AN012 — C8051F0xx 引导装入程序考虑及举例

相关器件

本应用笔记适用于下列器件：

C8051F000、C8051F001、C8051F002、C8051F005、C8051F006、C8051F010、C8051F011、C8051F012、C8051F015、C8051F016 和 C8051F017。

引言

本文介绍对 C8051F0xx 系列器件引导装入程序的一些考虑及使用方法。引导装入程序提供在系统复位或接收到命令后对程序存储器（FLASH）进行**在系统**重新编程的能力。本文讨论在实现引导装入程序时的一些考虑并给出一个引导装入程序的例子。

引导装入程序的操作

在器件复位后，一个引导装入程序将从一个指定的源（主机）下载程序代码。在复位时，引导装入程序会收到一个 *引导装入允许* 信号，将器件配置为能接收代码并将代码数据下载到存储器中（对于 C8051F0xx 器件，存储器为 FLASH）。在下载成功后，引导装入程序会转去执行新程序。C8051F0xx 器件的引导装入程序可以有很多形式，但在允许引导装入程序方面大都遵循同样的基本程序：

1. 配置用于下载数据的外设和输入/输出端口引脚（例如，SPI、SMBus、UART 等）。
2. 擦除用于接收下载数据的存储区。
3. 向主机发送一个准备好信号表明它已准备好接收数据。
4. 接收下载数据并存入存储器（这一步可能包含错误控制或传输协议）。
5. 跳转到已下载的程序的入口点并开始执行程序。

硬件考虑

引导装入程序需要在一个主机与 C8051F0xx 通信外设之间建立通信连接，还需要有一个通知器件启动引导装入程序的手段。

引脚分配和数字交叉开关

C8051F0xx 使用数字交叉开关为数字外设分配用于外设接口的端口引脚。（请见应用笔记“AN001—配置端口 I/O 交叉开关译码器”。）交叉开关允许使用数字外设的任意组合，但用户必须考虑到软件能够改变器件的引脚分配。

Silicon Laboratories Inc.
4635 Boston Lane
Austin, TX 78735
Email: mcuinfo@silabs.com
Internet: www.silabs.com

新华龙电子有限公司
深圳市福田区华强北路现代之窗大厦 A 座 13F C 室(518013)
电话：0755-83645240 83645242 83645244 83645251
电邮：shenzhen@xhl.com.cn
网址：www.xhl.com.cn

在大多数情况下，引导装入程序会使用与最终应用相同的引脚分配。

引导装入允许

在复位或其它条件要求**在系统**编程时，器件必须有一个输入用于通知开始下载过程。这可以通过读取一个作为引导装入信号的通用 I/O 引脚来完成。一旦确定了某一应用的引脚分配，应该安全地选择用哪一个引脚作为引导装入允许信号。这样可使主机或其它硬件能通知 C8051F0xx 开始装入过程。

在本应用笔记提供的例子中，P1.4 是引导装入允许信号，在复位的最后阶段被采样。当端口引脚 P1.4 保持低电平时，启动引导装入过程。注意，端口引脚在复位后的缺省状态为高电平（逻辑‘1’），因此硬件复位后通用 I/O 引脚的输入信号应为逻辑‘0’才能通知启动引导装入操作。

软件考虑

引导装入软件实现图 1 中所示程序流程的基本功能。当允许引导装入时，引导装入程序必须使器件准备好接收数据。首先，引导装入程序对所需要的通信外设进行配置。然后引导装入程序必须对用于下载的存储器进行擦除并允许对存储器写入。为了建立通信链路，引导装入程序可以通过自动波特率检测确定位速率。另外，主机和 MCU 器件还可以使用预定的波特率。一旦器件已准备好接收数据，应通知主机。主机接到通知后发送数据，在有用数据前可能还会加上有关下载的信息（例如主机将要发送的字节数）。

本文所提供的例子是一个没有错误控制的简单引导装入程序，数据被下载到连续的存储器空间。更复杂的引导装入程序可能会使用通信协议，数据也可能被下载到跨越多个扇区且不连续的存储器地址空间。

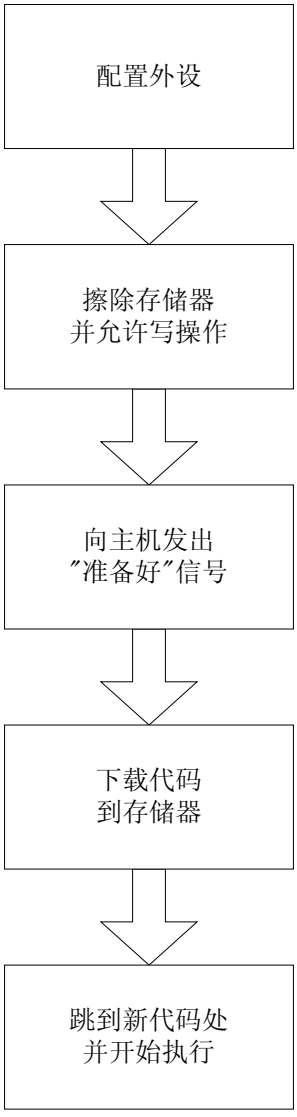

图1. 引导装入流程图

自动波特率检测

引导装入程序可以用自动波特率检测法确定主机的传输速率。例如，主机可以发送一个训练字节（例如，0x55），C8051F0xx 用该字节确定 UART 的波特率。另一个方案是，主机可以用预定的波特率发送一个字节来设置下装过程的传输速率。此后，器件就可以配置 UART 和定时器，以便在接收数据前能工作在正确的波特率。在所提供的例子中，我们没有使用自动波特率检测。我们假定采用 115.2k 位/秒的预定波特率和 18.432MHz 的系统时钟。

FLASH 存储器写入

在引导装入过程中，程序代码被写入到 FLASH。在对 FLASH 存储器进行擦除和写入操作时有一些特别需要注意的事项。请参见应用笔记：“AN009 – 从应用程序写 FLASH”。一般来说，引导装入程序会擦除一个或多个 512 字节的 FLASH 页。一旦 FLASH 被允许写入，目标板就已经准备好接收下装数据。注意，对于擦除操作写允许（PSWE）和擦除允许（PSEE）位必须被置‘1’。一旦擦除操作被允许，向一个 FLASH 存储器页内写入任一字节将擦除整个 512 字节的 FLASH 页。

使用 FLASH 时的一个限制是执行写操作的速度。完成写一个字节的操作需要最长 40 µs 的时间。因此主机的传输速率不能高于 25 k 字节/秒，或 250 kbps（在 UART 8 – N – 1 方式）。在使用 FLASH 时必须考虑到这一点，因为某些外设可能以高于此速率的速度传输数据。

本应用笔记中的引导装入程序使用 UART。最大的通用 UART 位率是 115.2 kbps，或 11.52k 字节/秒。这比写入 FLASH 所要求的传输速度限制值要小得多。

注意，在 FLASH 擦除和写操作期间 CPU 内核停止工作。外设部件如定时器和 UART 继续正常工作。在 CPU 停止工作期间发生的任何中断都将被保持并在写/擦操作完成后得到服务。

引导装入程序示例

本文提供的例子是一个没有通信协议的简单引导装入程序，可以在一个连续的地址空间内写入最多 512 字节（FLASH 存储器的一页）。本例中的引导装入程序在函数 *Main()*中完成对系统时钟和端口引脚的设置，并测试端口引脚 P1.4 是否为逻辑‘0’。如果 P1.4 为低电平，程序转到 *bootload()* 函数对 FLASH（擦除和写允许）和 UART 进行设置。该函数接着发送一个字节，以通知主机 MCU 已准备好接收下装数据。所接收到的头两个字节是将要下载的字节数，赋给变量 *NumBytes*。接着就是将数据下载到 FLASH 存储器中，最后引导装入程序转去执行刚刚下载的程序。

配置

本文的示例引导装入程序使用 UART 和定时器 1，工作波特率为 115.2 kbps。系统时钟源自 18.432 MHz 的外部晶体。本例所进行的配置过程如下：

1. **配置系统时钟源：**特殊功能寄存器 OSCXCN 和 OSCICN 被设置为使用外部 18.432 MHz 晶体作为系统时钟源。程序等到外部晶体有效（XTLVLD）标志置‘1’后将系统时钟切换到外部时钟源。
2. **配置 I/O 端口引脚：**该引导装入程序使用 UART 并假定最终应用中不使用 SPI 和 SMBus。通过设置 XBR0 来分配 UART 信号所用的端口引脚。通过将 PRT0CF 寄存器中的对应位置‘1’

将外设输出配置为推挽方式。其它端口引脚仍保持漏极开路的缺省设置。最后将 XBARE（XBR2.0）置‘1’，允许交叉开关工作。

在 *bootload()*函数中进行下面的配置过程：

3. **配置定时器 1 为波特率发生器：**为了得到正确的波特率，必须对 CKCON、TH1、TMOD、TCON 和 PCON 进行设置。
4. **配置 UART：**设置 SCON 以选择方式 1（8 位，可变速率）并允许 UART。
5. **配置 FLASH 预分频：**为了使用 FLASH 存储器，必须根据系统时钟频率配置 FLSCL 寄存器。

该引导装入程序使用 P1.4 作为引导装入允许信号。程序测试 P1.4 是否为逻辑‘0’。如果采样到 P1.4 为低电平，程序转去执行 *bootload()*函数。在完成前面所述的配置工作后，*bootload()*函数进行下述操作：

1. **擦除用于保存下装代码的存储器地址所在的 FLASH 扇区：**FLASH 擦除前要先将 PSCTL 寄存器中的 PSWE 和 PSEE 位置‘1’。一旦向该 FLASH 扇区的任一地址写入一个字节，整个页就被擦除。擦除操作完成后，PSEE 和 PSWE 被清除以禁止写和擦除操作。在示例代码的开始处定义了一个常数 *DWNLD_SECTOR* 作为下载地址。
2. **指示器件已处于引导装入方式并已准备好接收数据：**在本例中我们向主机发送字节 0x5A 以表明引导装入程序已准备好接收数据。
3. **指示要接收的字节数：**在本例中，最初收到的两个字节表示将要下载的字节数。因为我们最多只使用一个 FLASH 页，所以字节数必须小于或等于 512。
4. **下装代码到 FLASH：**将 PSWE 位置‘1’以允许写操作并用一个 *for* 循环将下载字节写入到连续的存储器空间。一旦循环计数器等于头两个字节所指定的字节数，则下装任务完成，函数退出 *for* 循环。下载完成后，清除 PSWE（PSWE=‘0’，FLSCL=0x8F），禁止写和擦除操作。
5. **执行下装的代码：**最后一步是执行刚下载的代码。本例中假定下装代码的入口点位于 *DWNLD_SECTOR*，这也是第一个下装字节的地址。在‘C’代码中，函数指针（**boot*）用于将程序计数器重新定向到刚下载的代码。

下页提供了上述引导装入程序的源代码。请注意这只是一个简单的例子。有很多不同的技术可以用来实现 C8051F0xx 系列器件的引导装入程序。

示例源代码

```
/*****************************************************************************/
   Copyright 2001, CYGNAL INTEGRATED PRODUCTS, INC.

   文件名      : bootloader.c
   目标 MCU    : C8051F0xx
   说明        : 引导装入程序和测试驱动程序
   作者        : RS


   注 :
   (1) 该程序适用 Keil C51 v6.01 编译器和链接器


*****************************************************************************/

// 编译器配置

#include <c8051F000.h>   // 包含 C8051F000 寄存器定义

#define LOGIC_LOW    0x00
#define LOGIC_HIGH    0x01
#define DWNLD_SECTOR 0x1000
#define timer_delay_const (-18432000/1000) // 要装入到定时器 1 的数值

// 函数原型
void boot_load ( void );

// 变量
sbit BLE_PIN = P1^4;

//////////////////////////////////////////////////////////////////////////
//  主程序代码
//////////////////////////////////////////////////////////////////////////

void main (void)
{
   int i;

   // 禁止看门狗定时器
   WDTCN = 0xDE;
   WDTCN = 0xAD;

   // 配置系统时钟源
   OSCXCN = 0x66;       // 选择外部振荡器(18.432MHz)作为系统时钟并
                        // 设置频率控制位。

   // 由于在 XTLVLD 位中可能存在偶然的正脉冲干扰，我们使用定时器 1 延时 1 ms。
   CKCON = 0x10;                        // T1 使用不分频的 sysclk
   TMOD = ((TMOD & 0x0F) | 0x10);       // 配置 T1 为 16 位方式
```

```
    TH1 = (timer_delay_const >> 8);   // 装入延时常数
    TL1 = timer_delay_const;
    TR1 = 1;                          // 启动定时器
    while (!TF1);                     // 等待溢出
    TR1 = 0;                          // 停止定时器
    TF1 = 0;                          // 清除溢出标志
    while (!(OSCXCN & 0x80));         // 等待晶体有效标志(振荡器稳定运行)

    // 配置端口 I/O
    OSCICN = 0x08;                    // 切换到外部时钟
    PRT0CF = 0x7F;                    // RX (P0.7) 为输入，其它为推挽输出
    XBR0 = 0x07;                      // 允许 UART、SPI、SMBUS
    XBR2 = 0x40;                      // 允许交叉开关

    if ( BLE_PIN == LOGIC_LOW )
    {
       boot_load();
    }

    while(1);


}

//////////////////////////////////////////////////////////////////////////
//
// boot_load()
//
// boot_load()将定时器 1 配置为波特率发生器，擦除用于保存下载代码的 FLASH 扇区，
// 然后发送 0x5A 表示已准备好接收下载代码。
// 然后等待 UART 输入。所接收到的头两个字节为要下载到 FLASH 中的字节数(0x00-0x200)。
// 接下来的字节被写到 FLASH，从地址 DWNLD_SECTOR 开始的连续地址。一旦下载完毕，
// 转去执行刚下载的代码。在写入和执行之前，没有对下载代码进行错误检测。
//
//////////////////////////////////////////////////////////////////////////
```

```
void boot_load ( void )
{
    void (*boot)( void);        // 用于转向下载代码的函数指针。
    int i, NumBytes;
    char xdata *address;
    address = DWNLD_SECTOR;

    // 将定时器 1 配置为波特率发生。
    TCON = 0x00;
    CKCON |= 0x10;
    TH1 = (-10);
    TMOD = ((TMOD & 0x0F) | 0x20);    // 设置定时器 1 为 8 位自动重装载方式
    TR1 = 1;                          // 启动定时器 1

    // 配置 UART(8-N-1)
    PCON |= 0x80;               // 置位 SMOD 得到 115.2KHz 的加倍波特率
    SCON = 0x50;                // 设置 UART 为方式 1 并允许 UART


    // 设置 FLASH 预分频器，18.432MHz 时钟
    FLSCL = ((FLSCL & 0xF0) | 0x09);

    // 擦除包含地址 DWNLD_SECTOR 的 FLASH 扇区
    PSCTL = 0x03;               // 允许写(PSWE)扇区擦除(PSEE)
    *address = 0x00;            // 向扇区空写，启动擦除操作
    PSCTL = 0x00;               // 禁止写和扇区擦除

    // 指示我们已处于引导装入方式并准备接收下载数据。
    TI = 0;
    SBUF = 0x5A;                // 表示准备好的字节
    while (!TI);                // 等待字接发送完
    TI = 0;                     // 用软件清除标志


    // 接下来读到的两个字节为要下载的字节数(0x00 to 0x200)。
    // 第一个读到的字符为 MSB，下一个为 LSB。
    while (!RI);                // 等待接收
    RI = 0;                     // 清除标志
    i = SBUF;                   // 保存第一个（高）字节
    while (!RI);                // 等待第二个字节
    RI = 0;                     // 清除标志
    NumBytes=(i<<8) | SBUF;     // 移动低字节
    NumBytes = (NumBytes<=0x200) ? NumBytes: 0x200;// 检查字节数是否< 512

    // 下载代码到 FLASH
    PSCTL = 0x01;                   // 允许写操作
    for( i = 0; i < NumBytes; i++)
    {
        while (!RI);                // 等待接收下一字节
        RI = 0;                     // 清除标志
```

```
        *address++ = SBUF;          // 写到 FLASH

    }
    PSCTL = 0x00;                   // 禁止写

    // 执行刚下载的代码。
    boot = DWNLD_SECTOR;
    (*boot)();

    // 我们永远不应执行到这个位置，只是以防万一。
    return;
}
```