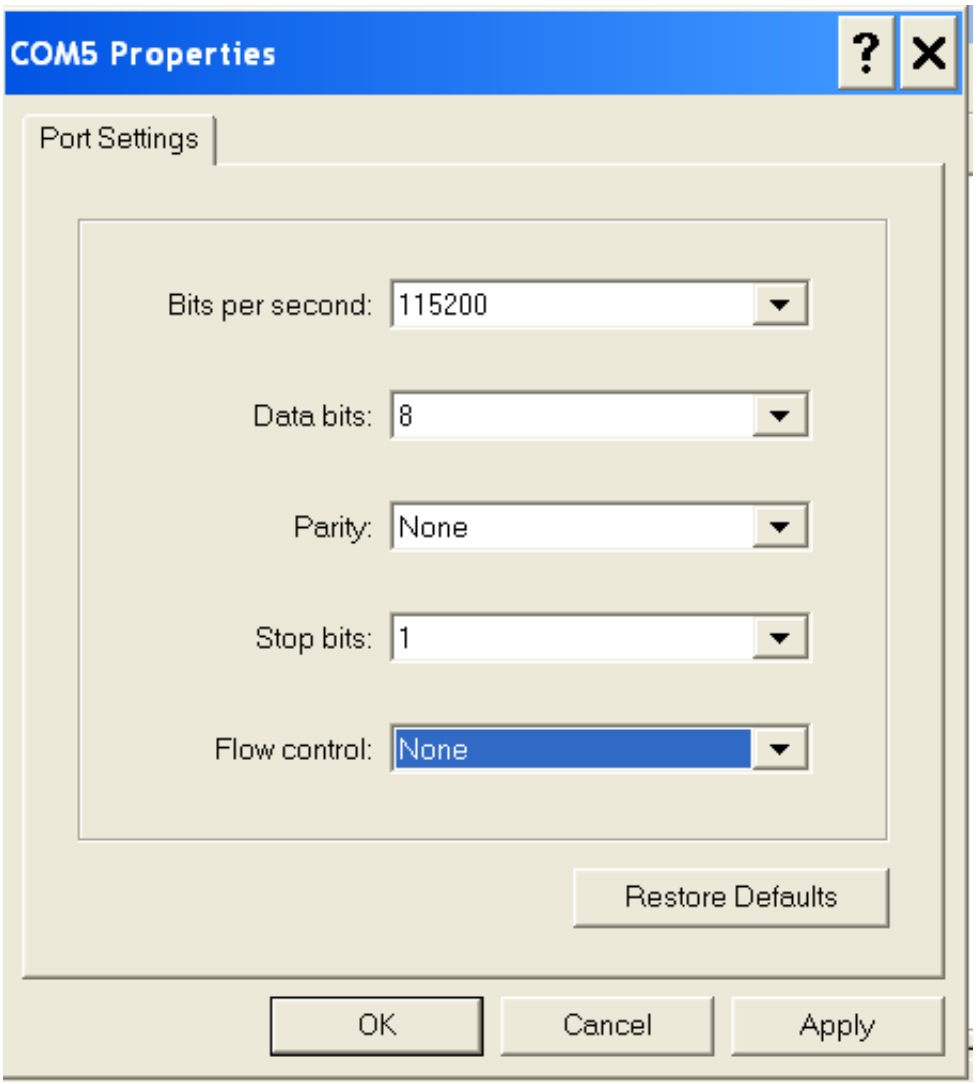

图 17. COM 属性

5. 已打开 HyperTerminal。

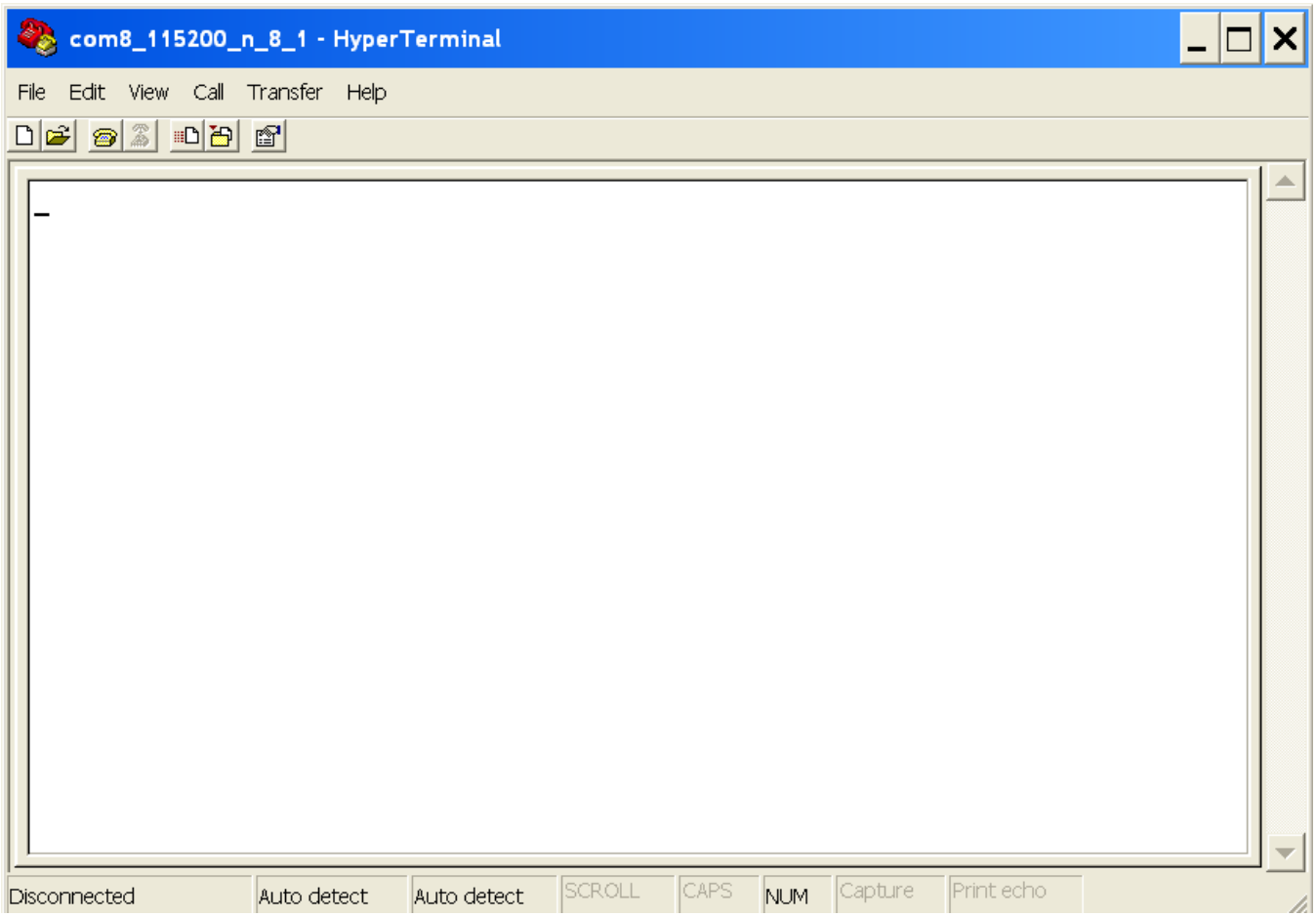

图 18. 已打开 COM 端口

6.5 运行引导加载程序

可以使用以下步骤来运行引导加载程序程序:

1. 按复位按钮将板复位，以运行引导加载程序。由于没有任何用户应用程序，因此，系统将进入引导加载程序模式。HyperTerminal 程序中将会显示下图中所示的消息。

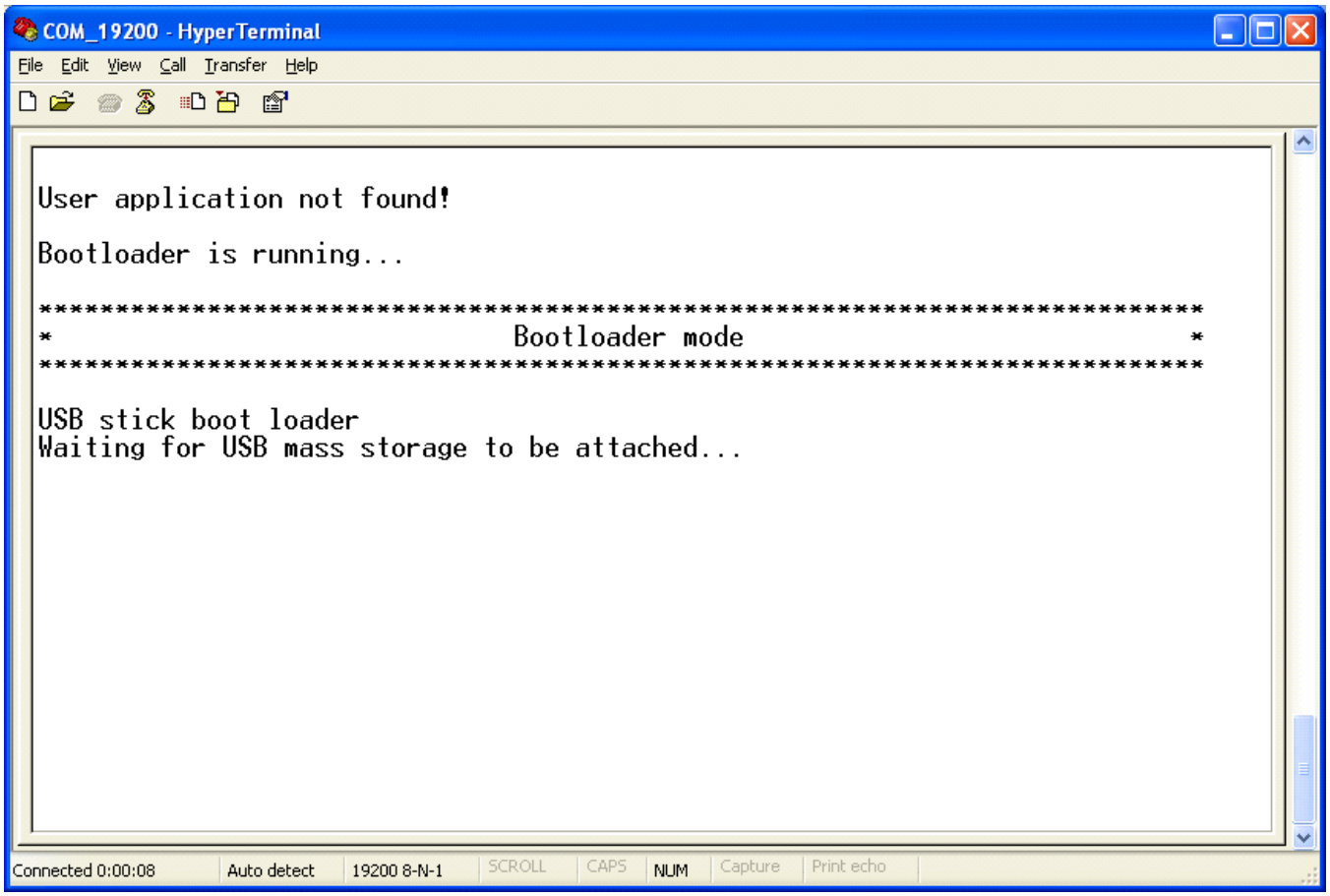

图 19. 运行引导加载程序

2. 将包含 image.s19 文件的 USB 记忆棒插入板的 USB mini-B 端口。下图显示了 HyperTerminal 程序中将会出现的消息。

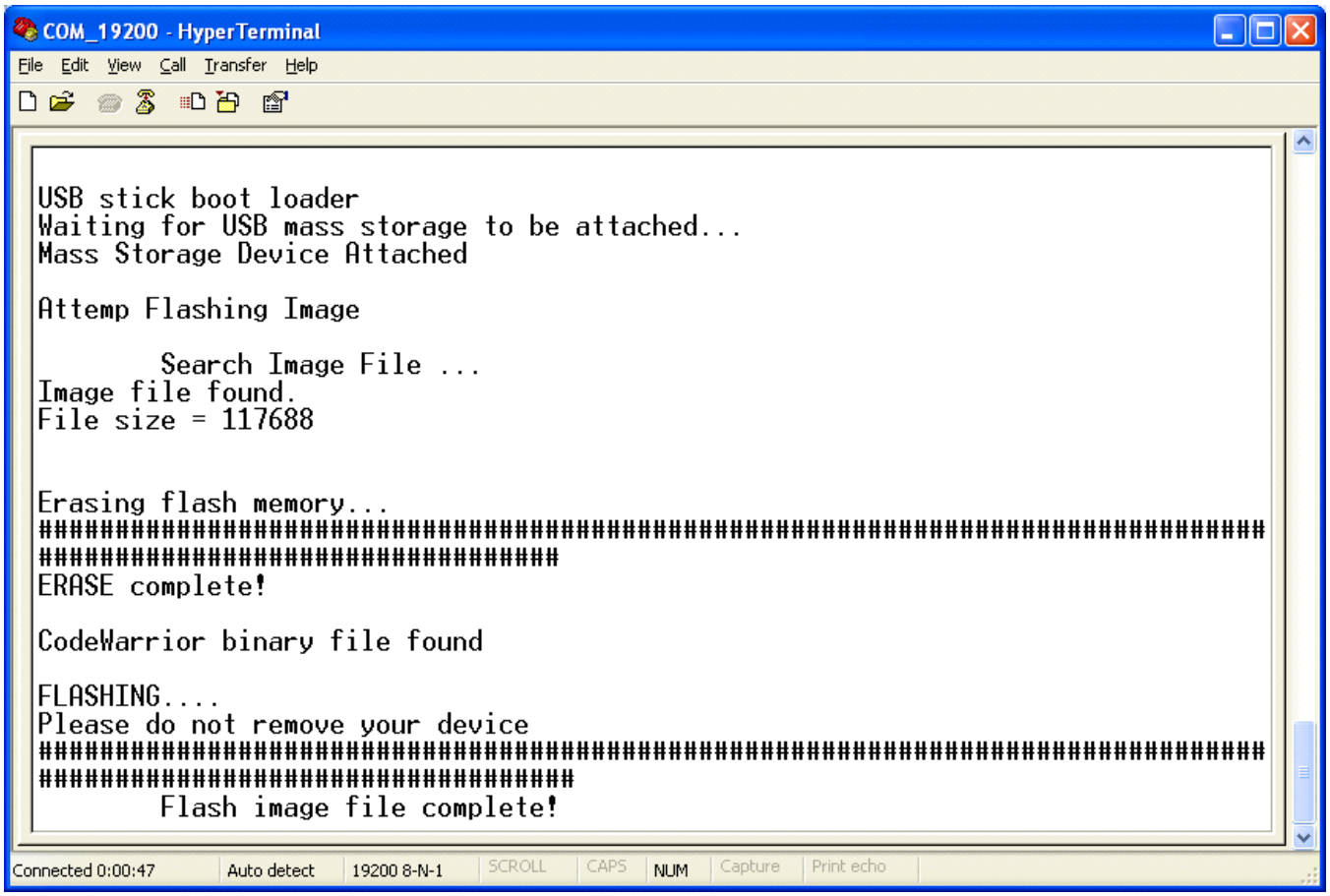

图 20. 引导加载程序消息

3. 插入 USB 记忆棒后，引导加载程序将尝试查找有效的映像文件，并将该文件编程到设备中。将出现下列结果之一：
   - 无响应 - 未识别记忆棒，请使用另一个记忆棒重试。
   - “Flash image file complete”- 已成功地将该映像文件编程到设备中。
   - “Image file not found”- 映像文件未载入记忆棒，或者记忆棒与系统不兼容。如果是后者，请使用另一个记忆棒重试。
   - “ERASE complete!”- 如果未显示“Flash image file complete”消息，则表示未将该映像文件编程到设备中。处理映像文件或者烧录映像文件时出现了一些错误。
4. 成功地将 MQX 应用编程到设备中后，请将板复位以运行用户应用。

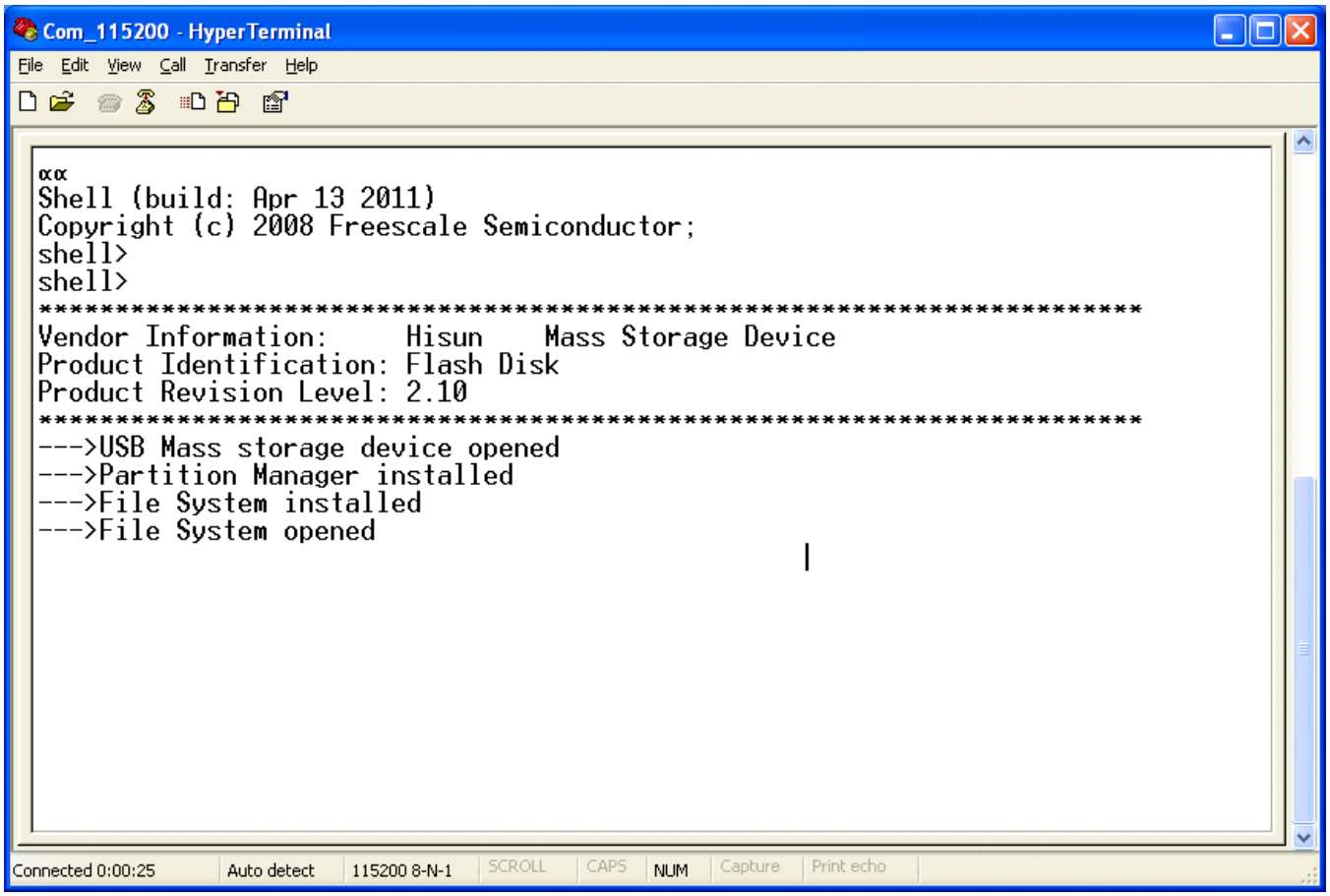

图 21. MQX 应用正在运行

5. 返回引导加载程序模式:
   - 按 SW1 和复位按钮。
   - 松开复位按钮。
   - 松开 SW1 按钮。
   - 引导加载程序程序应开始运行。

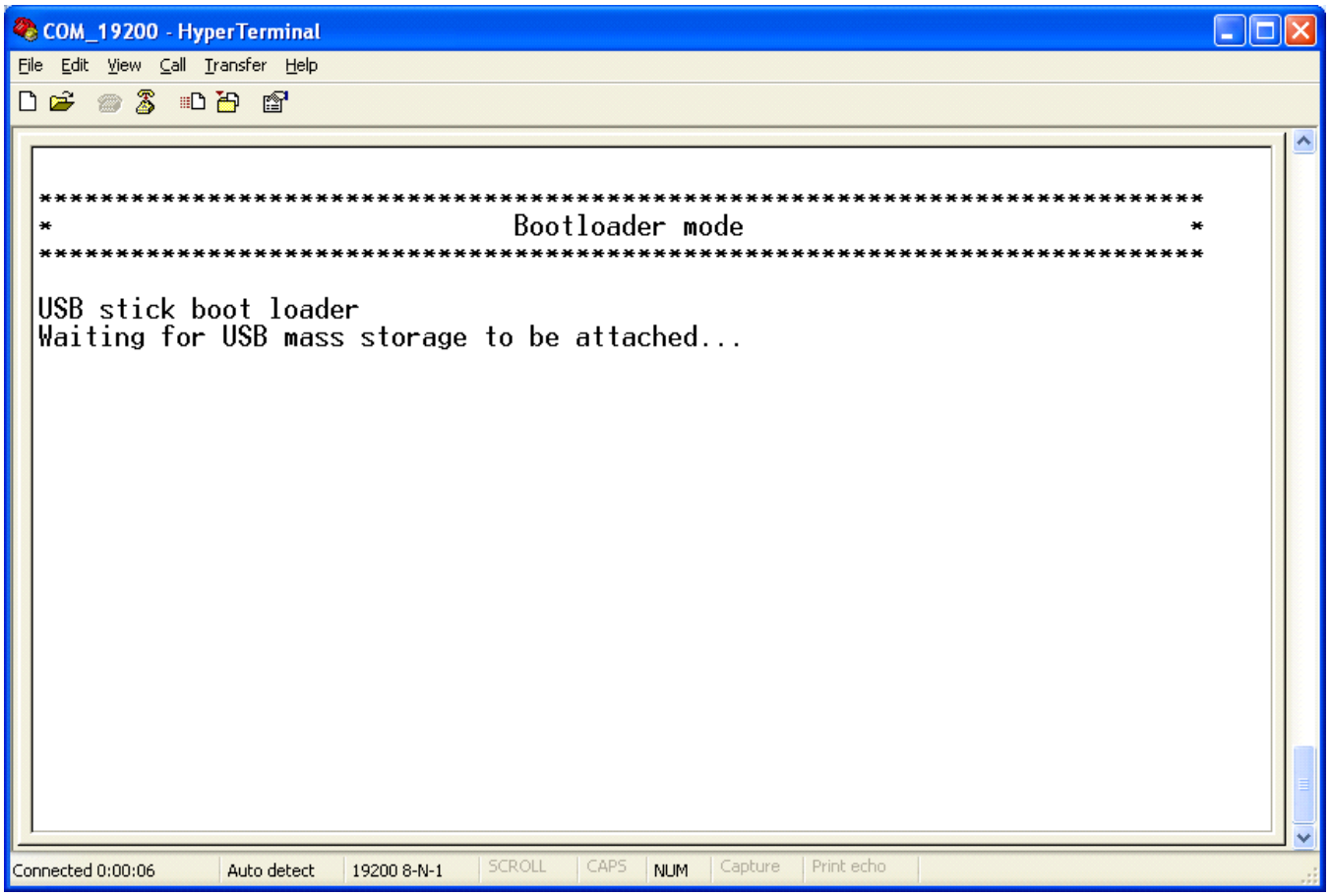

图 22. 返回引导加载程序应用

7 TWR-K60N512 的 MQX 引导

本部分通过 K60N512 板上的 MQX 引导示例，介绍如何使用引导加载程序。

本部分介绍以下步骤:

- 准备设置
- 准备映像文件
- 编译应用
- 运行应用

7.1 准备软件和硬件

所需软件:

- CodeWarrior 版本 10.1
- Freescale MQX 3.7
- P&E OSBDM OSJTAG Virtual Serial Toolkit (从 **pemicro.com/Docs and Downloads** 下载)

所需硬件:

- 个人计算机(PC)

- 面向 MK60N512VMD100 的 TWR-K60N512 Kinetis 开发套件
- USB 记忆棒
- USB 线缆

图 23. 硬件设置

硬件设置:

1. 连接 K60 控制器板 J6 端口的引脚 1-2
2. 连接 K60 控制器板 J9 端口的引脚 1-2
3. 连接 TWR-SER 板 J16 端口的引脚 1-2
4. 装配好塔式系统套件。
5. 使用 USB 线缆将 PC 连接到 K60 控制器板的 USB BDM 端口(J13)。

7.2 准备映像文件

本部分介绍有关创建引导加载程序将要加载的 MQX 映像的步骤。如果用户不想要编译 MQX 映像文件，则可以直接转到步骤 5，并使用提供的示例映像。

1. 打开 MQX 应用演示项目:
   - 打开 CodeWarrior 10.1。
   - 将工作区设置为 Freescale\Freescale MQX 3.7\mfs\examples\mfs_usb\cw10。

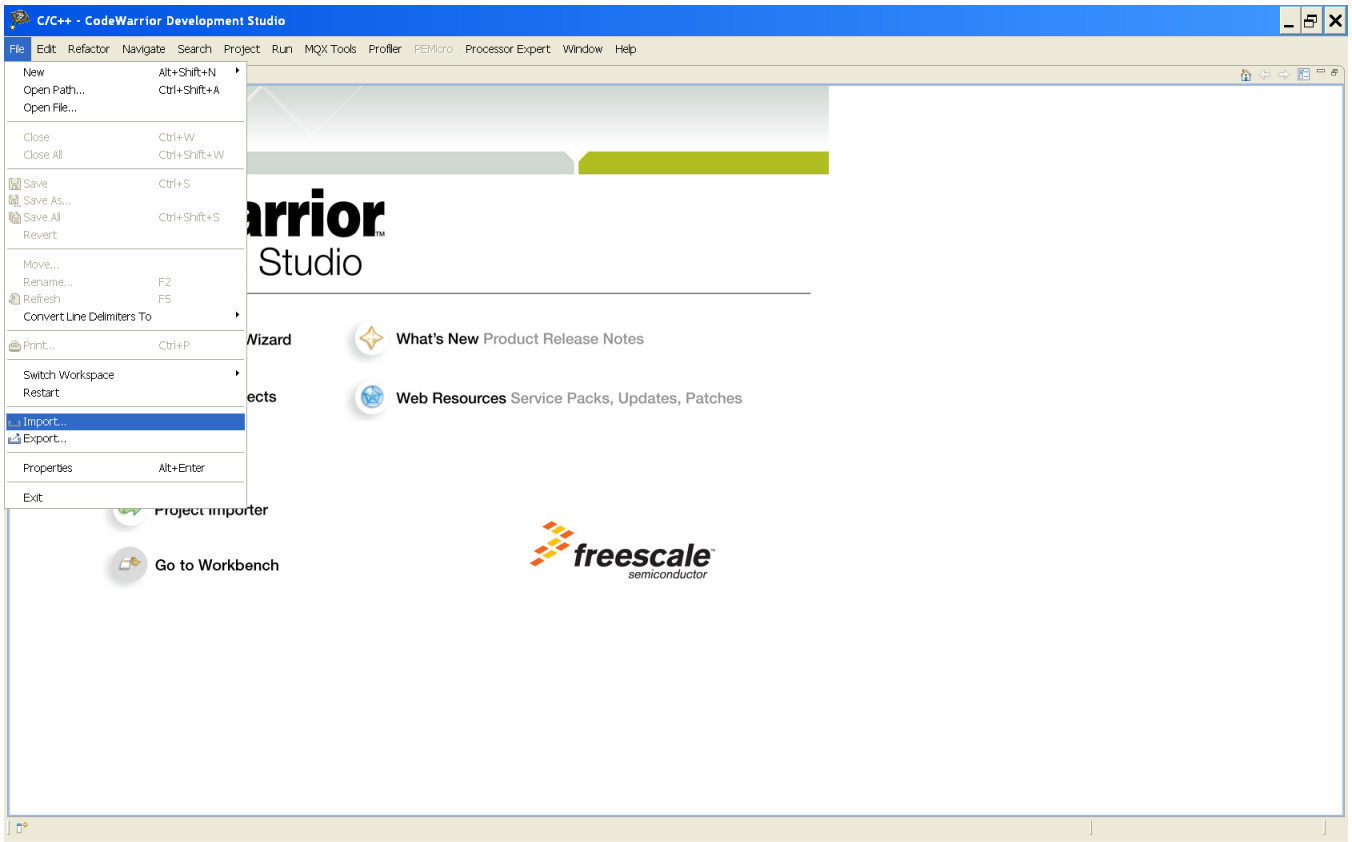

图 24. CodeWarrior 10.1

2. 导入项目：
   - 选择“File”->“Import”。

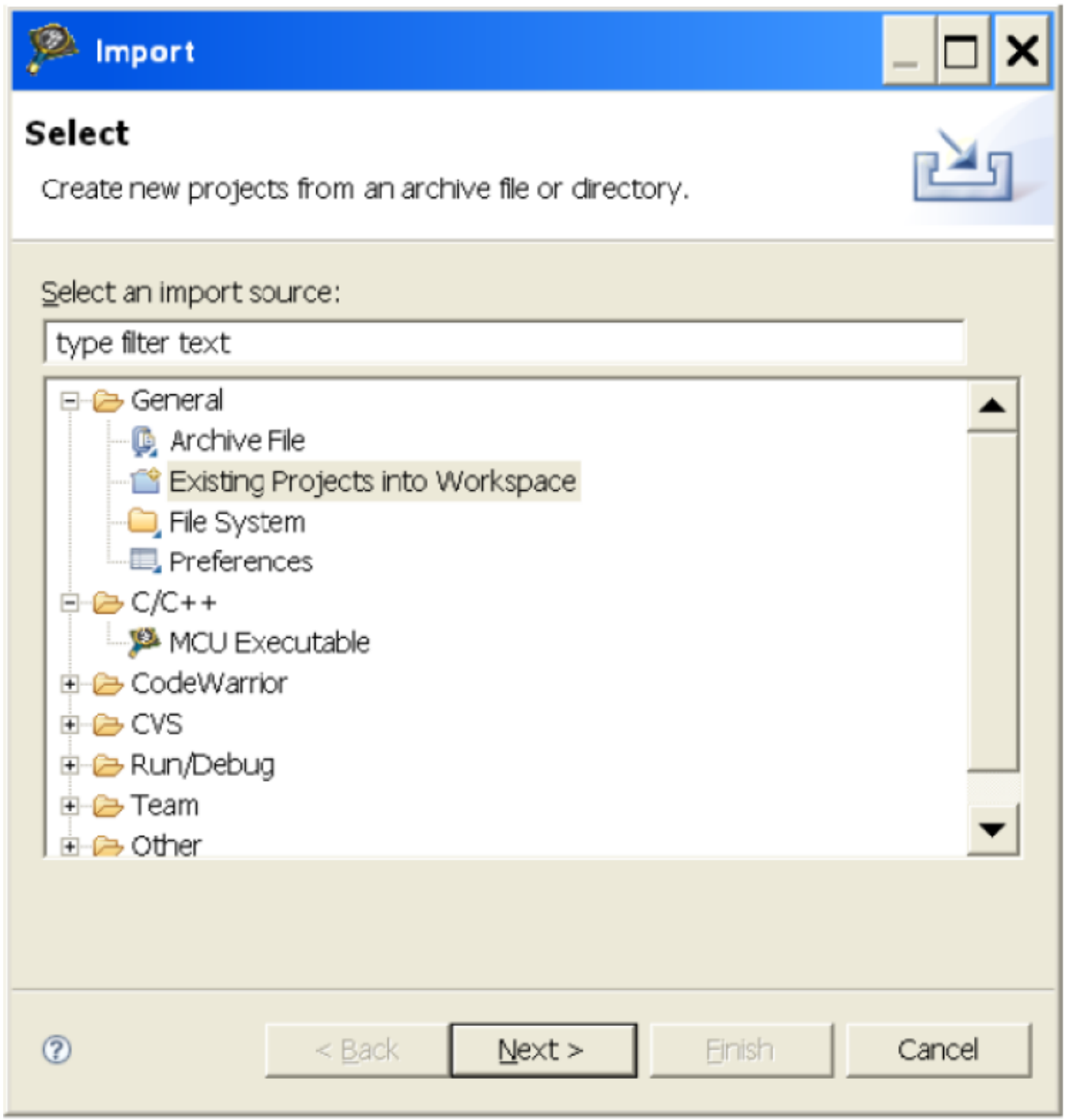

图 25. 导入现有项目

- 展开 General 目录。
- 选择要导入到工作区中的现有项目。
- 如果“Copy projects into workspace”复选框已选中，请将它取消选中。
- 单击“Next”按钮。

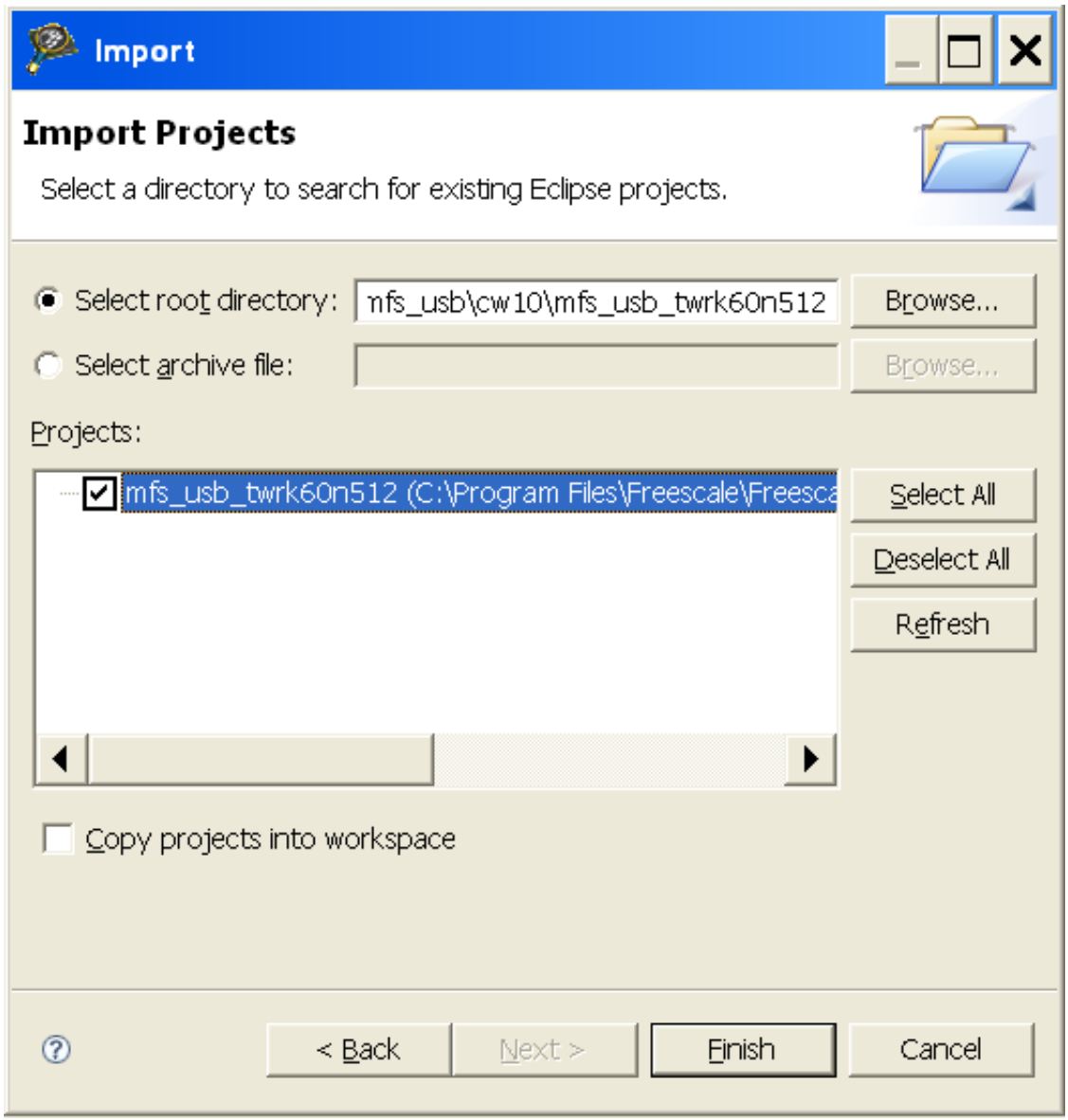

图 26. 导入 MFS_USB_TWRK60N512 项目

- 选择选项“Select root directory”。
- 单击“Browse”按钮。
- 选择项目 Freescale\Freescale MQX 3.7\mfs\examples\mfs_usb\cw10\mfs_usb_twrk60n512。
- 单击“Finish”按钮。
- 如果弹出了“Remote System Missing”窗口，请单击“Yes”按钮
- 关闭“Target Tasks”和“Welcome”窗口（如果已显示）。

3. 选择项目：

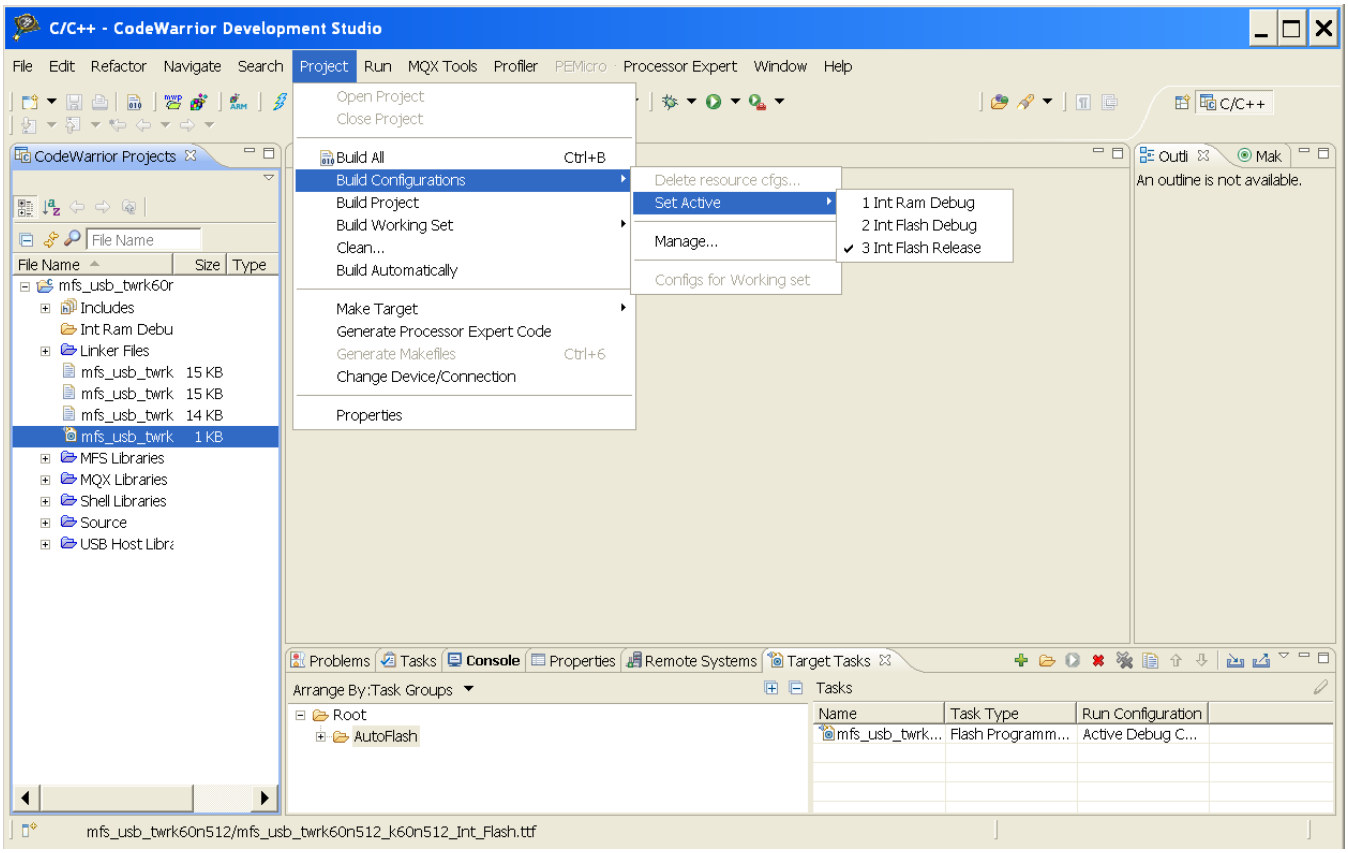

图 27. 选择项目

- 选择项目“mfs_usb_twrk60n512”。
- 选择“Project”->“Build Configurations”->“Set Active”->“Int Flash Release”。

4. 修改 intflash.lcf 链接器文件，以将 ROM 段移到应用闪存区。

```
/* original */
vectorrom (RX): ORIGIN = 0x00000000, LENGTH = 0x00000400
cfmprotrom (RX): ORIGIN = 0x00000400, LENGTH = 0x00000010
code (RX): ORIGIN = 0x00000410, LENGTH = 0x0007FBF0 # Code+Const data
/* modified */
vectorrom (RX): ORIGIN = 0x00010000, LENGTH = 0x00000400
cfmprotrom (RX): ORIGIN = 0x00010400, LENGTH = 0x00000010
code (RX): ORIGIN = 0x00010410, LENGTH = 0x0006FBF0 # Code+Const data
```

5. 编译项目：
   - 在 CodeWarrior 的“Projects”窗口中选择“mfs_usb_twrk60n512 : Int Flash Release”。
   - 选择“Project”->“Build Project”。
   - 文件夹 Freescale MQX 3.7\mfs\examples\ mfs_usb\cw10\mfs_usb_twrk60n512\Int Flash Release 中将会生成 intflash.afx.s19 文件。
   - 文件夹 USB_MSD_Host_Bootloader\bootload_code\Image_files\support_printf\ K60\MQX_USB_Shell 包含这个用于测试的示例映像文件。
6. 请将文件 intflash.afx.s19 重命名为 image.s19，并将它复制到 USB 记忆棒。

7.3 引导加载程序编程

以下步骤说明如何在设备中进行引导加载程序编程：

1. 切换工作区：

- 选择“File”->“Switch Workspace”，然后选择目录 USB_MSD_Host_Bootloader\bootload_code\Source\Host\examples\bootloader\cw10。

2. 导入项目：
   - 选择“File”->“Import”。
   - 展开 General 目录。
   - 选择“Existing Projects into workspace”。
   - 单击“Next”按钮。
   - 选择选项“Select root directory”。
   - 单击“Browse”按钮。
   - 选择项目 USB_MSD_Host_Bootloader\bootload_code\Source\Host\examples\bootloader\cw10\ kinetis_k60。
   - 单击“Finish”按钮。
   - 如果弹出了“Remote System Missing”窗口，请单击“Yes”按钮。
   - 关闭“Target Tasks”和“Welcome”窗口（如果已显示）。
3. 选择项目：
   - 在“CodeWarrior Projects”窗口中选择项目“kinetis_k60”。
   - 选择“Project”->“Build Configurations”->“Set Active”->“MK60N512VMD100_INTERNAL_FLASH”。
4. 编译项目：
   - 选择“Project”->“Build Project”。
5. 运行配置：
   - 选择“Run”->“Debug Configurations”。

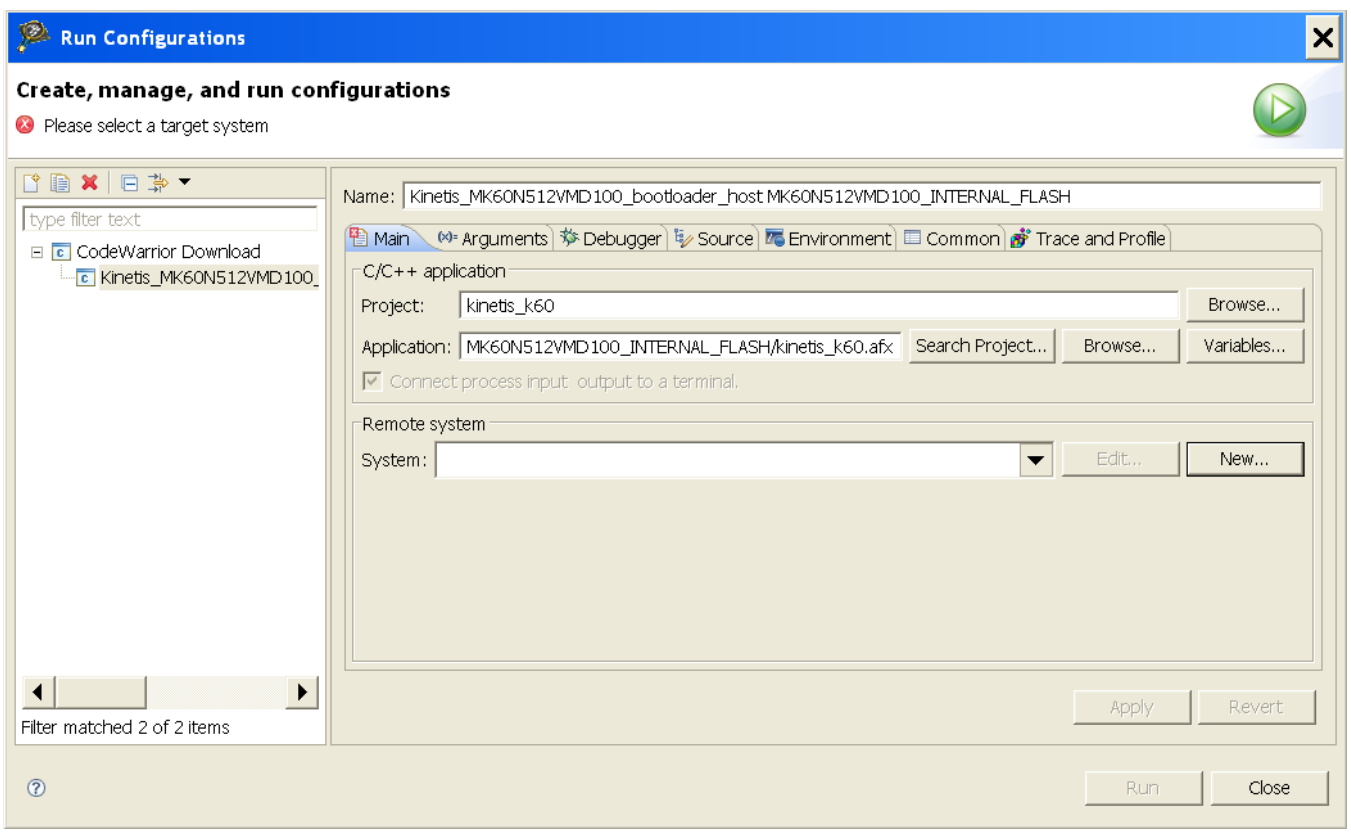

图 28. 运行配置

- 右键单击“CodeWarrior Download”并选择“New”。
- 在项目中键入一个名称，例如，kinetis_k60 MK60N512VMD100_INTERNAL_FLASH_OSJTAG。
- 单击“Browse”按钮。
- 单击“OK”按钮选择 kinetis_k60。
- 单击“Search Project”按钮。
- 单击“OK”按钮选择 kinetis_k60_afx 应用。
- 单击“New”按钮打开新连接。

6. 新建连接：

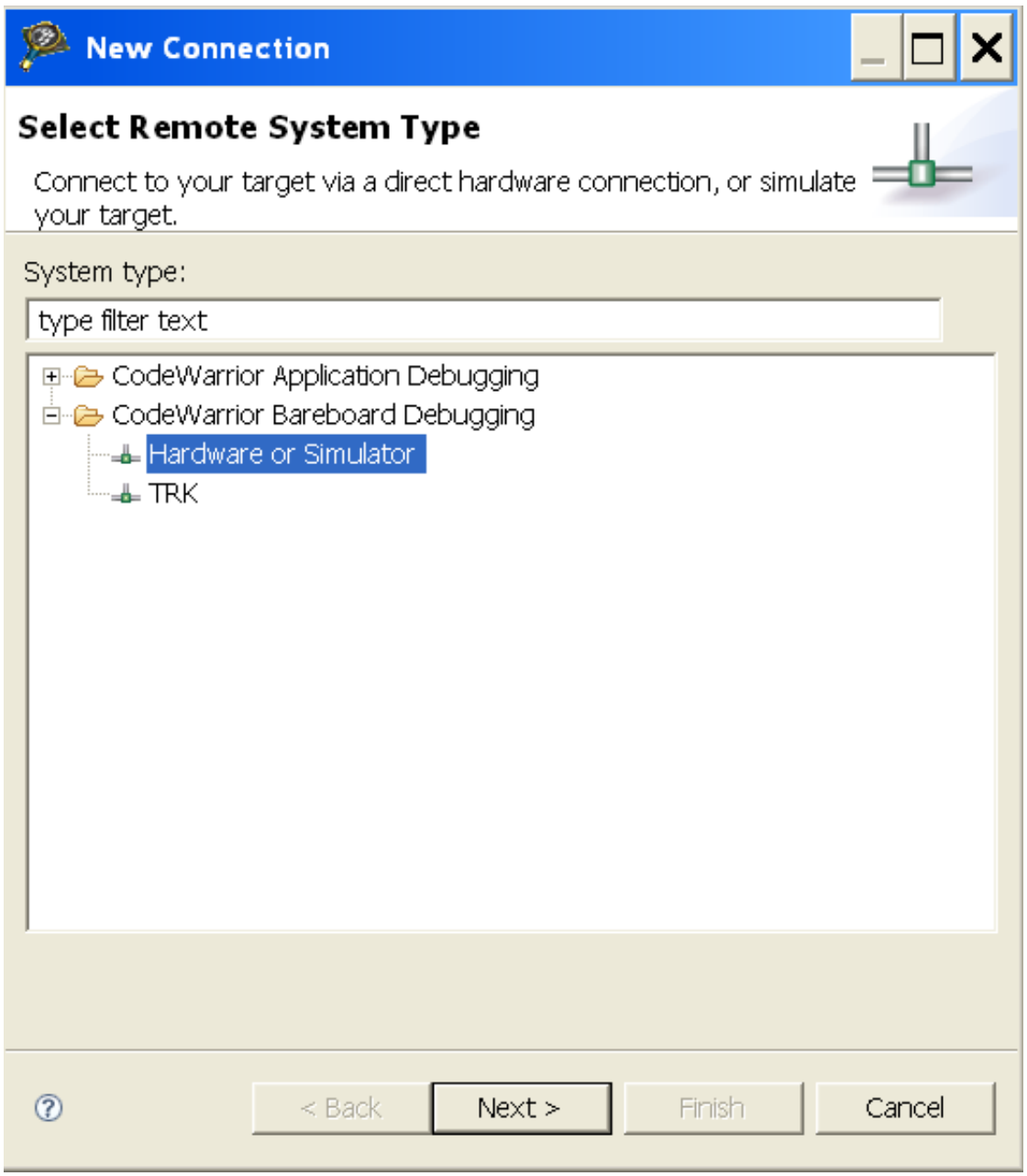

图 29. 新建连接

- 选择“Hardware or Simulator”。
- 单击“Next”按钮。

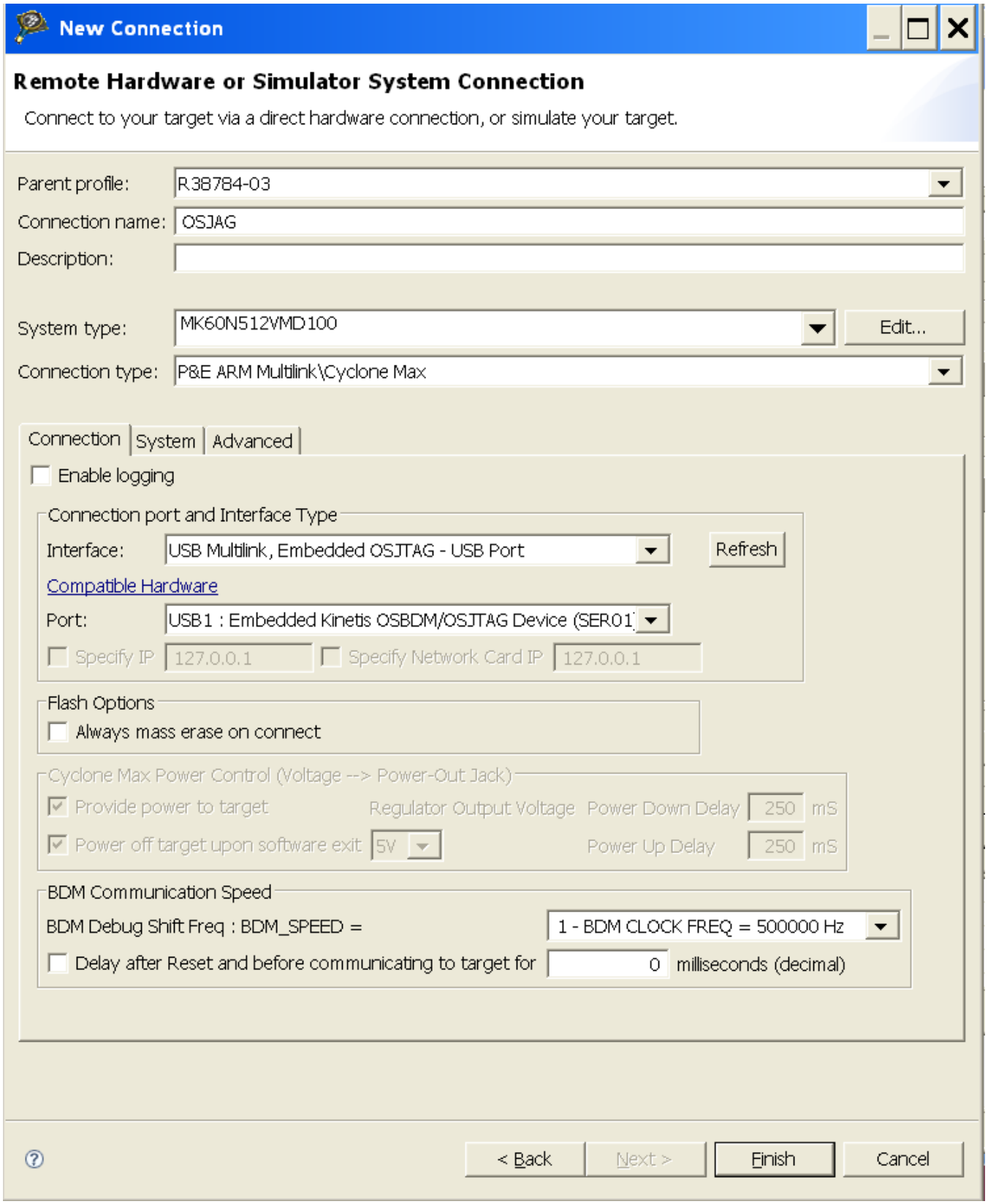

图 30. 远程硬件或仿真器系统连接

- 在“Connection name”框中键入连接名称，例如，OSJTAG。
- 在“System type”中，选择“kinetis_k60”->“MK60N512VMD100”。

- 在“Connection type”中，选择“P&E ARM Multilink\Cyclone Max”。
- 单击“Finish”按钮。

7. 为引导加载程序编程：
   - 单击“Debug”按钮在设备中进行引导加载程序编程。
8. 关闭 CodeWarrior。

7.4 打开虚拟终端

P&E OSBDM OSJTAG 终端实用程序可以从运行引导加载程序或应用的设备中获取事件。对于 TWR-K60N512 塔式系统板，通信是通过 K60 的 UART5 以及 USB OSBDM 实现的。

1. 打开 P&E 虚拟串行终端：
   - 选择“开始”菜单 ->“程序”->“Freescale”->“P&E OSBDM OSJTAG Virtual Serial Toolkit”->“实用程序”->“终端实用程序”。

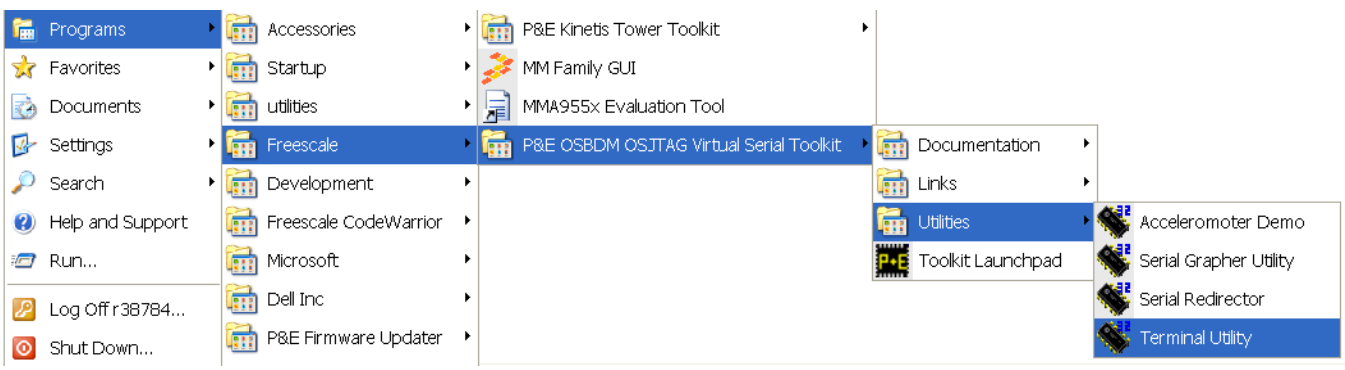

图 31. P&E 虚拟串行终端

2. 打开串行端口：
   - 确保已选择波特率为 115,200 的 USB COM。
   - 单击“打开串行端口”按钮。

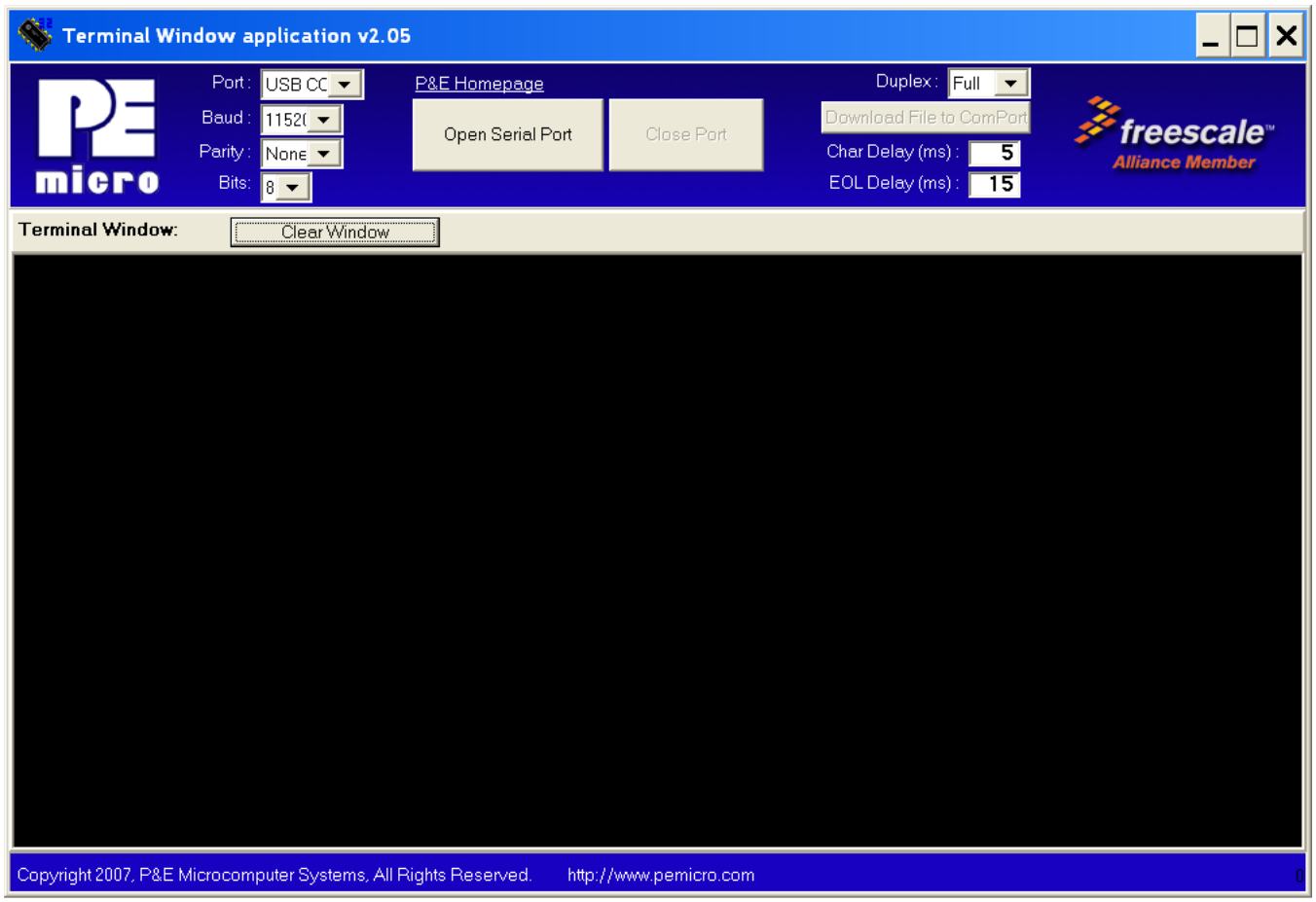

图 32. HyperTerminal GUI

7.5 运行引导加载程序

可以使用以下步骤来运行引导加载程序:

1. 按复位按钮将板复位，以运行引导加载程序。由于没有任何用户应用程序，因此，系统将进入引导加载程序模式。将显示消息“Waiting for USB mass storage to be attached …”。如果用户应用区域不是空白的，请执行步骤 4 进入引导加载程序模式，然后转到步骤 2。

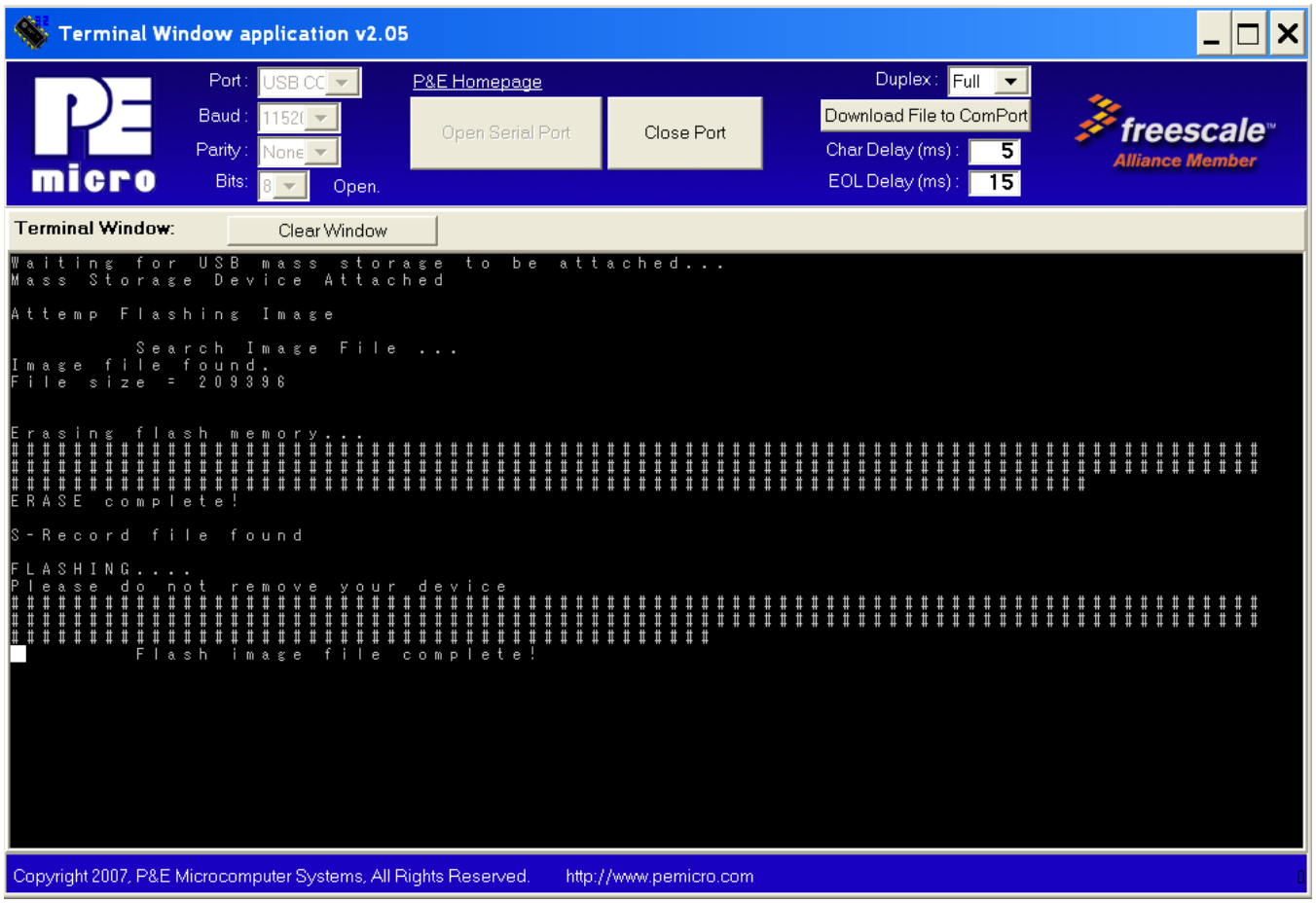

图 33. 运行引导加载程序

2. 将 USB 大容量存储设备插入 TWR-SER 板的 USB 端口。插入 USB 记忆棒后，系统将尝试查找有效的映像文件，并将该映像文件编程到设备中。可能会出现下列结果之一：
   - 无响应 - 未识别记忆棒，请换用另一个记忆棒重试。
   - “Flash image file complete”- 已成功地将该映像文件编程到设备中。
   - “Image file not found”- 映像文件未载入记忆棒，或者记忆棒与系统不兼容。如果是后者，请使用另一个记忆棒重试。
   - “ERASE complete!”- 如果未显示“Flash image file complete”消息，则表示未将该映像文件编程到设备中。处理或编程时出现了一些错误。
3. 成功地将 MQX 应用编程到设备中后，请将板复位以运行用户应用。

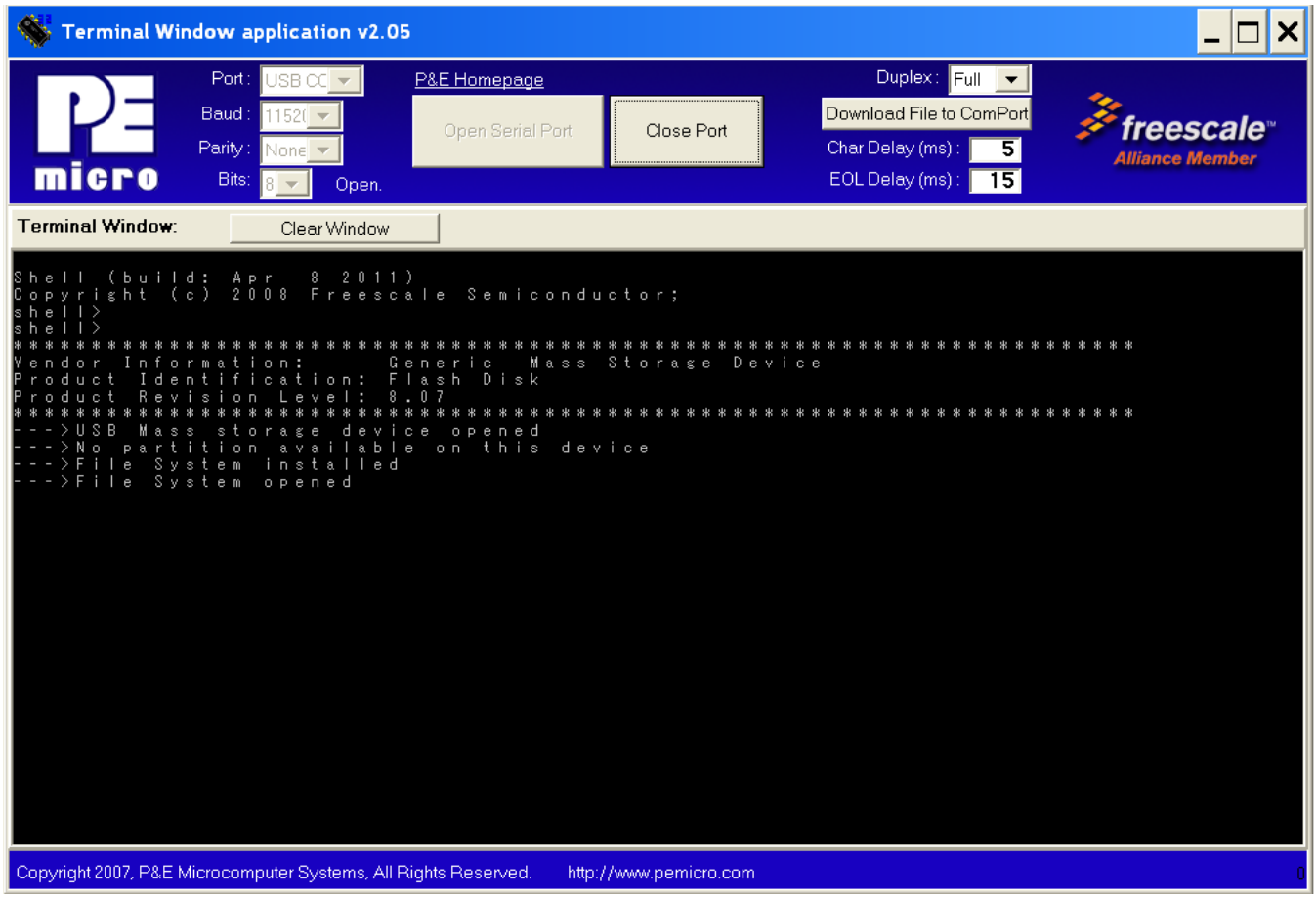

图 34. MQX 应用正在运行

4. 返回引导加载程序模式:
   - 按 SW1 和复位按钮。
   - 松开复位按钮。
   - 松开 SW1 按钮。
   - 引导加载程序程序应开始运行。

8 自定义

本部分介绍自定义时要考虑的因素。

根据其他平台修改示例时，必须考虑以下因素:

- BDM 或编程接口
- 重新进入引导加载程序模式的方法
- USB 集线器驱动程序
- 调试消息

8.1 BDM 或编程接口

飞思卡尔在 ColdFire 和 Kinetis MCU 开发板上提供了以下嵌入式 BDM 接口，您无需使用外部 BDM 硬件。

- P&E Multilink/Cyclone Pro，例如 M51JM128EVB
- CFV1 开源 BDM，例如 TWR-MCF51MM

- PEMICRO_USB，例如 M52259EVB
- ColdFire v2-v4 JM60 OSBDM，例如 TWR-MCF5225X
- USB Multilink，嵌入式 OSJTAG - USB 端口，例如 TWR-K60N512

在使用其他 BDM 接口时，用户必须正确设置相应的 BDM 接口。除 PEMICRO_USB 以外的其他所有嵌入式 BDM 都提供了虚拟 COM 接口。

8.2 重新进入引导加载程序模式的方法

在我们的演示中，上电期间按 SW1 会强制系统进入引导加载程序模式。用户可以在 main.c 文件中修改 Switch_mode 函数，以选择通过其他输入引脚或其他方法进入引导加载程序模式。

8.3 USB 集线器驱动程序

如果不需要在引导加载程序系统中支持集线器功能，则用户可以选择删除 USB 集线器驱动程序，以减少引导加载程序代码。为此，可以在 usb_classes.h 文件中取消定义 USBCLASS_INC_HUB 的标识符。

8.4 调试消息

可以通过 MCU 的 UART 端口显示调试消息。用户可以选择引导加载程序与最终用户通信的方法。例如，可以使用 LED 来指示引导加载程序状态，并禁用调试消息以减小引导加载程序代码的大小。可以通过在 derivative.h 文件中取消定义_DEBUG_标识符来禁用调试消息。

9 结语

我们针对飞思卡尔 32 位 ColdFire 和 Kinetis MCU 系列生成了 USB MSD 类引导加载程序示例代码。将 USB 记忆棒插入系统后，便可以在 MCU 中进行用户应用代码编程。

用户可以根据其他飞思卡尔 MCU 修改示例代码，也可以针对自己的应用自定义代码。

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number AN4368
Revision 1, 12/2012