Freescale Semiconductor
Application Note

Document Number:AN4775
Rev 0, 07/2013

Kinetis E 系列上的 IIC Boot Loader 设计

通过: **Wang Peng**

内容

1 概述.....1
2 简介.....1
3 软件体系结构.....2
3.1 转接板.....2
3.2 目标板.....3
4 内存分配.....8
5 结论.....9
6 参考.....9
7 术语表.....9
8 修订历史记录.....9

1 概述

在一个特定的场合，很多应用或产品都需要升级固件，以便修复某些发现的 Bug 或提高性能。 其中大多数的应用或产品都不使用专用的调试接口，而是使用 UART、USB、IIC 等通信接口。 这种情况下，就需要一个串行 Boot Loader 通过其中一个通信接口升级固件，而不需要调试器或特定的程序工具。

本应用说明将指导您使用 IIC 接口在 Kinetis E 系列 MCU 上设计 Boot Loader。

2 简介

Boot Loader 是一种内置固件，用于通过通信接口将应用代码编程到闪存。

本应用说明介绍如何使用 KE02Z Freedom 开发 (FRDM-KE02Z) 板将 PC 终端中的 UART 数据转换到 IIC 总线，以及如何与目标板（KE02Z 板）通信以执行目标应用代码的更新。 请参见下图。

© 2013 Freescale Semiconductor, Inc.

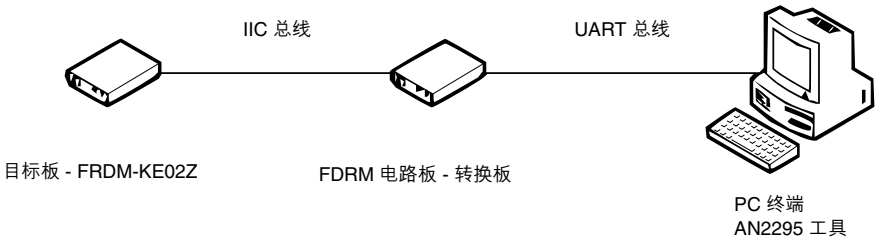

图 1. 顶视图

Boot Loader 利用 AN2295SW 软件工具，这些工具可从 **freescale.com** 下载，所有 Kinetis 产品都广泛地使用它们来实现 Boot Loader，以便通过 UART 接口更新应用代码。

转接板使用 Freedom 板 FRDM-KE02Z 将 UART 总线转换为 IIC 总线，重新将数据打包并传输到目标板。

目标板具有内置的 Boot Loader 代码，在收到命令和数据后，作为 I2C 从设备和转接板进行通信，并将应用代码编程到目标板上的闪存中。

本应用说明随附的软件 AN4775SW.zip 包含以下内容：

- 可以直接在 FRDM-KE02Z 板上运行的示例代码
- 应下载到目标板的“I2C_boot loader”
- 应下载到转接板的“Bridge_UartToIIC”
- 可以使用 PC 软件下载的项目“RTC_demo”用于生成 S19 文件

3 软件体系结构

本应用说明随附的软件 AN4775SW.zip（包含 win_hc08sprg.exe）可从 **freescale.com** 下载，可用来解码 S19 文件，并通过 FC 协议与转接板进行通信。

3.1 转接板

转接板通过 FC 协议与 PC 终端通信。 有关 FC 协议的详细信息，请参见以下网站上提供的《*AN2295*：适用于 *M68HC08* 和 *HCS08 MCU* 的开发者串行 *Boot Loader*》： **freescale.com**。

转接板将初始化为 IIC 主控并与目标板通信，使用 IIC 总线与目标板相互传输数据包；将使用数据长度与校验和重新打包数据帧。 以下是数据包的格式。

数据长度	原始数据帧	校验和

以下步骤解释了下图中显示的流程图。

1. 转接板向目标板发送 FC_CMD_HOOK(0x02)。
2. 然后，转接板读取目标板返回的状态，以确定目标板是在 Boot Loader 模式还是用户代码模式下工作。
3. 如果收到的状态为 FC_CMD_HOOK|0x80，则发送 0xFC 以开始和 PC 端握手，否则一直检查目标板的状态，直到收到 FC_CMD_HOOK|0x80 为止。
4. 其后，将接收目标板发送的数据帧，使用数据长度与校验和重新打包数据帧，并通过 IIC 总线发送打包的数据帧。
5. 此后，将读取从属设备发送的数据；接收的第一个数据用于确定从属设备是否已就绪。 如果已就绪（收到命令 |0x80），转接板则将向接收器指出已收到正确的应答。

例如，如果发送给从属设备的命令是 0x03，则收到的应答必须是 0x03|0x80。

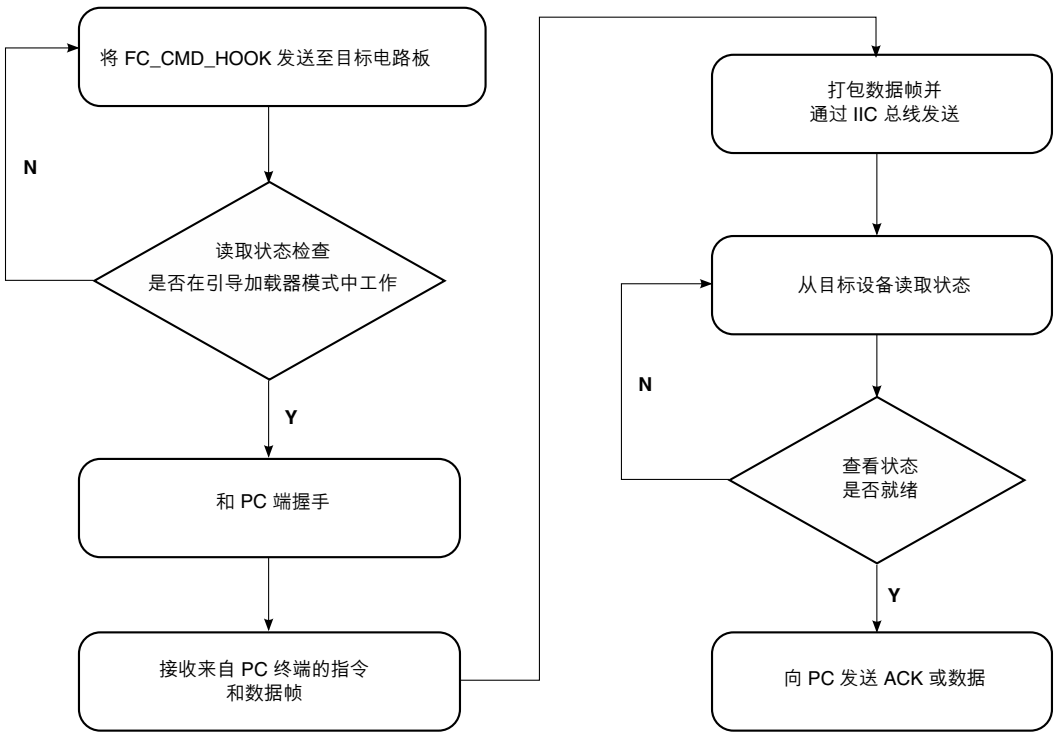

图 2. 转接板软件流程图

转接板充当 PC 终端与目标板之间的网桥，使用转接板可以通过 PC 将 S19 文件下载到目标板。

3.2 目标板

目标板包含内置 Boot Loader 代码。 通电后，目标板首先将检查工作模式，以确定自身是处于引导模式还是用户代码模式。 识别工作模式的方法之一是检查外部 GPIO 的电平。

- 如果 GPIO 引脚电平较低，则目标板将进入引导模式以运行 Boot Loader。
- 如果 GPIO 引脚电平较高，则目标板将进入用户代码模式以运行应用代码。

但是，某些应用的可用引脚/接线数目有限，并且没有针对这些操作提供额外的 GPIO。 对于这种情况，可以使用握手命令来确定工作模式。

- 如果发生超时并且握手失败，则目标板将进入用户模式。
- 如果握手成功，目标板将进入 Boot Loader 模式。

下图给出了用于检查工作模式的流程图。

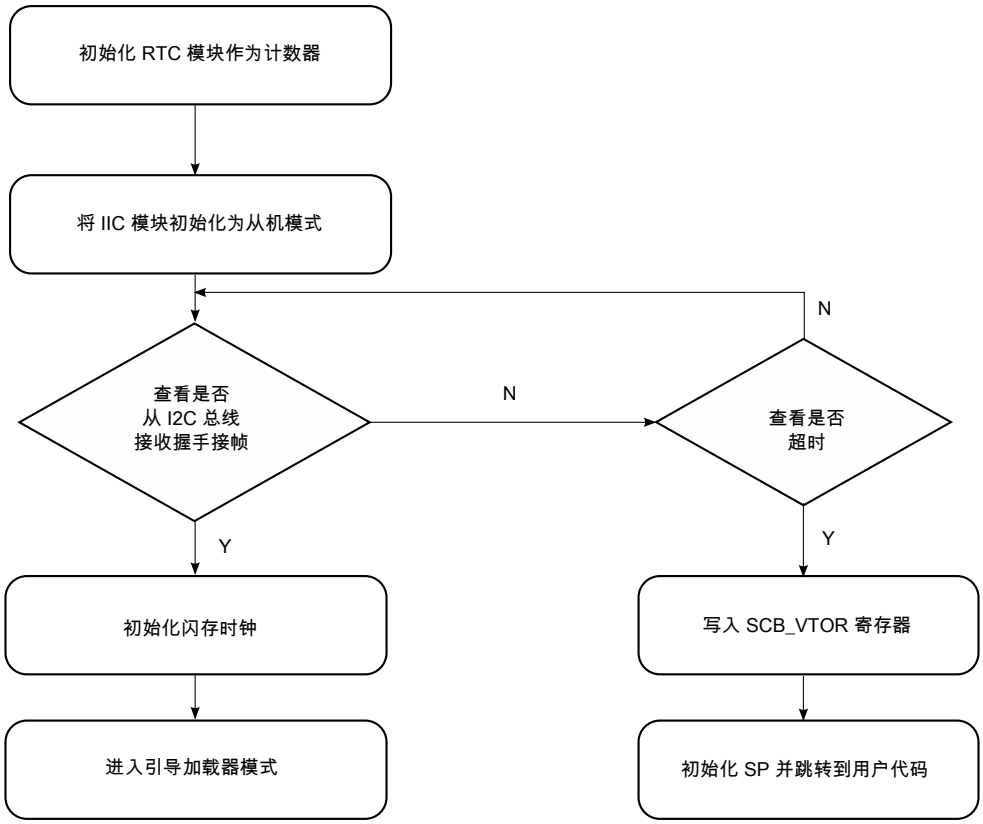

图 3. 用于检查工作模式的流程图

3.2.1 IIC 从属驱动器

目标板将 IIC 配置为从机模式。 使用中断方式在 I2C 中断服务程序中和主机通信。 有关详细的中断流程，请参见以下网站上提供的 KE02Z64P20SF0RM： freescale.com。 以下是示例代码段：

```
 void I2C_SlaveCallback( void )
{
I2C_ClearStatus(I2C0,I2C_S_IICIF_MASK);
if( I2C_GetStatus(I2C0) & I2C_S_ARBL_MASK )
{
I2C_ClearStatus(I2C0,I2C_S_ARBL_MASK);
if( !(I2C_GetStatus(I2C0) & I2C_S_IAAS_MASK) )
{
// IIAAS is 0
return;
}
}
if( I2C_GetStatus(I2C0) & I2C_S_IAAS_MASK )
{
I2C_SendAck(I2C0);
gbI2CRecFrameFlag = 0;
if( I2C_GetStatus(I2C0)& I2C_S_SRW_MASK )
{
// slave send data
I2C_TxEnable(I2C0);
u8SendIndex = 0;
I2C_WriteDataReg(I2C0,u8SendBuff[u8SendIndex++]);
}
```

```
else
{
I2C_RxEnable(I2C0);
I2C_ReadDataReg(I2C0);
u8RecIndex = 0;
}
}
else
{
if( I2C0->S & I2C_S_SRW_MASK )
{
// if require ACK from master
if( I2C0->S & I2C_S_RXAK_MASK )
{
// no receive the ACK, switch to RX
I2C_RxEnable(I2C0);
I2C_ReadDataReg(I2C0);
}
else
{
if( u8SendIndex < I2C_TX_BUFF_LENGTH )
{
I2C_WriteDataReg(I2C0,u8SendBuff[u8SendIndex++]);
}
else
{
/* here do nothing, clock stretching or send a 0xff to master. */
I2C_WriteDataReg( I2C0, 0xff );
}
}
}
else
{
if( u8RecIndex < I2C_RX_BUFF_LENGTH )
{
u8RecBuff[u8RecIndex++] = I2C_ReadDataReg(I2C0);
if( u8RecIndex > sizeof(uint32_t) )
{
pRxFrameLength = (uint32_t *)&u8RecBuff[0];
if( u8RecIndex >= (*pRxFrameLength) )
{
// receive a frame data from master
gbI2CRecFrameFlag = 1;
Memcpy_Byte((uint8_t *)&gu8I2CRxFrameBuff[0],
(uint8_t *)&u8RecBuff[0],u8RecIndex);
// reset index counter
u8RecIndex = 0;
// change MCU state to BUSY
u8SendBuff[0] = SLAVE_MCU_STATE_BUSY;
}
}
}
}
}
}
```

IIC 在收到数据帧后将设置标志 (g_bIICRecFrameFlag)，以便应用程序代码能够进一步处理该数据帧。

3.2.2 指令说明

引导程序会一直检查标志（g_bIICRecFrameFlag）。 当标志（g_bIICRecFrameFlag）为 1 时，Boot Loader 开始处理收到的帧。 并且会使用校验和来验证收到的帧是否正确，验证后，解包并处理相应的指令。 以下是收到的帧的格式。

总数据长度（4 字节）	指令（1 字节）	地址（4 字节）	数据长度（1 字节）	数据	校验和（1 字节）

下表提供了指令列表。

指令功能	指令	肯定应答	否定应答
握手	0x02	0x82、0xFC	0x82、0x03
标识	0x49	0xC9，标识信息	0xC9、0x03
擦除扇区	0x45	0xC5、0xFC	0xC5、0x03
写入	0x57	0xD7、0xFC	0xD7、0x03
读取	0x52	0xD2，数据	0xD2、0x03
退出	0x51	无应答	无应答

- **握手指令**

  下面提供了握手指令（编码为 0x02）收到的数据包。

总数据长度（4 字节）	指令（1 字节）	地址（4 字节）	数据长度（1 字节）	数据	校验和（1 字节）
6	0x02	-	-	-	CS

  应答指令如下：

指令（1 字节）	数据
0x82	0xFC/0x03

  - 如果接收状态为 0xFC，则表示目标板正在 Boot Loader 模式下工作，并已准备好与转接板通信。
  - 如果接收状态为 0x03，则表示目标板处于用户模式，且无法接收其他指令。

- **标识指令**

  收到的标识指令（编码为 0x49）如下：

总数据长度（4 字节）	指令（1 字节）	地址（4 字节）	数据长度（1 字节）	数据	校验和（1 字节）
6	0x49	-	-	-	CS

  需要提供的 MCU 信息如下：

  - 协议版本 - 1 字节
  - 系统设备标识寄存器（SDID）内容，r（13 - 16 位）是反映当前芯片修订版号 - 2 字节
  - 可重新编程的 flash 扇区数 - 4 字节
  - 可编程区域的起始地址 - 4 字节
  - 可编程内存区域的结束地址 - 4 字节
  - 原始向量表的地址（1KB） - 4 字节
  - 新向量表的地址（1KB）- 4 字节
  - MCU 擦除块的长度 - 4 字节
  - MCU 写入块的长度 - 4 字节
  - 标识符信息，以零结尾的字符串 - n 字节

  下面代码是标识信息的结构体。

```
typedef uint32_t addrtype;
typedef struct
{
```

```
    unsigned char Reserve ; // reserve bytes for 4 bytes align
    unsigned char Version; /** version */
    uint16_t Sdid; /** Sd Id */
    addrtype BlocksCnt; /** count of flash blocks */
    addrtype FlashStartAddress; /** flash blocks descriptor */
    addrtype FlashEndAddress;
    addrtype RelocatedVectors; /** Relocated interrupts vector table */
    addrtype InterruptsVectors; /** Interrupts vector table */
    addrtype EraseBlockSize; /** Erase Block Size */
    addrtype WriteBlockSize; /** Write Block Size */
    char IdString[ID_STRING_MAX]; /** Id string */
   }FC_IDENT_INFO;
```

对应的应答指令如下：

指令（1 字节）	数据
0xC9	标识信息

- 擦除指令

  收到的擦除指令（编码为 0x45）的数据包如下：

总数据长度（4 字节）	指令（1 字节）	地址（4 字节）	数据长度（1 字节）	数据	校验和（1 字节）
10	0x45	地址	-	-	CS

  应答指令如下：

指令（1 字节）	状态
0xC5	0xFC/0x03

- 写入指令

  收到的写入指令（编码为 0x57）的数据包如下：

总数据长度（4 字节）	指令（1 字节）	地址（4 字节）	数据长度（1 字节）	数据	地址（1 字节）
总长度	0x57	地址	-	-	CS

  对应的应答指令如下：

指令（1 字节）	状态
0xD7	0xFC/0x03

- 读取指令

  收到的读取指令（编码为 0x52）如下：

总数据长度（4 字节）	指令（1 字节）	地址（4 字节）	数据长度（1 字节）	数据	校验和（1 字节）
11	0x52	地址	要读取的数据长度	-	CS

  对应的应答指令如下：

指令（1 字节）	数据
0xD2	数据

- 退出指令

  该指令不需要任何应答。

  收到该指令后，需要修改标志并跳转到新中断向量表的起始地址。

4 内存分配

Boot Loader 代码占用 flash 的第一个扇区（最低内存地址空间）。 参见下图。 这种放置方式需要转移 flash 空间的开始位置，因此用户需要修改链接文件（IAR 中的 ICF 文件，CodeWarrior 中的 LCF 文件）中的地址定义。 以下是修改 ICF 链接器文件的示例：

Kinetis E KE02Z

以下代码段提供了在 IAR6.5 中修改 ICF 文件的示例。

```
// default linker file

define symbol __ICFEDIT_region_ROM_start__ = 0x00;

// modified Linker file for KE02Z 64k flash

define symbol __ICFEDIT_region_ROM_start__ = 0x1000;
```

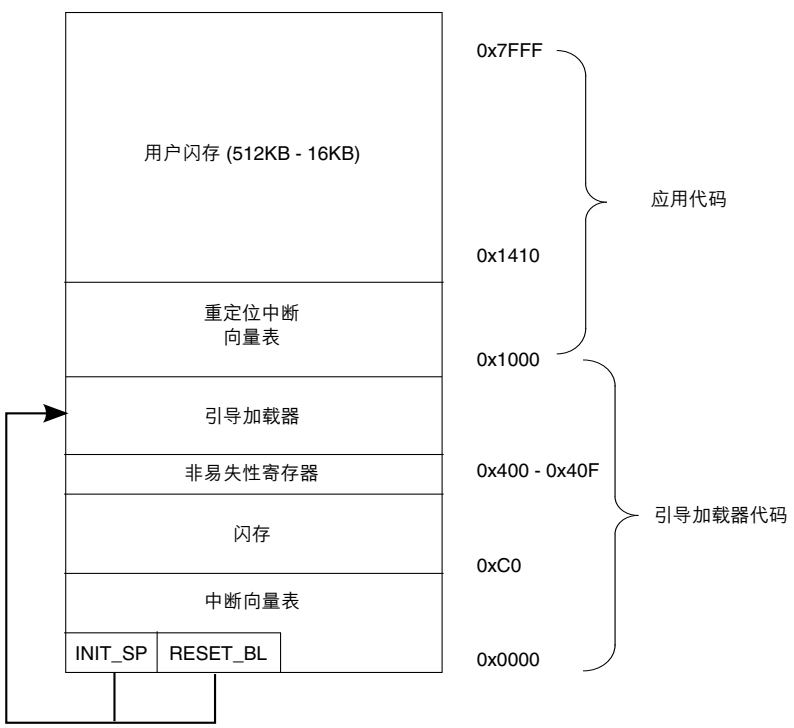

图 4. 内存分配

5 结论

本文档介绍如何使用转换板作为转接桥，使用另一块板作为目标板 (KE02Z-FRDM)，在 Kinetis E 系列 MCU 上实现 IIC Boot Loader。 用户也可以在应用软件中自行添加 Boot Loader。

6 参考

下列的参考文档可从 **freescale.com 上获得**

- 《KE02Z64M20SF0RM：KE02 子系列参考手册》
- 《AN2295：适用于 M68HC08 和 HCS08 MCU 的开发者串行 Boot Loader》

7 术语表

UART	通用异步接收器/发送器
IIC	内置集成电路
FCCOB	Flash 通用命令对象
WDOG	看门狗
MCG	多用途时钟发生器

8 修订历史记录

修订版本号	日期	重要改动
0	07/2013	最初公开版本

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

本文档中的信息仅供系统和软件实施方 使用飞思卡尔产品。 未包含基于本文档信息设计或加工 任何集成电路的任何明确或隐含的版权许可授权。飞思卡尔保留对此处任何产品进行更改的权利， 如有更改，恕不另行通知。

飞思卡尔对其产品在任何特定用途方面的适用性不做任何担保、声明或保证，也不承担因为应用 或使用产品或电路所产生的任何责任， 明确拒绝承担包括但不局限于因果性或附带损害在内的所有责任。 飞思卡尔 数据表和/或规格中所提供的“典型”参数在不同应用中可能 并且确实不同，实际性能会随时间而有所变化。 所有工作参数， 包括“典型值”在内，在每个 客户应用中必须经由客户的技术专家进行验证。 飞思卡尔未转让与其专利权及其他 权利相关的许可。 飞思卡尔销售产品时 遵循以下网址中包含的标准销售条款和条件： freescale.com/SalesTermsandConditions 。

Freescale, Freescale logo, and Kinetis are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. All other product or service names are the property of their respective owners.

© 2013 飞思卡尔半导体有限公司

Document Number AN4775
Revision 0, 07/2013