MSP430 单片机 C 语言和汇编语言混合编程

Mixing C and Assembler With the MSP430

刘玉宏
Liu,Yuhong

摘要：为了发挥C语言和汇编语言各自的优点，二者需要相互调用函数。本文首先介绍了MSP430单片机的C语言函数的参数传递规则，然后对C语言和汇编语言的混合编程进行了详细描述，最后给出应用实例。

关键字：MSP430 单片机 IAR C 语言 汇编语言 混合编程

中图分类号：TP368.1 文献标识码：A

Abstract: In order to play the vritues of c and assembler language,they need to call each other's function. This paper describes the rules of C-Compiler for passing variables between functions, mixing c and assembler with MSP430 in details,then gives an application example.

Keyword:MSP430 MCU;IAR C-Compiler;Assembler Language;Mixing Programming

MSP430 是一款 16 位的单片机，它具有超低功耗、丰富的片内外围模块、多样的可选型号、软件对硬件的灵活控制能力等优点。因此特别适合于以电池为电源的应用场合或手持设备，目前在国内主要应用于三表系统和消防设备方面。MSP430 单片机的开发软件较常用的是 IAR 公司的 IAR Embedded Workbench 集成开发环境，它可以编辑、汇编和编译汇编语言和 C 语言源文件，并且其 C 语言和汇编语言具有相同格式的头文件，给开发带来了灵活性。C 语言具有编程简单，可以移植等优点，但是产生代码较长，对硬件的直接控制能力相对较弱；汇编语言产生的代码较小，控制硬件灵活，但是可读性差，移植困难，因此为了发挥各自优点，产生高速度、高效率的代码混合编程是最好的选择。

1 IAR C 语言编译器的参数传递规则

1.1 寄存器应用

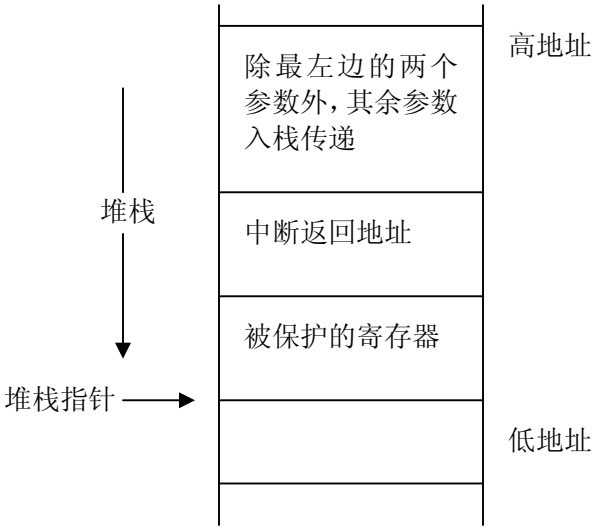
C 语言编译器把单片机的寄存器分成两组来使用：

（1）高速暂存器（R12－R15），这组寄存器专门用作参数传递，因此调用时不需要保护。

（2）其它普通寄存器（R4－R11），这组寄存器主要用作寄存器变量和保存中间结果，因此调用时必需保护，这一点 C 语言编译器是自动处理的。

1.2 堆栈结构和参数传递

每一次函数调用会创建一个如图所示的堆栈结构



一个调用者函数传递给被调用函数的参数按照从右到左的顺序传递的，换句话说就是除了最左边的两个参数用寄存器传递外，其余参数用堆栈传递，并按从右到左的顺序入栈。若最左边的两个参数属于结构或联合类型，那么它们也用堆栈传递。函数的返回结果根据其类型存放在 R12 或 R13:R12 寄存器对，若返回结果属于结构或联合类型，那么 R12 中存放的是指向返回结果的指针。

1.3 中断函数

C 语言编译器编译中断函数时会自动保护所有用到的寄存器（包括 R12－R15 在内），状态寄存器 SR 的保护是中断处理过程自动完成的。中断函数中用到的任何寄存器都会用 PUSH Rxx 指令保护，中断服务结束用后 POP Rxx 指令恢复；RETI 指令会自动恢复状态寄存器 SR 和从中断返回。

2 对汇编语言函数的约定

一个能被 C 语言函数调用的汇编语言函数必须做到以下几点：

（1）符合 C 语言编译器的参数传递规则。

（2）具有 PUBLIC 入口标号。

（3）对 C 语言调用者函数声明为外部函数，并且允许参数类型检查和提升（可选）。

2.1 局部存储分配

如果汇编语言函数需要局部存储空间，有两种分配方法：

（1）分配在硬件堆栈

（2）分配在静态空间，但是函数不能重入。

2.2 中断函数

因为中断可能发生在程序执行的任何期间，所以调用约定并不适用于中断函数。因此必需注意以下几点：

（1）必须保护所有用到的寄存器。

（2）必须用 RETI 返回。

（3）把 SR 中各标志位当做未定义来使用。

（4）中断向量定义在 INTVEC 段

3 混合编程

明确了以上约定，混合编程就非常容易。基本做法是：

（1）C 语言源文件用‘extren’关键字导入被汇编语言源文件导出的标号。

（2）汇编语言源文件用‘PUBLIC’关键字把标号导出给 C 语言源文件。

（3）汇编语言源文件用‘EXTREN’关键字导入被 C 语言源文件导出的标号。

（4）C 语言源文件把标号导出给汇编语言文件，则不需要关键字。

（5）把写好的 C 语言源文件和汇编语言源文件加入工程，并用各自调用函数的指令调用即可。

4 应用实例

4.1 C 语言函数和汇编语言函数相互调用

在这个示例中 C 语言函数 main（）调用汇编语言函数 get_rand()以得到一个随机数；汇编语言函数 get_rand()首先调用 C 语言的标准库函数 rand（）得到一个整型随机值，然后用调用 C 语言函数 mult()的方法把这个随机值乘以 main()函数传递给自己的实参，并把乘积值返回给 main()函数。

4.1.1 C 语言源文件

```
/*************************************************************/
/* 文件名：c_source.c                                2003-01-05 */
/* C 语言和汇编语言混合编程，C 源程序                          */
/* 这段源程序调用汇编语言函数 get_rand()                       */
/* 注意工程必需包含汇编语言源文件 "asm_source.s43"             */
/*************************************************************/
```

```
#include <MSP430x14x.h>        /* 头文件 */

extern unsigned long get_rand(unsigned char seed);       /* 汇编语言函数原型声明 */

/*****************************************************************/
/* 主函数                                                        */
/*****************************************************************/
void main( void )
{
      unsigned char seed;     /* 局部变量定义*/
      unsigned long value;

// === 系统初始化 ========================================
      IFG1 = 0;         /* 清除中断标志 1 */
      WDTCTL = WDTPW+WDTHOLD;     /* 停止看门狗 */
      P1DIR = 0xff;
// === 系统初始化结束=====================================

      seed = 0x55;
      value = get_rand(seed);   /* 调用汇编语言函数 get_rand()得到一个随机数 */
      while(1);     /*程序结束*/
}
// === 主程序结束 ====================================================

/*****************************************************************/
/* 乘法子程序，供汇编语言函数调用 */
/*****************************************************************/
unsigned long mult(int x , int y)
{
      return (x *y);     /*x 乘 y */
}
// === 乘法子程序结束 ================================================
```

4.1.2 汇编语言源程序

```
; ********************************************************************
;  文件名: asm_source.s43
; C 语言和汇编语言混合编程，汇编语言源程序
;  这段源程序调用两个 C 语言函数，标准库函数 rand()和用户自定义函数 mult()
; ********************************************************************
            #include "msp430x14x.h"        ;  头文件
            NAME asmfile

            EXTERN rand                    ; C 语言标准库函数 rand()
            EXTERN mult                    ; c_source.c 中用户自定义函数

;=====================================================================
; get_rand
;=====================================================================
            PUBLIC get_rand                ;  导出函数名给 C 语言函数
            RSEG CODE
```

```
get_rand;
        push R11                    ; 普通寄存器入栈保护
        mov.b R12,R11               ; C 函数传递的实参在 R12 中，送入 R16 暂存

        Call #rand                  ; 调用 C 函数 rand()
                                    ; 函数值为整型返回在 R12 中

                                    ; rand()函数值作为 mult()函数的第一实参
                                    ; 送入 R12 进行参数传递
        mov R11,R14                 ; C 函数传递的实参作为 mult()函数的第二实参
                                    ; 送入 R14 进行参数传递
        Call #mult                  ; mult()值返回在 R12 / R13 寄存器对
        pop R11                     ; 出栈恢复寄存器内容
        ret
END
```

4.2 汇编语言编写中断服务程序

为了提高整个系统响应速度，要求中断服务程序的执行时间较短，执行速度较快，因此最好的方法就是用汇编语言编写中断服务程序。但要注意：1、中断服务程序不能有参数传递和返回值。2、中断服务程序中所有被用到的寄存器都需要保护。本示例用汇编语言编写了看门狗定时器的中断服务程序，用 C 语言编写了主程序。

4.2.1 C 语言主程序

```
/************************************************************************/
/* 文件名：c_main.c                                          2003-01-08 */
/* C 语言和汇编语言混合编程，C 源程序                                    */
/* 这段源程序被看门狗定时器中断后执行汇编语言函数编写的中断服务程序      */
/* 注意工程必需包含汇编语言源文件 "wdt_int.s43"                          */
/************************************************************************/
#include <MSP430x14x.h> /*  头文件  */

/************************************************************************/
/*主函数                                                                */
/************************************************************************/
void main( void )
{
// === 系统初始化 ============================================
    IFG1=0;      /* 清除中断标志 1 */
    WDTCTL=WDT_MDLY_32;      /* 看门狗的定时间隔为 32ms */
    P1DIR = 0x01;    /* P1.0 设置为输出 */
    IFG1 &= ~WDTIFG; /* 清除已挂起的看门狗定时器中断 */
    IE1 |= WDTIE;    /* 允许看门狗定时器中断 */
    _EINT();
// === 系统初始化结束==========================================
    while(1);    /*主程序是一段死循环
}
// === 主函数结束 ============================================
```

4.2.2 汇编语言中断服务程序

```
;*******************************************************************
; 文件名: wdt_int.s43
```

```
; C 语言和汇编语言混合编程，汇编语言源程序
; 看门狗定时器中断服务程序
;*******************************************************************************
      NAME WDT_ISR

      #include "msp430x14x.h"        ; 头文件
;  ============================================================
; 看门狗定时器中断服务程序
;============================================================
          PUBLIC wdt_isr                    ; 导出函数名给 C 语言函数
          RSEG CODE
          wdt_isr
          xor.b #001h,&P1OUT                ; 触发 P1.0,led 亮灭转换
          reti                              ; 中断返回
;============================================================
          COMMON INTVEC(1)                  ; 中断向量段
;============================================================
          ORG WDT_VECTOR
          DW wdt_isr

END
```

5 结束语

以上方法已用于笔者的实际项目，取得良好效果，但是要注意编译器的某些选项对程序生成代码是有影响的。例如：汇编语言函数对标号大小写敏感与否，影响 C 语言函数的变量名、程序名。若使用 ROM MONTIOR，则 C 编译器要用-ur45 选项编译,并且汇编语言中只要使用 R4 和 R5，都要加以保护，否则无法返回 ROM MONTIOR。

参考文献

[1] IAR MSP430 C Compiler Programming Guide
[2] IAR MSP430 Assembler, Linker and Librarian Programming Guide
[3] MSP430x3xx Family User's Guide, literature number SLAU012
[4] MSP430x1xx Family User's Guide, literature number SLAU049
[5] MSP430x4xx Family User's Guide, literature number SLAU056

作者简介:刘玉宏,男,1972 年 9 月生,汉族,河海大学常州校区讲师,主要从事单片机及嵌入式系统方面的教学和研究。E_MAIL：vido_liu@163.com

（213022 江苏常州 河海大学常州校区信息学院）刘玉宏

introduction of author: Liu,yuhong,male,born in may,1972,the Han nationality, the teacher of Hohai university,Changzhou,maily engaged in the teaching and research of MCU and embedded system.

(College of Computer & Information Engineering, Hohai University, Changzhou, Jiangsu, 213022 China) Liu,Yuhong