无中断向量重定位单片机中实现 IAP 和 APP 中断的方法

韩雨泓[1],陈良勇[2]

(1.四川大学 计算机学院,四川 成都 610000;2.原成都军区联勤部后勤信息中心,四川 成都 610000)

摘　要:独创了一种基于 RAM 中转的中断跳转方法,该方法以软件形式实现了单片机的中断向量重定位功能,实现了在应用中编程,克服了某类普通经济型单片机无法通过硬件寄存器设置中断跳转地址来实现跳转的局限性,使得这类单片机也能在同一片 Flash 内运行 IAP 和 APP 并相互跳转,大大拓展了实用性。采用某国产单片机(SWM240)实现了 IAP 和 APP 部分,并在生产实际中得到检验。

关键词:单片机;IAP;中断跳转;更新

中图分类号:TP273　　**文献标识码**:A　　DOI: 10.19358/j.issn.2096-5133.2020.02.010

引用格式:韩雨泓,陈良勇.无中断向量重定位单片机中实现 IAP 和 APP 中断的方法[J].信息技术与网络安全,2020,39(2):53-56.

The method of implementing IAP and APP interrupt in MCU without interrupt vector relocate function

Han Yuhong[1],Chen Liangyong[2]

(1.School of Information Science and Technology,Sichuan University,Chengdu 610000,China;
2.Chengdu Military Region Joint Logistics Department, Chengdu 610000,China)

Abstract:This article uniquely created an interrupt jump method based on RAM. This method is implemented in software. The microcontroller's in-program programming function overcomes some types of ordinary economical microcontrollers. It is not possible to set a jump address through a hardware register to implement jumps. Making this kind of Microcontroller also be able to run IAP and APP in the same flash and jump each other,which greatly expands the practicality. A domestic single-chip microcomputer is used to implement the IAP and APP parts,and has been tested in production.

Key words: microcontroller;IAP;interrupt jump;update

0　引言

在实际的生产工作中,从经济和实用性方面考虑,选择嵌入式系统 MCU 控制器时常常会使用到一些性能相对较弱,功能配置也不太强大的单片机,这类单片机有时候仅仅够用,无法从根本上满足多元化的需求。如果需要更多的功能就需要选择更高端的单片机,同时也意味着成本的增加。本文在项目过程中选用的 MCU 即为这种类型。在产品的整个生命周期中,需要对应用程序进行及时的升级换代,这就用到了 IAP 功能[1-2],而该款 MCU 不具备硬件级的中断向量重定位功能,因此只能另辟蹊径,以软件的方式解决该问题。

1　IAP 系统介绍

在传统的单片机应用开发中,程序员将写好的代码通过 IDE 编译链接后生成可执行的二进制文件下载到单片机 Flash 中即可[3],功能的修改或者错误的修正需要程序员在完成代码修改后,再次通过专用调试器下载到单片机中才能实现。在简单小批量产品功能的实现中,这已经可以满足要求。如果是已经量产的产品,要想再修改功能几乎不可能。但是面对越来越复杂的应用环境,越来越多的应用要求,一次性调试好代码并下载运行的情况已经变得越来越难,而且随着代码量的增多,隐藏的 BUG 也越不容易被发现,需要在后期的应用中不断完善。

IAP(In Application Programming)功能即是为了应对这种情况应运而生的。IAP 是一段完整的程序,它与应用程序(APP,即具体实现功能的单片机

代码)是相互独立和分开的两部分,它的功能是在产品全生命周期中随时根据需要对应用程序部分代码进行擦除和烧写,目的是为了在产品发布后可以方便地通过预留的通信接口对产品中的功能进行更新升级[4]。

IAP 和 APP 是同时存在于单片机上的两个完整的代码段。IAP 实现对 APP 段代码的擦除和重新烧写,APP 则负责实现具体的功能任务,当需要时跳转至 IAP 进行程序更新。它们分别烧写在单片机 Flash 的两个区域。对于本文所使用的华芯微特 SWM240D8U7,它拥有 64 KB 的片上 Flash(0x0——0xFFFF),就可以划分 16 KB 给 IAP(0x0——0x3FFF),余下 48 KB 留给 APP(0x4000——0xFFFF)。需要特别注意的是 IAP 代码和 APP 代码编译后的大小不能超过所分配的空间大小。

2 IAP 与 APP 程序运行流程

本文使用的 SWM240 单片机在上电后固定地从 Flash 地址 0 处开始执行代码[5],因本文将 IAP 程序放于此处,因此首先运行 IAP 程序,在 IAP 程序中通过检查预先设置的标志信息来确定 APP 代码段是否正常,若符合要求就跳转到 APP 段开始执行,否则进入等待升级烧写状态。

ARM 的 Cortex M0 单片机将代码的最初几十个字作为中断向量入口,该区域存放的是对应的中断处理程序代码所在的开始位置。中断发生时由硬件将 PC 指针拉到对应的中断入口处,取得地址后跳转到中断处理程序,完成完整的中断处理过程[6]。在 IAP 和 APP 共存的理想情况下,如果在 APP 中发生中断,中断指针应该指向 APP 入口处的中断向量;同样,如果在 IAP 中发生中断,中断指针就应该指向 IAP 入口处的中断向量。可实际情况是,Cortex M0 系列没有像其他 M3/M4/M0 + 系列核心所具备的中断矢量表重定位寄存器[7],其中断发生时会始终会指向从 0x0 开始的某入口向量处,即本文存放 IAP 程序的向量位置。那么当 APP 中发生中断时,取得的是 IAP 中的处理程序地址如图 1 所示。

3 解决思路

由于每次中断发生时指针都指向从 IAP 开始的入口地址[8],因此可以考虑将 IAP 的所有可屏蔽中断(或者只是用到的中断)程序都写成一个跳转程序,根据当前所处的位置来取得不同的跳转

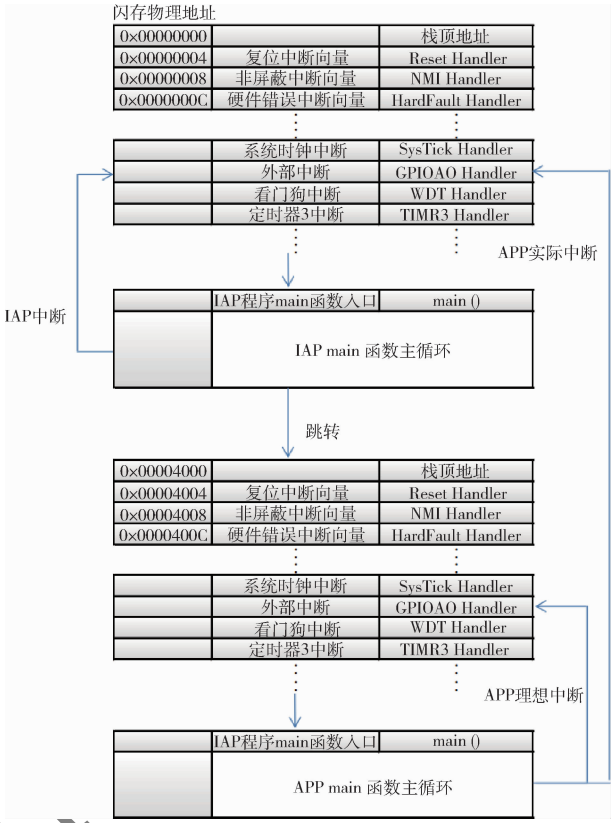

图 1 IAP 启动流程及中断流程

地址,这在单片机的启动文件中可以通过汇编语言来实现。跳转地址的取得可以有多种方法。这里通过将向量地址拷贝到 RAM 指定位置来实现。在 IAP 进程时内存存储的是 IAP 中断向量地址,当跳转到 APP 时,在 APP 中将该内存地址存储内容改写为 APP 的中断向量地址。同样当再次跳转回 IAP 时也会执行同样的操作。这样无论是在哪个运行空间,虽然中断都是跳回 IAP 向量位置,但是获取的中断地址却是与运行空间对应,如图 2 所示。

4 实现步骤

首先要实现的是将原 IAP 入口处的中断函数改写为跳转函数。在 ARM 的启动文件中定义了与硬件相对应的中断函数名称,如图 3 所示。通过修改启动文件中的函数内容,使用 LDR 和 BX 两条指令来实现取值和跳转。

用户如果在自己的代码空间中实现了该函数,并且在启动文件中 EXPORT 了该函数,中断后就会跳转到用户函数中。可以修改为自定义的跳转指令。以 GPIOA0 外部中断函数为例,图 4 显示的是未修改前在启动文件中的代码段,它的作用是当发

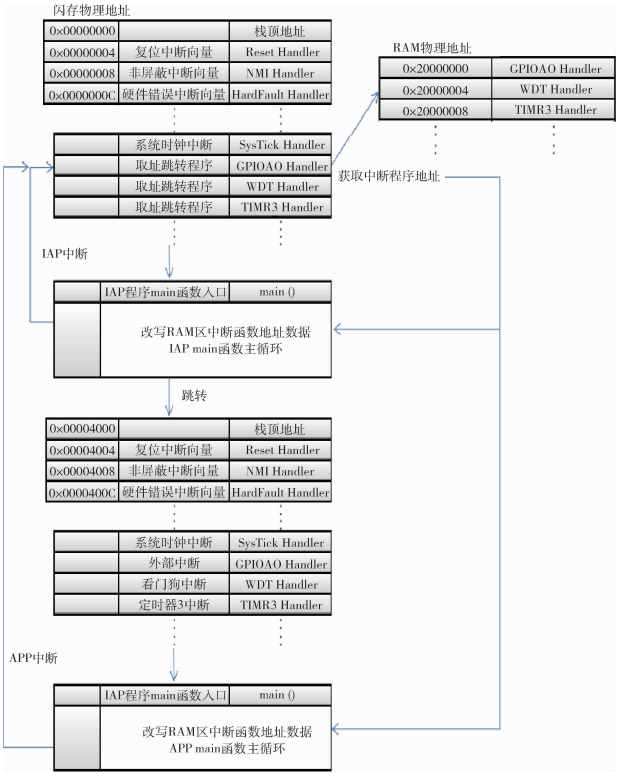

图 2　修改后的中断处理流程

```
__Vectors DCD Stack_Mem + Stack_Size; Top of Stack
          DCD Reset_Handler          ; Reset Handler
          DCD NMI_Handler            ; NMI Handler
          DCD HardFault_Handler      ; Hard Fault Handler
          DCD 0
          DCD 0
          DCD 0
          DCD 0
          DCD SVC_Handler            ; SVCall Handler
          DCD 0
          DCD 0
          DCD PendSV_Handler         ; PendSV Handler
          DCD SysTick_Handler        ; SysTick Handler
; External Interrupts
          DCD GPIOA0_Handler     ;启动文件定义的外部中断响应函数名
          DCD WDT_Handler        ;启动文件定义的外部中断响应函数名
          DCD TIMR3_Handler      ;启动文件定义的外部中断响应函数名
```

图 3　启动文件定义的中断函数名

生该中断时，使程序跳转到 GPIOA0_Handler 函数去执行（函数名其实就是一个地址）。图 5 显示的是修改以后的代码段，它的作用是当发生中断时，使程序跳转到 GPIOA0_ISR_FUN_ADD 所指的地址去执行。这里__GPIOA0_ISR_FUN_ADD 是强行指定的 RAM 地址，该地址的内容根据是在 IAP 还是在 APP 有所不同，如图 6 所示。

```
SysTick_Handler PROC
                EXPORT SysTick_Handler  [WEAK]
                B .
                ENDP
GPIOA0_Handler  PROC
                EXPORT GPIOA0_Handler  [WEAK]
                B .
                ENDP
```

图 4　修改前的中断函数

```
SysTick_Handler PROC
                EXPORT SysTick_Handler  [WEAK]
                B .
                ENDP
GPIOA0_Handler  PROC
                LDR R1, =__GPIOA0_ISR_FUN_ADD
                LDR R0, [R1]
          BX    R0
                ENDP
```

图 5　修改后的中断函数

```
; Reset Handler
__ISR_ADD                EQU    (0x20000000)
__GPIOA0_ISR_FUN_ADD     EQU    (__ISR_ADD+4*0)
__TIMR0_ISR_FUN_ADD      EQU    (__ISR_ADD+4*1)
__GPIOC0_ISR_FUN_ADD     EQU    (__ISR_ADD+4*2)
__TIMR1_ISR_FUN_ADD      EQU    (__ISR_ADD+4*3)
```

图 6　指定存储中断地址的内存位置

其次，在完成跳转函数后，需要分别在 IAP 和 APP 代码段将自己空间的中断函数地址填入对应的 RAM 位置。做法是在代码中定义全局变量，并将变量指定在 RAM 的固定位置，实现代码如下：

```
uint32 _ t  VectorTable [ 4 ]  _ _ attribute _ _ (( at (0x20000000)));
```

此处只使用了 4 个中断并将该数组固定在 RAM 起始位置。最后，需要分别在 IAP 和 APP 代码初始化阶段，将自己运行空间的处理函数地址填入对应的数组中即可。实现代码如下：

```
VectorTable[0] = (uint32_t)GPIOA0_Handler;
VectorTable[1] = (uint32_t)TIMR0_Handler;
VectorTable[2] = (uint32_t)GPIOC0_Handler;
VectorTable[3] = (uint32_t)TIMR1_Handler;
```

需要说明的是，在编写代码时，由于 IAP 和 APP 是两个独立的工程，分别都有自己的启动文件，而单品机发生中断时起作用的是 IAP 中的启动文件，因此只需要将 IAP 中的启动文件进行修改即可，

APP 中的不做改动。另外,在 IAP 和 APP 初始化过程中,文中所述获取中断处理函数地址的方法,APP 中还可以通过直接拷贝对应中断向量表中的内容来得到,但是在 IAP 中必须按照上面的方法,如果拷贝中断向量表的内容,得到的就是启动代码中 XXX_Handler PROC 的地址,这将导致程序进入死循环。

经过上面的设置后,分别将 IAP 和 APP 烧入对应的 Flash 空间,就可以实现两个空间的中断。如果用户有多个运行空间,也可以按照此方法来进行处理。

5 运行与测试

按照本文所述,在某国产 SWM240D8U7 单片机上分别建立了基于 CAN 通信的 IAP 和 APP 程序。程序上电后首先按照硬件默认设置从 0x0 处运行代码,该处代码完成 MCU 的启动,中断向量的定位,APP 应用标志的检测(检查 APP 代码段标志是否正常),如果一切正常则跳转到应用程序段执行,否则等待接收从 CAN 通信接口传送过来的升级数据包。经测试文章所述方法完全满足要求,在 IAP 和 APP 之间相互跳转的效果和基于硬件设置的单片机效果一致,达到了设计初衷。

参考文献

[1] 意法半导体公司. STM32 in-application programming (IAP) using the USART[Z]. 2010.

[2] 意法半导体公司. STM8L IAP 使用说明[Z]. 2007.

[3] 意法半导体公司. 在 STM32L011 上通过 I^2C 接口实现 IAP[Z]. 2012.

[4] 意法半导体公司. STM32Cube 以太网 IAP 示例[Z]. 2015.

[5] Synwit Technology Co., Ltd. SWM240 系列 MCU 数据手册[Z]. 2017.

[6] Arm Co. Ltd. ARM ® Architecture Reference Manual [Z]. 2010.

[7] 意法半导体公司. STM32F0x1 Reference Manual[Z]. 2012.

[8] 意法半导体公司. 关于 AN4065 中 STM32F0 IAP 升级后的外部中断不响应问题[Z]. 2011.

(收稿日期:2019-12-24)

作者简介:

韩雨泓(1983 -),男,硕士研究生,工程师,主要研究方向:信息融合与高速信号处理。

陈良勇(1977 -),男,硕士研究生,工程师,主要研究方向:嵌入式系统应用。

(上接第 44 页)

[13] Wang Jingna, Hua Guowei. Design of Android warehouse management software based on web service[J]. IOP Conference Series: Earth and Environmental Science, 2019, 252(4):1-7.

[14] Yue Huanjing, Sun Xiaoyan, Yang Jingyu, et al. Landmark image super-resolution by retrieving web images [J]. IEEE Transactions on Image Processing, 2013, 22(12): 4865-4878.

[15] 王成,李少元,郑黎晓,等. Web 前端性能优化方案与实践[J]. 计算机应用与软件,2014,31(12):89-95,147.

[16] HAO L, WAN F C, MA N, et al. Analysis of the development of WeChat mini program [J]. Journal of Physics: Conference Series, 2018, 1087(6): 062040.

(收稿日期:2019-11-05)

作者简介:

张承强(1993 -),男,硕士研究生,主要研究方向:深度学习、计算机视觉。

张永爱(1977 -),通信作者,男,博士,研究员,博士生导师,主要研究方向:信息显示技术、集成电路。E-mail:yongaizhang@fzu.edu.cn。

顾兴权(1995 -),男,硕士研究生,主要研究方向:深度学习、语音识别。

版权声明