Freescale HCS12 系列 MCU 的 Flash 存储器在线编程方法

Method of In – Circuit Program about Flash Memory of Freescale HCS12 MCU

聂章龙　张　静　(常州信息职业技术学院 计算机系 江苏 常州 213164)

摘　要： 文章简要阐述了 Freescale HCS12 系列单片机的 Flash 存储器的编程模式,然后介绍了 HCS12 单片机的基本存储空间分配及扩展机制,重点阐述了页面 Flash ROM 实现的技术要点。随后给出 HCS12 系列单片机的 Flash 存储器编程方法的要点,并给出了具体的在线编程的 MCU 方和 PC 方程序的编程步骤及流程图,对其中的技术难点进行了深入分析。

关键词： HCS12 系列单片机 Flash 存储器 扩展机制 页面 Flash ROM 在线编程

1　引言

近几年 Flash 存储器技术趋于成熟,具有电可擦除、无需后备电源来保护数据、可在线编程、存储密度高、低功耗、低成本等特点[1]。而这些特点,正是 MCU 所期望的。目前,许多 MCU 内部都集成了 Flash 存储器,具有在线编程(In – Circuit Program)功能,允许单片机内部运行程序去改写 Flash 存储器的内容,利用这个特点,不仅可以在运行过程中修改某些参数,也为研制新型嵌入式应用开发工具提供了技术基础。Freescale 公司最近推出了新一代 16 位嵌入式单片机 HCS12 系列(下文简称为 S12 系列),其片内集成 Flash 存储器,具有单一电源电压供电、支持在线编程等特点。本文设计的在线编程方法主要是针对 Freescale S12 系列 MCU,实现对该系列空白芯片的程序下载。今后,嵌入式应用软件趋向于大型化,复杂化,代码量较大,因此,本文给出了 Flash 在 64KB 以上的单片机上如何采用页面 Flash ROM 的寻址技术来增加存储容量,以适应嵌入式应用软件发展的需求。

2　S12 系列单片机 Flash 存储器编程模式

Flash 存储器的在线编程方法与其编程模式是紧密相关的。S12 系列单片机的编程模式有两种:监控模块和用户模式。

本文只讨论用户模式下 S12 系列单片机 Flash 存储器的编程方法。S12 系列单片机的 Flash 存储器工作于用户模式时,在单片机正常工作的过程中,程序可以随时转入对 Flash 存储器进行编程操作。这种情况下对 Flash 存储器的擦除与写入,需要 PC 方通过 BDM 接口与目标板相连,BDM 的主控芯片(如:MC68HC908JB8)就向目标板 MCU 的 BKGD 引脚输入低电平,使 MCU 进入 BKGD 模式(特殊单片模式),在该模式下就可对目标芯片的 FLASH 进行擦写。

3　S12 系列单片机 Flash 存储器扩展机制

3.1 存储空间的扩展

S12 单片机的寻址空间允许超过 64KB,达到 128KB、256KB 或 512KB。CPU12 寻址空间的扩展是基于分页 Flash ROM 来实现的[2,3]。CPU12 内部有一个存储器页面寄存器 PPAGE,以进行页面的选择,PPAGE 是 5 位的寄存器。CPU12 规定每一页的大小固定为 16K 字节,因而 CPU12 的最大寻址空间就是 32 ×16KB,为 512KB。

下面以 MC9S12DG128 为例说明 Flash 在 64KB 以上的单片机是如何进行页面管理的。图 1 说明了如何将 64KB 寻址空间扩展到 128KB。

128KB 的 Flash 先被分成 Block0 和 Block1 两块,每块大小 64KB,每块还可以细分为 16KB 一页,共 8 页。

MC9S12DG128 的 128KB Flash 可分为 8 个 16KB 存储器,每个存储器页的页面编号为 0x38 ~0x3F 的某个值。CPU12 在 64KB 的寻址空间的 0x8000 ~0xBFFF 这一段开了一个窗口,永远只能看到页面寄存器的某一页,这里指 0x38 ~0x3D 这 6 个页面。128KB 的 Flash 中,0x3F 这一页永远定位在 0xC000 ~0xFFFF 这一段,

0x3E 这一页永远定位在 0x4000 ~ 0x7FFF 这一段。另外 6 页只能通过 0x8000 ~ 0xBFFF 这一个窗口来访问。Flash 的换页是通过向 PPAGE 寄存器(地址为 0x30)写入页面编号实现的。

3.2 页面 Flash ROM 的实现

3.2.1 线性地址转换为扩展内存

对于 16 位的 S12 系列单片机,由于寻址能力被限制在 64KB 以内,为了增加片内 Flash ROM 的容量,采用了页面 Flash ROM 的寻址技术,即当应用程序做得很大,超过了 64KB 寻址空间以后,S1 格式已经不能适应应用的要求,需要按照 S2 格式下载程序。S2 格式与 S1 格式的区别仅仅是 S2 格式使用 3B 表示线性地址,而 S1 格式使用 2B 表示线性地址。S2 格式中 3B 线性地址的第一个字节可转换为相应的页面编号(即页面寄存器 PPAGE 的值)。根据 CPU12 用户手册[4],编译器编译时,将线性地址转换为扩展内存地址的原理如下:所有 M68HC12 系列 MCU 的内存地址至少都是 16 位线性地址,当使用到扩展地址时,则要在 16 位线地址之前再附加一个 6 位地址,组成一个 22 位的线性地址,其高 8 位与 PPAGE 寄存器的值复用,即可将其转换为 PPAGE 寄存器的值(页面编号);低 14 位经过映射后可与扩展内存地址的低 14 位相对应。转换方法为:首先,将线性地址的 ADDR[21:14]位的值作为页面编号,如:线性地址为 0x0f0000,则其 ADDR[21:14]位为"00111100",其值为:0x3c,所以 PPAGE 寄存器的值应该是 0x3c;其次,线性地址的低 14 位 ADDR[13:0]可映射为扩展内存中的低 14 位地址 ADDR[13:0] +0x8000。

注:因为所有的扩展内存地址都存放于该页面内的 0x8000 ~ 0xbfff 范围内,因而要把线性地址的低 14 位加上 0x8000 才能映射到相应页面的 0x8000 ~ 0xbfff 扩展地址内。

3.2.2 S2 格式文件的生成

要生成 S2 格式机器码,则要做以下两方面的修改。

①集成开发环境中 GCC 编译器参数要修改如下:

//C 编译器编译参数

#make. exe – f . /include – mshort – m68hc12 – fno – ident – fno – common – c 语句中的" – mshort" 改为" – mlong – calls" 。

//链接器参数

#make. exe – f . /lib/m68hc12/ – mshort 语句中的" – mshort" 改为" – mlong – calls" 。

//汇编编译器编译参数

#make. exe – f . /include – m68hc12 – mshort 语句中的" – mshort" 改为" – mlong" 。

②程序中的汇编指令修改如下:

在汇编指令集内,使用 call 和 rtc 来调用和返回该类型的子程序,在调用和返回子程序时都会在堆栈中自动保存页面寄存器 PPAGE。

3.2.3 编写生成 S2 格式的 Linker. ld 文件

一般 GCC 编译器在编译过程中会将指定的代码直接放入 Linker. ld 文件中的. bank 段中,下面介绍针对 MC9S12DG128 的 Page Window 的脚本文件 Linker. ld 的编写,由于使用到扩展空间,因而要编译生成 S2 格式文件,使用 3 位装载地址。

通过链接器(linker),可以将浮动的 C 语言目标文件按照给定的地址空间进行链接,生成. elf 的输出文件。给定地址空间的脚本文件 linker. ld 可以实现该功能。下面介绍该链接器的一些属性和使用方法。

(1) 段在整个链接过程中,代码和数据的基本单位是"段"。用户将不同属性的内容放入不同的段中,链接器识别这些段,按照用户指定将各个段放入相应的存储单元中,完成链接。Linker. ld 文件中应该包括所需要的段,这些段的名称和属性是默认的。当然用户完全可以设定自己的段,并设置其链接地址。限于篇幅,此处不列出各个段的含义。

(2)链接文件的基本命令格式

带页面分割的链接脚本文件就与普通的链接脚本文件有所不同,Page Window 的脚本文件基本命令格式有以下两种:

①MEMORY 命令

MEMORY 命令是一种描述方式,它告诉链接器在整个可用的空间中哪些区域是可以为哪一类段使用。每个段名称后面的括号里是该段存储单元的属性标识,r:可读,w:可写,x:可执行。以下程序段可说明该命令的功能。

```
MEMORY {
/ * 用户程序存放的第一块 ROM 区 * /
bank0(rx):ORIGIN =0x0f0000,LENGTH = 0x3FFF
bank1(rx):ORIGIN =0x0f4000,LENGTH = 0x3FFF
...
bank6(rx):ORIGIN = 0x4000, LENGTH = 0x3FFF
bank7(rx):ORIGIN =0xC000, LENGTH = 0x3FFF
/ * 程序的矢量区,存放常量数据 * /
rom (r) : ORIGIN = 0xff80, LENGTH = 0x0080
/ * 程序 RAM 区,存放初始化全局和静态变量 * /
```

```
    data(rwx):ORIGIN = 0x1000, LENGTH = 0x0400
    }
    /*程序堆栈指针*/
    PROVIDE (_stack = 0x4000 -1);
```

Bank0 ~ bank7 对应的分页值为:0x38 ~ 0x3f,是用户程序可存放的区域,bank0 段中存放的用户程序从 0x388000 开始,长度是 0x3fff 个字节,即用户程序从 0x388000 处开始载入,程序代码长度被限制在 0x3fff 个字节内。bank6 段中存放的用户程序实际是从 0x4000 开始,长度是 0x3fff 个字节,bank7 段中存放的用户程序实际是从 0x8000 开始,长度是 0x3fff 个字节。因为 bank6 和 bank7 是始终分别对应 0x4000 ~ 0x7fff 和 0xc000 ~ 0xffff 这两段。

Rom 段是只读数据段,用来存放一些不会修改的常量数据。

Data 段是从 0x1000 开始,长度是 0x0400 个字节的片内 RAM,全局变量和静态变量存放在该段,该段数据可读写,也可执行。

PROVIDE 是定义初始化时的堆栈指针值。

② SECTIONS 命令

SECTIONS 命令是脚本文件中最基本的命令,它的功能非常强大。下面也通过一个程序段来说明其用法,要求程序中的代码段放在 0x388000 起始地址处,数据段放在 0x1000 起始地址处,未初始化数据段放在数据段之后,向量表段放在 0xFF80 起始地址处。

首先,SECTIONS 将当前指针设为 0x1000,随着程序运行,当指针指到 bank0 段时,在该段中指定了用户程序的目标文件 main.o、isr.o、setup.o 放在该段从 388000 开始处的 ROM 区。若用户程序的目标文件没有指定在任何一个代码段,则系统默认在 0x4000 起始处装载。

当指针指向向量表段时,向量表目标文件 vectors.o 放在该段的 0x FF80 起始处。

```
SECTIONS {
    .bss 0x1000: {
        *(.bss)
    } > data
    bank0 0x0f0000: {
        *(.init)
 *(. bank0)
. /OBJ/main.o
. /OBJ/isr.o
. /OBJ/setup.o
    } > bank0
...
    vectors 0xff80: {
        ./OBJ/vectors.o(*.rodata)
    } > rom
}
```

3.2.4 设置 PPAGE 和 FCNFG 寄存器

在编写 Flash 存储器的擦写程序时,要先根据 S2 格式文件装载地址的第一个字节的内容,来决定对 PPAGE 赋相应的页面值,然后才能完成 Flash 存储器的擦写操作。

4 S12 系列单片机 Flash 存储器在线编程方法

Flash 存储器一般作为程序存储器使用,不能在运行时随时擦除、写入。S12 的 Flash 存储器提供了用户模式(BDM 模式)下的在线编程功能,对 RAM 的读写,需要专门的过程,下面将介绍 S12 的 Flash 存储器的基本操作方法。

4.1 FLASH 的擦除与写入操作步骤如下[5]:

① 清除 Flash 状态寄存器 FSTAT 中的出错标志位 ACCERR 和 PVIOL。

②写 Flash 配置寄存器 FCNFG 中的 D1 和 D0 位。这两位表示选择 S12 系列单片机的 Flash 中的哪一个 64KB 块。

③写 PPAGE 寄存器。如果要擦除或写入的 Flash 是存储空间窗口 0x8000 ~ 0xBFFF 中的某一页,这一步是必须的。

④检查前一次 Flash 处理的命令是否执行完毕,能不能写入新的命令,这是通过 Flash 状态寄存器 FSTAT 中的命令缓冲区的标志位 CBEIF 是否为 1,即命令缓冲区是否可以使用来实现的。若不能使用,则等待,直到可以使用。

⑤把要写入的数据字写到相应的地址中。

⑥向 Flash 命令寄存器 FCMD 中写命令字。

⑦向 Flash 状态寄存器 FSTAT 中命令缓冲区的标志位 CBEIF 写 1,以清除 Flash 状态寄存器 FSTAT 中的命令缓冲区的标志位 CBEIF,这时状态寄存器中的 CCIF 位将置位,说明操作成功。

4.2 Flash 擦写程序的流程图

由于在 Flash 的擦写过程中,Flash 是不能读的,所以擦除和写入 Flash 的程序要放在 RAM 中,即在 Flash 的擦除或写入之前,要把擦除或写入的可执行代码复制到 RAM 中去,并让程序在 RAM 中执行。

但擦除或写入程序的编译是在 PC 机方完成，并保存在磁盘上，然后再通过下载工具（如：BDM 调试器）下载到 RAM 中执行。然后通过调用执行 TBDML 动态链接库中的函数[6]：tbdml_write_reg_pc（Address）和 tbdml_target_go（），若 tbdml_target_go（）返回值为"0"，则说明擦除或写入程序获得正确执行。

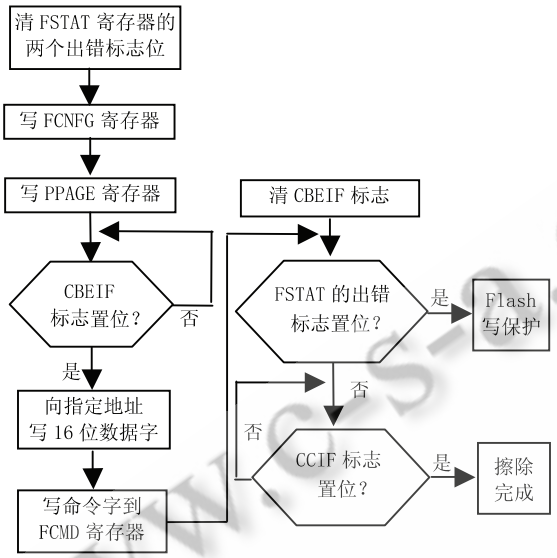

图 2　MCU 方擦写 Flash 操作的程序流程图

Flash 擦除操作方式有：块擦除和整体擦，它们的基本操作流程完全一样，如图 2 所示，唯一区别是在写命令字到 FCMD 寄存器时，若用块擦除方式，则写 FCMD 的值是：0x40；而用整体擦除方式，则写 FCMD 的值是：0x41。由于 HCS12 系列单片机的擦写速度很快，因而一般采用整体擦除方式。按照上面的流程很容易写出块擦除和整体擦除的子函数，块擦除子函数的入口参数为要擦除的块起始地址（addr），整体擦除子函数的入口参数为要擦除的 Flash 起始地址（addr），对 S12 系列单片机而言，一般为 0x4000。

Flash 写入操作一次只能写一个字（即两个字节），写一个字的操作也可按照图 2 流程进行。写一个字的命令字是：0x20。按照图 2 的流程图写一个字的子程序也很容易编写，该子函数有两个入口参数，第一个参数为数据要写入到 Flash 的地址（addr），第二个参数为要写入的数据（data）。要是写批量数据，只要循环调用写一个字的子程序即可实现。

若是擦除操作，则向指定地址写 16 位数据字的内容为：0xffff，而为写入操作时，写 16 位数据字的内容为：要写入的一个字数据 data。

4.3 擦除与写入子程序编程要点说明

由于擦除与写入操作是个复杂的过程，根据实际编程调试与项目开发过程中积累的经验，现提出以下建议：

① RAM 中要留有足够的缓冲区，以便存放复制到 RAM 中的子程序，具体值是取擦除与写入子程序中的大者即可。它们的大小可在编译后. LST 文件中查得。

② 一次擦除后未被写入过的区域可以再次调用写入子程序写入，但写入过的区域，未经擦除不能重写。

③擦除时若用的是每次擦除一页（256 字节），则应对数据进行合理安排，避免误擦。

5　PC 机方界面及程序框架

PC 机方程序采用 Visual Basic6. 0 语言编写，程序下载由 S19 文件分析模块、TBDML 通信模块和 Flash 存储器的擦除和写入模块构成。TBDML 通信模块负责通过 USB 接口将 PC 方的 S – record 代码写入到空白的 Flash 存储器的指定区域。S19 文件分析模块则负责将 S – Record 标准的 S19 文件进行分析[7]，将文件的内容转换成方便传输的格式，以及判断文件中程序的起始地址、页数、是否越界等。下载主界面如图 3 所示。

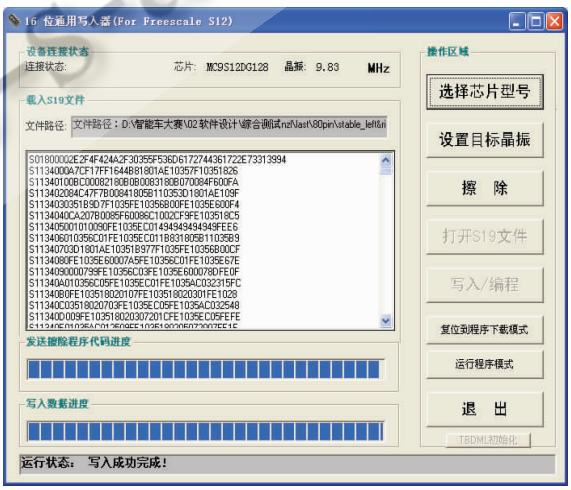

图 3　Flash 在线编程 VB 程序界面

Flash 在线编程 PC 机方程序的主要功能为：先将 Flash 擦写程序数据和用户程序数据写入到 RAM 的指

定区域(注意要事先为他们分配好固定的 RAM 区,以免出现相互重叠,从而造成数据丢失情况),当一页的用户程序数据写入到事先分配的 RAM 区后,就调用 tbdml 动态链接库函数:tbdml_write_reg_pc(address)和 tbdml_target_go()去执行 RAM 区存放的擦除和写入程序代码,若 tbdml_target_go()返回值为"0",则程序就能正确执行,从而将事先存放在 RAM 区的一页数据写入到指定的 Flash 区。

6 结束语

本文设计的 Freescale S12 系列 MCU 的 Flash 存储器在线编程方法,主要是实现把应用程序经编译后生成的 S 格式文件代码下载到空白芯片的 Flash 区。重点阐述了旨在增加片内 Flash ROM 容量的页面 Flash ROM 实现技术要点。基于这些设计思想,作者于 2006 年下半年开发了 16 位通用写入器,目前已经用于 Freescale S12 系列 MCU(如:MC9S12DG128、MC9S12NE64、MC9S12UF32 和 MC9S12xDP512 等)的教学实验和产品开发中,基本实现了预期的目标。但目前还没有完全适用于该系列的所有芯片,今后我会在现有的基础上继续不断深入研究,使该写入器功能更加完善,通用性更强。

参考文献

1 吴东坡. Flash 存储器技术与应用. 微电子学与计算机. 1998,(6):55 – 56.

2 邵贝贝. 单片机嵌入式应用的在线开发方法. 清华大学出版社,2004.

3 王宜怀,陈帅. 嵌入式微控制器 MMC2107 的 Flash 存储器扩展技术. 微电子学与计算机. 2004,21(10):105 – 109.

4 Motorola. CPU12 REFERENCE MANUAL. 2001.

5 涂时亮. MC68HC908 Flash 存储器的在线擦写方法. 单片机与嵌入式系统应用. 2001,(9):180 – 183.

6 Daniel Malík. Turbo BDM Light interface. 2005,

7 王宜怀. 嵌入式应用在线编程开发系统的研制. 计算机工程. 2002,28(12):22 – 24.