《嵌入式系统及应用》

2013 版

实验要求

实验报告要求

- 格式方面美观简洁即可，不需要严格按照常规实验报告撰写。
- 必须写学号、姓名；电子版和书面报告均可。
- 实验报告中要能显示上机内容已做（可附关键代码）；对于要求记录结果的问题，请在实验报告上记录结果；对于要求分析的结果，请给出文字分析。
- 除标明为选作的思考题外，需要在实验报告上作答。
- 实验内容和思考题均有编号，在实验报告中请保持编号信息。
- 各个实验单独撰写实验报告，在完成实验项目的一周内提交。

中国科学技术大学 • 电子工程与信息科学系

目录

图表目录

实验1　MPLAB IDE 使用与代码调试

1.1　实验目的

（1）　掌握 IDE 使用、理解一个项目编译连接直至下载到 dsPIC 中的工作过程

（2）　汇编代码编写的一般语法和技巧，掌握编写子程序的方法

（3）　掌握常规代码调试技巧

（4）　理解调试器工作过程

（5）　理解 dsPIC 编程者模型

a)　SFR 的作用

b)　堆栈的工作过程

（6）　理解配置寄存器与其他 SFR 的区别

（7）　了解 LCD 模块工作过程

1.2　预习内容

（1）　请复习如下理论课专题

《心率与健康：数字系统的时钟》

《同芯不同样：dsPIC 内核与编程者模型》

《千里之行始于足下：dsPIC 芯片系统管理》

（2）　请提前预读《dsPIC 30F 程序员手册》并熟悉该文档的结构

1.3　示例说明

表 1-1　实验 1 示例代码功能介绍

Project 名称	Project 中文件名称	功能描述
example1_1.mcw	sample1_1.s	8bit 乘法器
example1_2.mcw	sample1_2_main.s	调用 8bit 乘法器的主程序
	sample1_2_mul_8.s	8bit 乘法器子程序
example1_4.mcw	sample1_4_lcd.s	LCD 显示子程序

	sample1_4_main.s	调用 LCD 显示子程序显示字符串并控制换行的示例
example1_5.mcw	sample1_5_main.s	调用 LCD 显示子程序显示单个字符的简单示例
	sample1_4_main.s	LCD 显示子程序

（1） 8bit 乘法器

a) 由于是第一次做实验，应该介绍在 MPLAB IDE 中如何建立一个 Project。

（2） 子程序调用方法

1.4 实验内容

1.4.1 通过调试简单的 8bit 乘法器练习基本调试技巧

（1） 通过 IDE 环境的主菜单“File”→“Open WorkSpace”打开示例代码的 Project 文件“example1_1.mcw”

（2） 设置代码调试的工具为 ICD2：主菜单“Debugger”→“Select Tool”→“MPLAB ICD2”

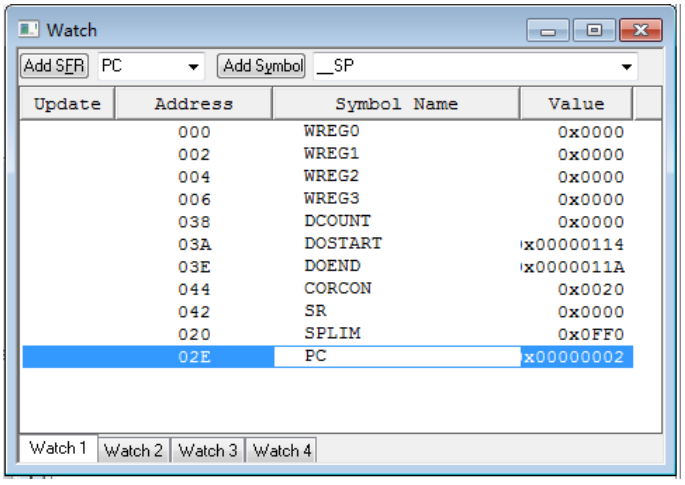
（3） 通过“Project”→“Make”编译、连接生成 HEX 文件

（4） “Debugger”→“Program”将代码烧到 dsPIC0F 4011 片内

（5） 单步执行程序

（6） “View”→“Special Function Registers”

（7） “View”→“Watch”添加对如下寄存器的查看。



1.4.2 子程序调用及参数传递

（8） 通过 IDE 环境的主菜单“File”→“Open WorkSpace”打开示例代码的 Project 文件“example1_2.mcw”，该示例中 Sample1_2_main 中调用了子程序 samp1_2_mul_8。

（9） 在如下位置设置断点，然后单步运行，观察 PC 和 W15 的变化过程。

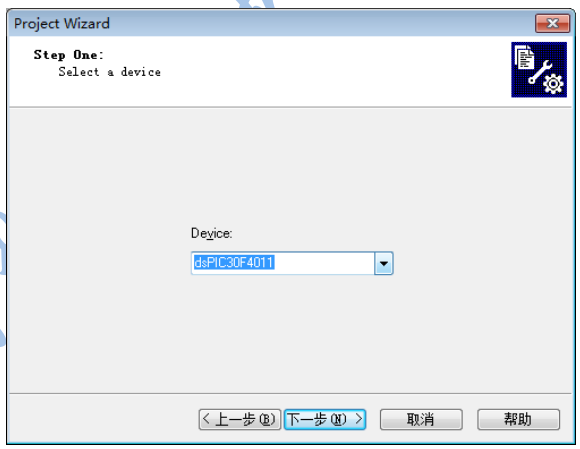
```
MainLoop:
    CLR W10          ;为了加
    NOP
    CALL mul_8
    bra MainLoop
.end
```

a) 注意：实验所用开发板只支持 2 个断点的设置

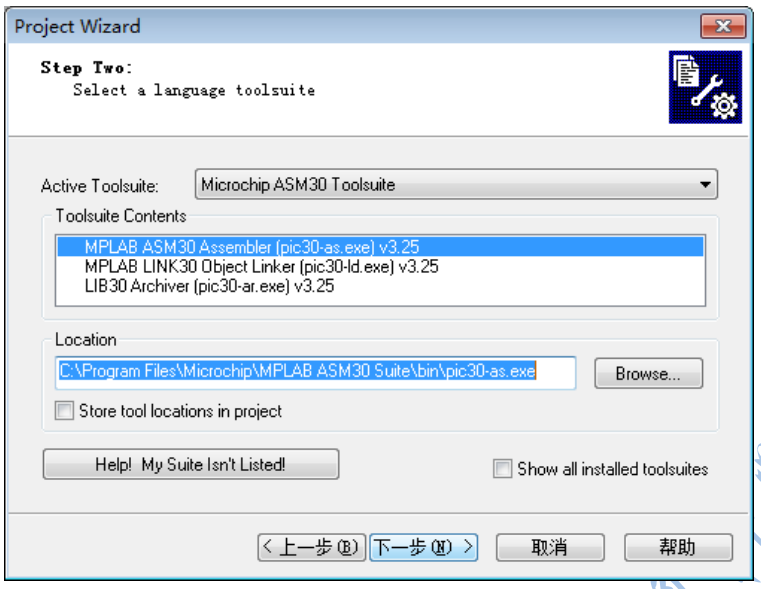
（10） 根据“Sample1_2_main.s”和“Sample1_2_mul_8.s”。编写“lab1_2_main.s”调用“lab_1_2_mul_8.s”函数。“lab1_2_main.s”利用堆栈将乘数和被乘数两个参数传递给“lab_1_2_mul_8.s”。

（11） 通过主菜单“Project”→“Project Wizard”创建新项目。项目的名称为“lab_1_2”

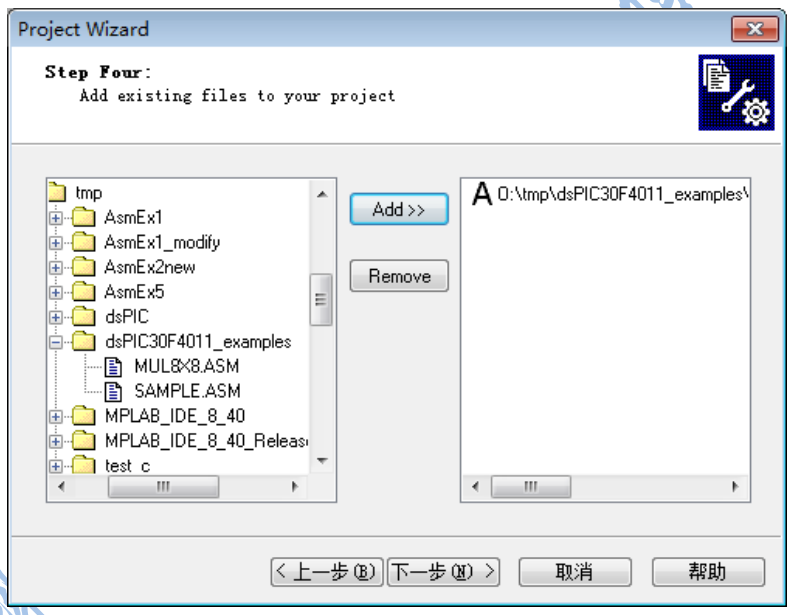
（12） 选择 Device 为“dsPIC30F4011”



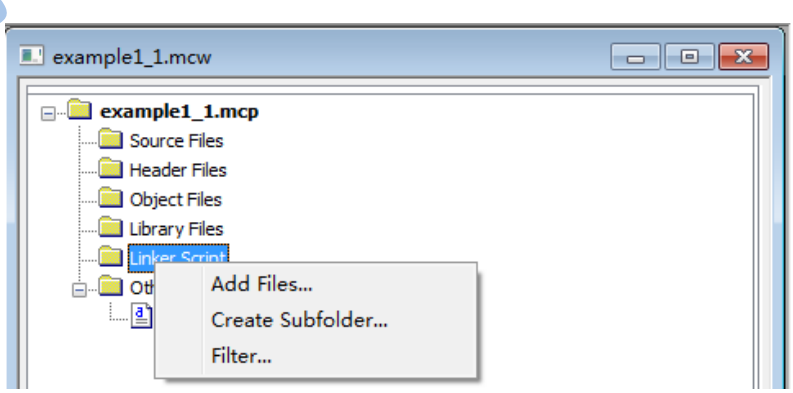
（13） 选择编译器为“MPLAB ASM30 Suite”



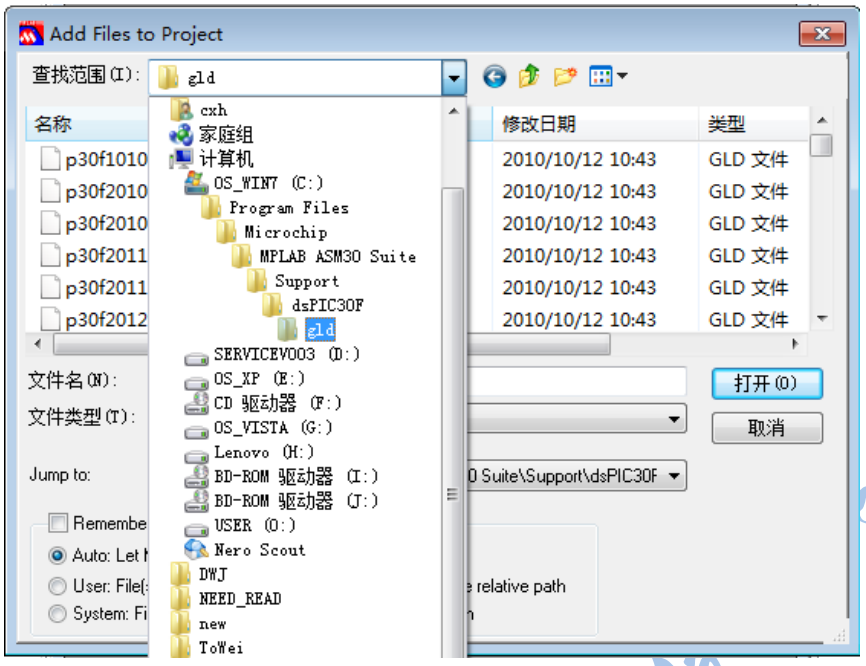
（14） 添加文件到 Project 中。将“lab_1_2_main.s”和“lab_1_2_mul_8.s”添加到项目中。



（15） 添加 GLD 文件



需要添加的 GLD 文件在 Microchip MPLAB IDE 安装目录下的如下位置：



选择与实验所用开发板上一致型号的 GLD 文件：p30f4011.gld

（16） 通过主菜单“Project”→“Project Wizard”创建新项目。项目的名称为“lab_1_2”。将文件“lab1_2_main.s”和“lab_1_2_mul_8.s”加入项目，编译连接测试运行结果是否正常。

（17） 在合适的位置设置断点，然后单步运行，采用合适的方法（图或者文字描述）记录堆栈内容的变化情况。

1.4.3 建立 Project 完成 10bit 乘法器设计

（18） 根据“Sample1_2_mul_8.s”所提供的 8bit 乘法器 ASM 代码，做一个 10bit 的乘法器。编写完整代码：代码中乘法功能要写成子函数，单独保存为文件“lab_1_3_mul10.s”。

a) 提示：可以先做 10bit 被乘数与 8bit 乘数的乘法，然后再分别处理乘数的最高 2 个比特，将所有部分积结果进行合适的移位后累加。

（19） 根据“Sample1_2_main.s”编写调用“lab_1_3_main.s”的主函数。

（20） 通过主菜单“Project”→“Project Wizard”创建新项目。项目的名称为“lab_1_3”

（21） 编译、连接项目“lab_1_3”，验证自己的 10bit 乘法器结果是否正确。

1.4.4 学习如何使用 LCD 进行字符串

（22） 通过 IDE 环境的主菜单“File”→“Open WorkSpace”打开示例代码的 Project 文件“example1_4.mcw”，该示例中 Sample1_4_main 中调用了子程序 sample1_4_lcd。阅读

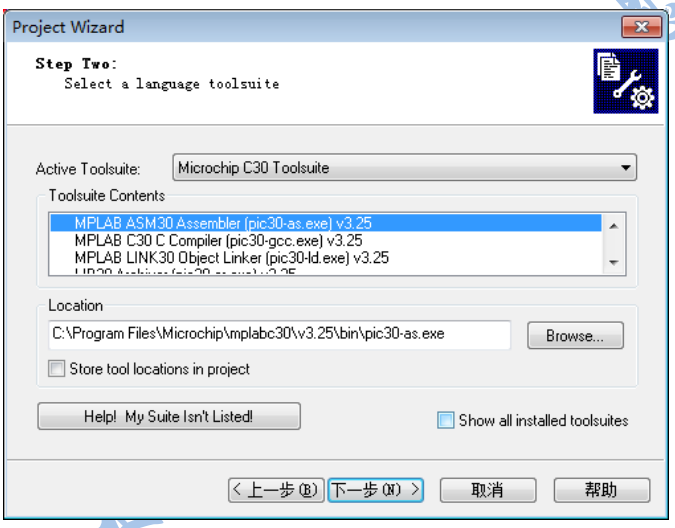
程序学习使用 LCD 显示一个字符串的基本过程。

1.4.5 使用 LCD 显示 10 比特乘法器的输出结果

（23） 通过 IDE 环境的主菜单“File”→“Open WorkSpace”打开示例代码的 Project 文件“example1_5.mcw”，该示例中 Sample1_5_main 中调用了子程序 samp1_4_lcd。阅读程序学习使用 LCD 显示一个字符的简单过程。

（24） 根据“Sample1_5_main.s”编写一个在 LCD 上显示 W6 寄存器内容的程序“lab_1_5_main.s”

（25） 通过主菜单“Project”→“Project Wizard”创建新项目。项目的名称为“lab_1_5”，将“lab_1_3_mul10.s”、“lab_1_5_main.s”和“sample1_4_lcd.s”加入项目。请选择 C30 编译器。



（26） 修改“lab_1_5_main.s”调用 10 比特乘法器。验证程序是否能够在 LCD 上显示乘积结果。

1.4.6 理解系统配置寄存器

（27） 计算“lab_1_2_mul10.s”中乘法需要的时间。

a) 提示，参考代码中的配置为信息

```
; 配置位
    config __FOSC, CSW_FSCM_OFF & XT_PLL4                ;
    config __FWDT, WDT_OFF                               ;
    config __FBORPOR, PBOR_OFF & BORV_27 & PWRT_16 & MCLR_EN    ;
    config __FGS, CODE_PROT_OFF                          ;
;---------------------------------------------------------------
; 与程序有关的常数（代码中使用的立即数）
    .equ FCY, #7372800          ; 指令周期频率（Osc x PLL / 4）
;===============================================================
```

b) 打开“C:\Program Files\Microchip\MPLAB ASM30 Suite\Support\dsPIC30F\inc\p30f4011.inc”文件查看关于 XT_PLL4 的定义（该文件可能由于 MPLAB IDE 的安装路径不同位置不同）。

（28） 将代码中关于时钟的配置信息修改如下，计算所配置情形下“lab_1_2_mul10.s”中乘法需要的时间。

原来代码： config __FOSC, CSW_FSCM_OFF & XT_PLL4 ;

修改为： config __FOSC, CSW_FSCM_OFF & LPRC ;

1.5 思考题（内容需要上机调试才能完成）

（29） 将自己学号末三位（应该是一个小于 512 的正整数）作为被乘数，实验当天的日（应该是个小于 32 的正整数），你所设计的乘法器输出结果是多少？

（30） “sample1_1.s”中关于堆栈的配置情形下，使用堆栈进行参数传递，主程序最多可以向子程序传递多少个参数？（仅考虑主程序调用一个子程序，无嵌套调用的情形）

（31） 查看项目中 GLD 文件“p30f4011.gld”中存储的信息，这个文件的主要用途是什么？

（32） 配置寄存器中 Brown Out Voltage 最小可以设置为多少还能保证 dsPIC 正常工作，为什么？【*选做】

a) 提示：可查询数据手册中芯片和管脚 DC 特性有关内容

（33） 你所设计的乘法器运行过程中可能修改系统标志寄存器的哪些位？如果采用系统的乘法指令用哪条（或者哪几条可以实现 10bit 的乘法）【*选做】

实验2　存储器访问与控制

2.1　实验目的

（1）　理解 dsPIC 数据存储区的寻址方式：模寻址、位反转寻址

（2）　理解 dsPIC 程序存储区的寻址方式：PC、表操作、PSV 映射

（3）　陷阱和中断的使用方法

（4）　掌握定时器的基本使用方法

（5）　掌握 I/O Ports 的基本操作方法

2.2　预习内容

（1）　请复习如下理论课专题

《同芯不同样：dsPIC 内核与编程者模型》

《调试与设计哪个更重要：Debugger》

《无序的管理：中断》

《数字电路那点事：I/O 端口》

《寻址的艺术：dsPIC30F 的存储器》

《计数的名堂：定时器/计数器、输入捕捉/输出比较》

（2）　请提前预读《dsPIC 30F 程序员手册》并熟悉该文档的结构

2.3　示例说明

表 2-1　实验 2 示例代码功能介绍

Project 名称	Project 中文件名称	功能描述
example2_1.mcw	sample2_1_data_memory.s	数据空间访问基本寻址方式演示
	sample1_4_lcd.s	实验 1 中所用 LCD 模块
example2_2.mcw	sample2_2_io_ports_basic.s	端口操作基本方法演示
example2_3.mcw	sample2_3_io_ports_timer.s	基于定时器实现的端口控制

example2_4.mcw	sample2_4_program_memory.s	程序存储器基本访问方法演示
example2_5.mcw	sample2_5_address.s	位反转寻址和模寻址演示

2.4 实验内容

2.4.1 Data Memory 操作（8bit/16bit，乘法器结果显示到 LCD）

（1） 通过 IDE 环境的主菜单“File” “Open WorkSpace”打开示例代码的 Project 文件“example2_1.mcw”。

（2） 将断点设置在下图所示位置，观察 W0-W15 变化状况，你有什么结论（记录在实验报告上）

```
CLR W10          ;为了加调试断点添加的代码
;直接寻址与间接寻址
;****"View"-->"Watch": 观察W0-W12的变化
MOV #0, W11              ;MOV #lit16,Wnd 将16 位立即数传送到Wnd
MOV #0, W12
DO #8, END_INIT_1
    MOV W12, [W11]       ;MOV Wns,[Wd+Slit10] 将Wns 中的内容传送到[Wd + 有符号10位偏移量]
    ADD #2, W11          ;ADD #lit10,Wn Wn = lit10 + Wn
    ADD #1, W12
END_INIT_1:
NOP
```

（3） 将断点设置在下图所示位置，观察 W0-W15 变化状况，你有什么结论（记录在实验报告上）

```
CLR W10          ;为了加调试断点添加的代码
;递增方式的间接寻址
;****"View"-->"Watch": 观察W0-W12的变化
MOV #0, W11
MOV #0, W12
DO #8, END_INIT_2
    MOV W12, [W11++]
    ADD #1, W12
END_INIT_2:
    NOP
```

（4） 将断点设置在下图所示位置，读懂这一段代码，观察结果是否和预期一致（记录在实验报告上）

```
CLR W10          ;为了加调试断点添加的代码
;从RAM取两个数，运算后再存到RAM
;将两个数写入RAM
MOV #0x0B00, W8          ;W8用于间接寻址
MOV #4, W7               ;W7保存立即数
MOV W7, [W8++]
MOV #2, W7               ;W7保存立即数
MOV W7, [W8++]

;读出RAM中存放的两个数
MOV [--W8], W6               ;MOV [Ws+Slit10],Wnd 将 [Ws + 有符号10 位偏移量] 中的内容传送到Wnd
MOV [--W8], W5               ;MOV [Ws+Slit10],Wnd 将 [Ws + 有符号10 位偏移量] 中的内容传送到Wnd

;将W5和W6中保存的两个数相乘并存入地址 0x1804
MOV #0x0B04, W8              ;W8用于间接寻址
ADD W5, W6, W7
MOV W7, [W8]
;读出计算结果
MOV [W8], W5                 ;MOV [Ws+Slit10],Wnd 将 [Ws + 有符号10 位偏移量] 中的内容传送到Wnd
```

（5） 阅读《dsPIC30F4011/4012 数据手册》DS70135E_CN 第 26 页，单步调试如下代码，运行结果是否和预期一致？解释运行的结果（记录在实验报告上）。

```
MOV #0x1800, W8              ;W8用于间接寻址
MOV W7, [W8++]
```

（6） 根据“sample2_1_data_memory.s”编写“lab2_1_data_memory.s”，功能为：从 RAM 地址“0x0B10”和“0x0B12”读取两个数→调用在实验 1 中自己编写的 10bit 乘法器→乘积保存到地址“0x0B20”。

（7） 修改通过主菜单“Project”→“Project Wizard”创建新项目。项目的名称为“lab_2_1”，将“lab2_1_data_memory.s”和“sample1_4_lcd.s”加入项目。对“lab2_1_data_memory.s”做相应的修改使之能够将乘积结果写入 RAM 的同时在 LCD 上显示。

2.4.2 I/O Ports 操作（板上所有灯点亮）

（8） 通过 IDE 环境的主菜单“File”→ “Open WorkSpace”打开示例代码的 Project 文件“example2_2.mcw”。看懂“sample2_2_io_ports_basic.s”，观察代码执行情况是否能够将开发板上 PORTE 和 PORTF 对应的 LED 点亮→熄灭→点亮→熄灭→点亮→熄灭…。

（34） 阅读开发板原理图“Sch_APP009_20050110.pdf”查找开发板上发光二极管 D7-D10 对应的端口信息。自己根据“sample2_2_io_ports_basic.s”编写一个控制 D7-D10 亮灭的程序“lab2_2_io_ports_basic.s”；通过主菜单“Project”→“Project Wizard”创建新项目。项目的名称为“lab_2_2”，将“lab2_2_io_ports_basic.s”加入项目。演示结果要检查。

2.4.3 定时器与中断（乘法器溢出时候使能定时器中断服务）

（9） 通过 IDE 环境的主菜单“File”→ “Open WorkSpace”打开示例代码的 Project 文件“example2_3.mcw”。看懂“sample2_3_io_ports_timer.s”，观察代码执行情况是否能够将开发板上的 D13 和 D14 进行点亮→熄灭→点亮→熄灭→点亮→熄灭…。

（10） 根据“sample2_3_io_ports_timer.s”编写“lab2_3_io_ports_timer.s”，要求完成功能：

使用 Timer2 作为定时控制器、D13 或 D14 亮灭的周期为 2 秒。

（11） 通过主菜单“Project”→“Project Wizard”创建新项目。项目的名称为“lab_2_3”，将“lab2_3_io_ports_timer.s”加入项目，验证功能是否正常。

（12） 将“sample1_4_lcd.s”加入项目“lab_2_3”，做相应的修改使程序能够在 D14 亮的时候在 LCD 上显示字符串（内容自定义），灭的时候 LCD 上不显示任何字符。

（13） 修改项目“lab_2_3”，对两个 16bit 数做乘法（直接用系统指令），当计算发生溢出后 D13 和 D14 全部点亮不灭。

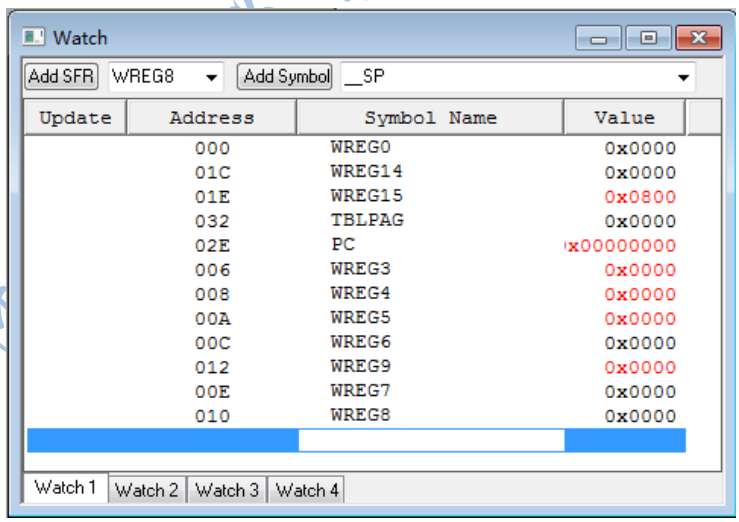
2.4.4 Program Memory 操作（乘法器读取乘数和被乘数，乘法结果保存，代码启动在 LCD 上显示乘法结果）

（14） 通过 IDE 环境的主菜单“File”→ “Open WorkSpace”打开示例代码的 Project 文件“example2_4.mcw”。看懂“sample2_4_program_memory.s”。

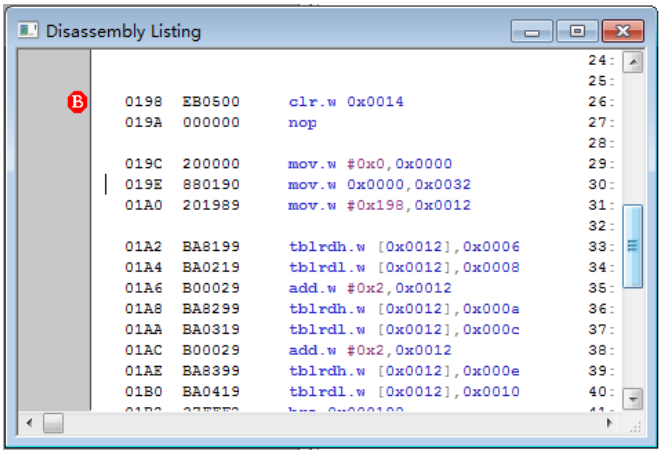
（15） 在如下位置设置断点，单步运行其中关于读取 FLASH 内容的代码段。

```
_main:
    CLR W10          ;为了加调试断点添加的代码
    NOP
```

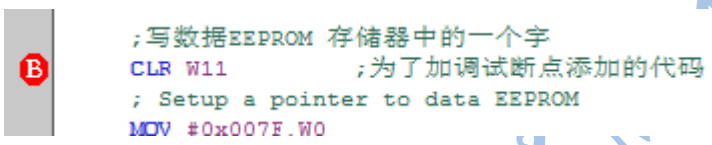
（16） 观察如下寄存器信息



（17） IDE 主菜单“View”→“Dissembly Listing”打开汇编代码窗口，观察上一步骤中单步调试过程中“Watch”窗口和“Dissembly Listing”窗口的对应关系。在实验报告上记录这段代码执行后 W3、W4、W5、W6、W7、W8 存放的是什么信息？NOP 指令的机器码是什么？



（18） 在如下位置设置断点，单步运行其中关于写 EEPROM 内容的代码段。



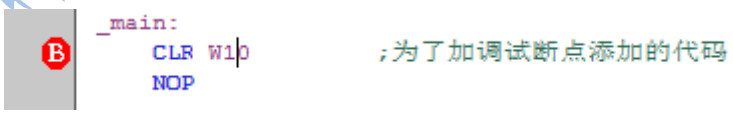
（19） 根据“sample2_4_program_memory.s”编写“lab2_4_program_memory.s”，功能为：从 FLASH 地址读取上一步骤中断点所在行命令的机器码到 W7 和 W8 寄存器→调用在实验 1 中自己编写的 10bit 乘法器→乘积保存到 EEPROM 地址“0x007FFFE0”。

（20） 通过主菜单“Project”→“Project Wizard”创建新项目。项目的名称为“lab_2_4”，将“sample2_4_program_memory.s”加入项目，验证功能是否正常。

2.4.5 Data Memory 位反转寻址与模寻址

（21） 通过 IDE 环境的主菜单“File”→ “Open WorkSpace”打开示例代码的 Project 文件“example2_5.mcw”。看懂“sample2_5_address.s”。

（22） 在如下位置设置断点，单步运行其中关于模寻址的代码段。记录此段代码执行完毕后 W1 的值，



如果将

MOV #0x080F,W0

MOV W0,XMODEND ;set modulo end address

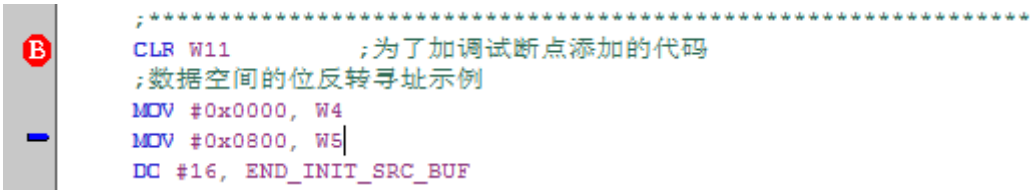
修改为：

MOV #0x0803,W0

MOV W0,XMODEND ;set modulo end address

这段代码执行完 W1 的值又是多少（记录到实验报告）？

（23） 在如下位置设置断点，单步运行其中关于，在实验报告上记录 W0 和 W1 值的变化过程。



```
;*********************************************************************
CLR W11          ;为了加调试断点添加的代码
;数据空间的位反转寻址示例
MOV #0x0000, W4
MOV #0x0800, W5
DO #16, END_INIT_SRC_BUF
```

2.5 思考题

（24） 项目“example2_1.mcw”中“sample2_1_data_memory.s”文件如下代码将立即数“4”写入了 RAM 哪个地址？如果需要将一个 8bit 的立即数“0x10”写入 RAM 的地址“0x0A01”，需要对代码做和修改，给出修改后的代码。

```
MOV #0x0B00, W8          ;W8用于间接寻址
MOV #4, W7               ;W7保存立即数
MOV W7, [W8++]
```

（25） 根据项目“example2_1.mcw”的调试过程；阅读《dsPIC30F/33F 程序员参考手册》的“4.1.3 寄存器间接寻址”部分。总结读取 RAM 数据的时候有哪些可用的寻址方式？分别给出示例。

（26） 项目“example2_2.mcw”中“sample2_2_io_ports_basic.s”文件中如果将写 LTEx 修改为 PORTx（如下图所示），代码执行是否还能完成预期亮灭 LED 的功能？这两种方式有什么区别？

```
;MOV W6, LATE
;MOV W6, LATF
MOV W6, PORTE
MOV W6, PORTF
```

（27） 阅读《dsPIC30F4011/4012 数据手册》的“3.1 程序地址空间”，如果执行如下代码，可能会产生什么结果？

```
MOV #0xF800,W0
MOV W0,TBLPAG
MOV #0x0000,W0
MOV #0x0806,W1
TBLWTL W1,[ W0]
```

（28） 阅读《dsPIC30F4011/4012 数据手册》的“图 3-6： dsPIC30F4011/4012 的数据存储空间映射”，然后分析如下代码执行可能会产生什么结果？

```
MOV #0x1100,W0
```

```
MOV W0,XMODSRT ;set modulo start address
MOV #0x1163,W0
MOV W0,XMODEND ;set modulo end address
MOV #0x8001,W0
MOV W0,MODCON ;enable W1, X AGU for modulo
MOV #0x0000,W0 ;W0 holds buffer fill value
MOV #0x1100,W1 ;point W1 to buffer
DO #49,FILL ;fill the 50 buffer locations
FILL:
MOV W0,[W1++] ;fill the next location
```

（29） 将乘数和被乘数分别放到 RAM 的 X 空间和 Y 空间（地址自行确定）。比较三种乘法完成需要的指令周期数。【*选做】

a) 自己实验 1 中实现的 10 比特乘法器

b) 系统 MCU 乘法指令

c) 系统 DSP 乘法指令

实验3　ADC 与 PWM

3.1　实验目的：外设访问+中断

（1）　各类定时器使用

（2）　I/O Ports 读写

（3）　AD 读写（对 FIFO 进行读写和对端口进行读写有什么差异？）

（4）　用比较器输出连续脉冲

（5）　简单的 PWM 控制

3.2　预习内容

（1）　请复习如下理论课专题

《同芯不同样：dsPIC 内核与编程者模型》

《无序的管理：中断》

《寻址的艺术：dsPIC30F 的存储器》

《感知世界：ADC》

《从数字到模拟的抉择：PWM》

（2）　请提前预读《dsPIC 30F 程序员手册》并熟悉该文档的结构

3.3　示例说明

表 3-1　实验 3 示例代码功能介绍

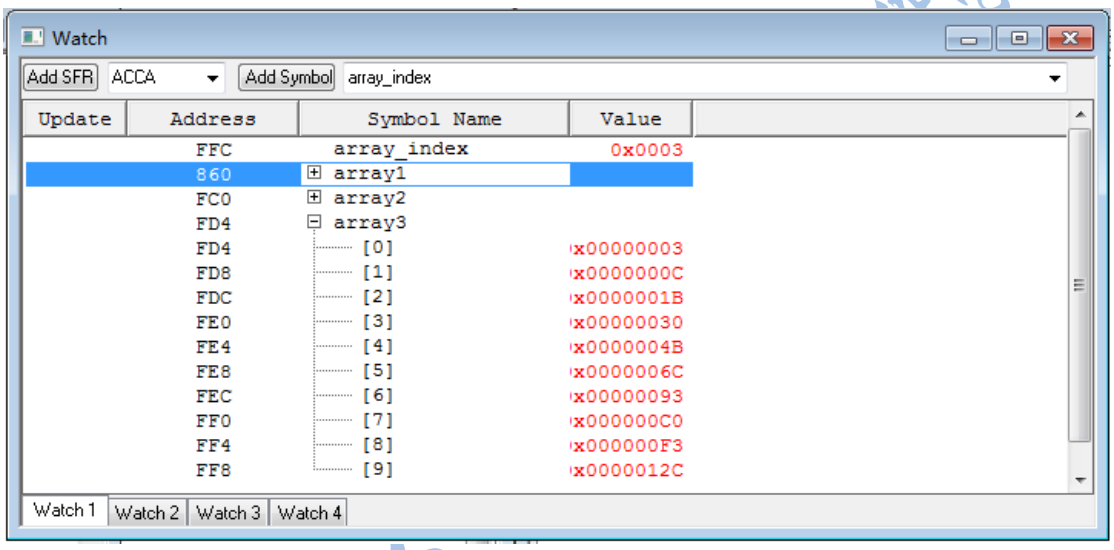
Project 名称	Project 中文件名称	功能描述
example3_1.mcw	sample3_1_main.c	用 C 语言调用模寻址子模块
	sample3_1_modulo.s	ASM 模寻址子程序
example3_2.mcw	sample3_2_main.c	手动 AD 示例
example3_3.mcw	sample3_3_main.s	用 A/D 实现的键控累加器并将结果显示到 10 个发光二极管上
	sample1_4_lcd.s	实验 1 中所用 LCD 模块
example3_4.mcw	sample3_4_main.s	PWM 示例

3.4 实验内容

3.4.1 使用 C 与 ASM 混合开发 dsPIC 程序

（1） 通过 IDE 环境的主菜单“File”→“Open WorkSpace”打开示例代码的 Project 文件“example3_1.mcw”。这是一个 C 语言作为主程序示范模寻址的示例，请阅读程序。

（2） IDE 主菜单“View”→“Watch”,；练习通过 “Add Symbol”添加对变量的查看（如下图所示）。



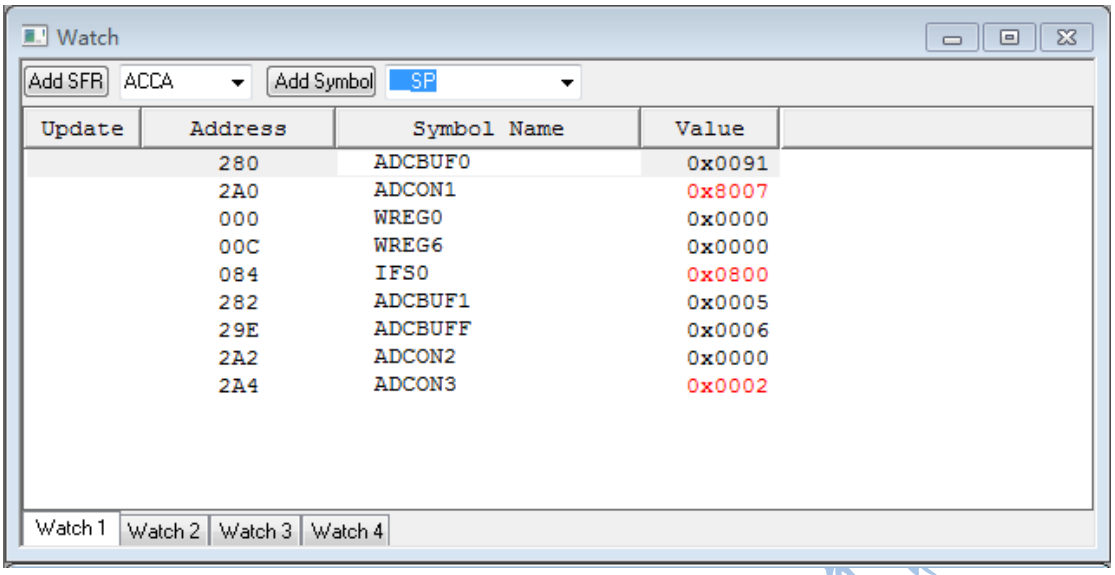
（3） 分析主程序中“array1”、“array2”和“array3”三个数组的关系，在实验报告中记录结论。

3.4.2 手动采样 AD 的基本操作

（4） 通过 IDE 环境的主菜单“File”→“Open WorkSpace”打开示例代码的 Project 文件“example3_2.mcw”。请在如下位置添加断点，单步执行

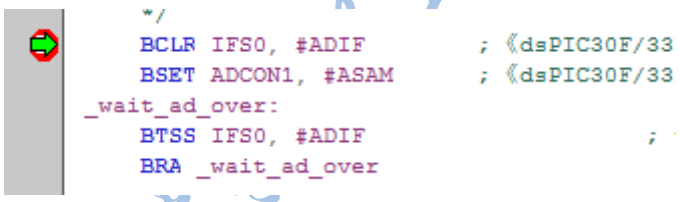
```
_wait_convert_over:
    CLR W11
    BTSS ADCON1, #DONE                  ; 《d
    BRA _wait_convert_over

    MOV ADCBUF0, W0             ;读取AD样本
```

观察如下寄存器变化情况

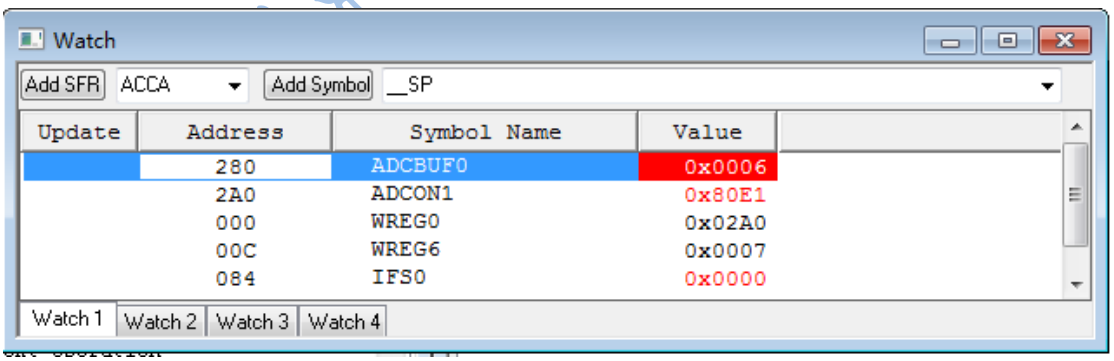


3.4.3 自动采样 AD 的基本操作【*选做】

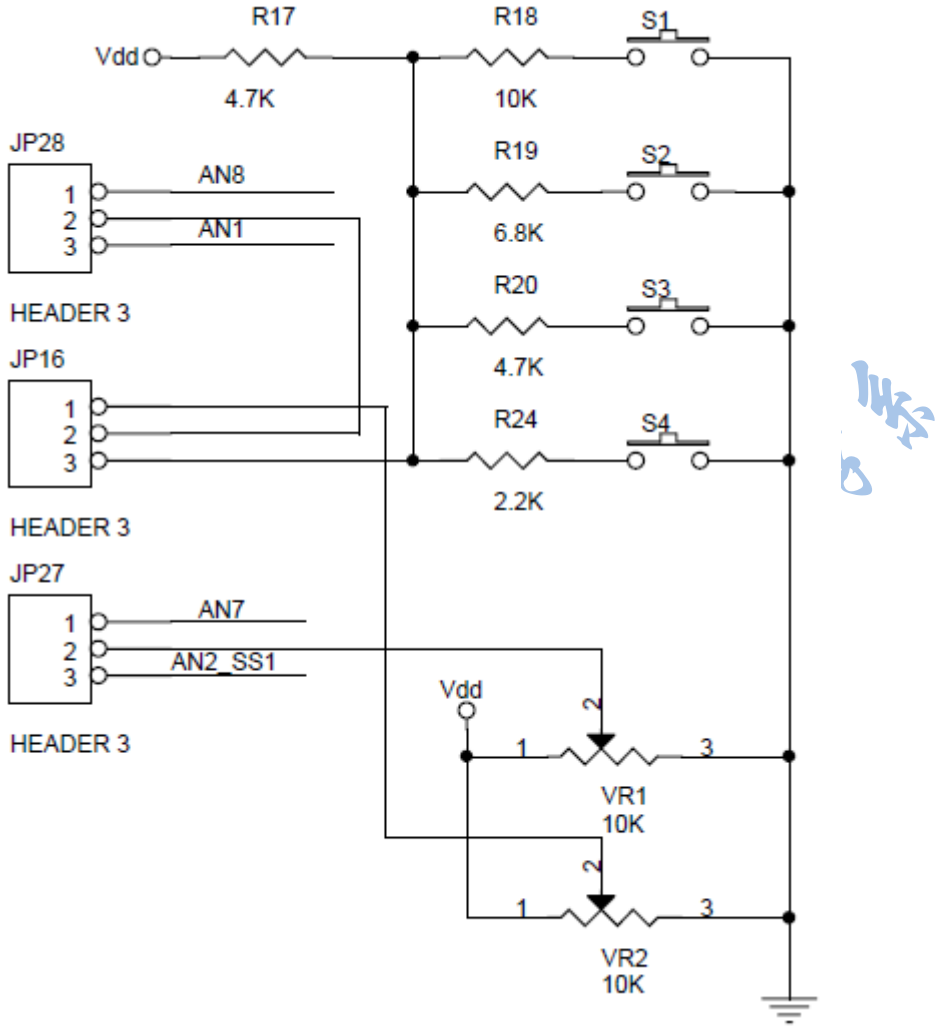
（5） 通过 IDE 环境的主菜单“File”→“Open WorkSpace”打开示例代码的 Project 文件“example3_3.mcw”。读懂代码。请在如下位置添加断点，单步执行



观察如下寄存器变化情况



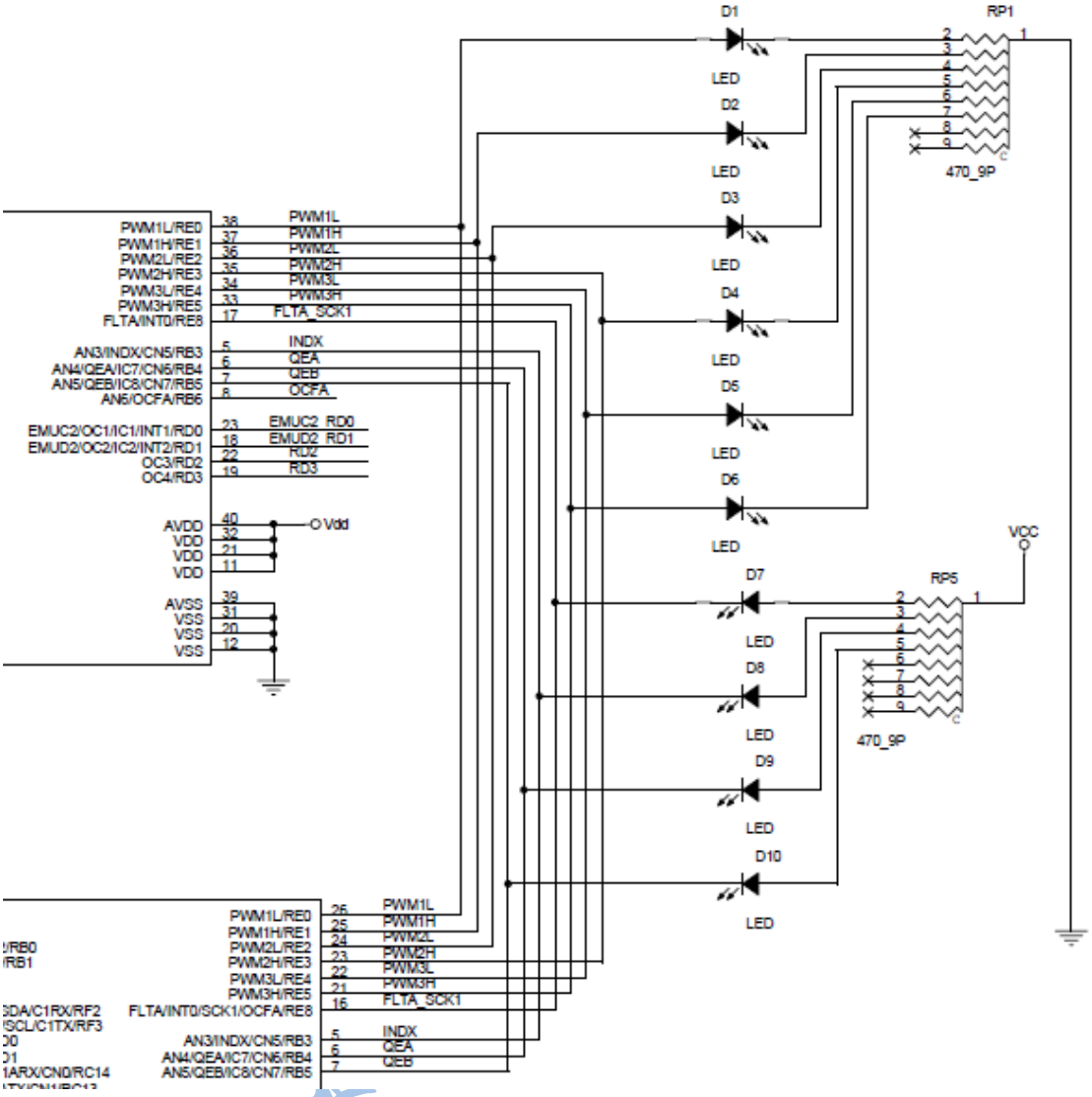
（6） 根据“sample3_3_interrupt_ad.s”所示自动 AD 的过程，编写“lab3_3_main.s”。要求能够利用 AD 读入 VR1 和 VR2 的值。【VR1 到底对应 AN2 还是 AN7、VR2 到底对应 AN1 还是 AN8 可通过采样值确定。下图为开发板原理图中有关部分】



（7） 通过主菜单“Project”→“Project Wizard”创建新项目。新建过程中请选择编译器为“MPLAB C30…”。项目的名称为“lab_3_3”，将“lab3_3_main.c”和“sample1_4_lcd.s”加入项目。修改“lab3_3_main.c”使之能够将两数的比值（VR1/VR2）显示在 LCD 上。

3.4.4 PWM 模块简单控制

（8） 通过 IDE 环境的主菜单“File”→“Open WorkSpace”打开示例代码的 Project 文件“example3_4.mcw”。阅读代码中的注释，读懂代码；观察代码运行效果。提示：原理图中 PWM 输出管脚连接如下。



（9） 如果将代码中如下四行注释掉的代码使能，运行结果有何差异？

```
MOV #0x0FFF, W0
MOV W0, PWMCON1
;BCLR PWMCON1, #8
;BCLR PWMCON1, #9
;BCLR PWMCON1, #10
;BCLR PWMCON1, #11
```

（10） 利用 PWM 模块以 VR1/VR2 作为占空比输出方波。【根据实验室条件选择是否做：用示波器查看】。

3.5 思考题

（11） 查看如下 3 个文件（实际机器上安装的目录可能与下面列出的不一致）并分析对比，在实验报告中记录 TRISE 寄存器的地址。【*选做】

a) “C:\Program Files\Microchip\mplabc30\v3.25\support\dsPIC30F\inc\p30f4011.inc”

b) “C:\Program Files\Microchip\mplabc30\v3.25\support\dsPIC30F\h\p30f4011.h”

c) “C:\Program Files\Microchip\mplabc30\v3.25\support\dsPIC30F\gld\p30f4011.gld”

（12） AD 模块中 FIFO 的作用？是否越大越好，FIFO 容量是靠什么确定的？【*选做】

（13） FIFO 的电路实现和 RAM 有何区别？实际 dsPIC 中 FIFO 的访问方式和 RAM 访问方式有何区别？【*选做】

（14） 示例“example3_4.mcw”中 D2 亮灭的周期是多少？控制 D2 的 PWM 输出波形的占空比是多少？【*选做】

实验4　DSP 引擎特性

4.1　实验目的

（1）　理解 CPU 和 DSP 的本质差异

（2）　掌握 PSV 方式访问 FLASH 的方法

（3）　学习在 dsPIC 上使用 DSP 算法的方法

4.2　预习内容

（1）　请复习如下理论课专题

《同芯不同样：dsPIC 内核与编程者模型》

《寻址的艺术：dsPIC30F 的存储器》

《小数点引发的血案：dsPIC 的 DSP 引擎》

（2）　请提前预读《dsPIC 30F 程序员手册》并熟悉该文档的结构

4.3　示例说明

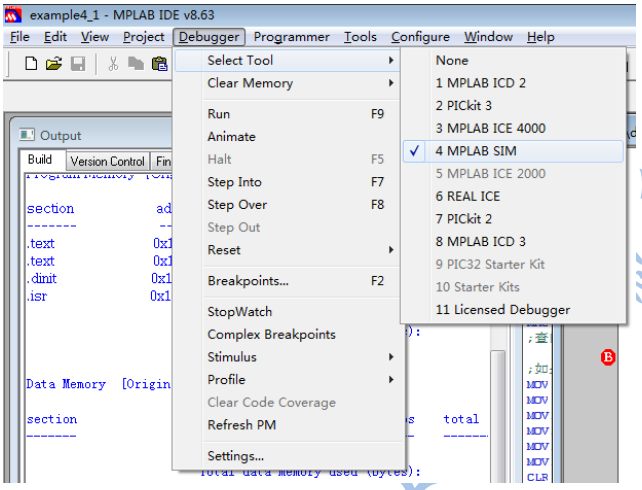
表 4-1　实验 4 示例代码功能介绍

Project 名称	Project 中文件名称	功能描述
example4_1.mcw	sample4_1_main.c	常用累加器操作指令
example4_2.mcw	sample4_2_main.c	常用 MAC 指令
Save_Restore_PSVPAG		使用 C 的 PSV 访问
example4_3_2.mcw	sample4_3_2_main.s	使用 ASM 的 PSV 访问
CE005_FIR_DSP_lib_Filter		基于 C 的 FIR 示例
CE018_FFT_DSPlib		基于 C 的 FFT 示例

4.4 实验内容

4.4.1 数据格式调整与累加器操作

（1） 通过 IDE 环境的主菜单“File”→“Open WorkSpace”打开示例代码的 Project 文件“example4_1.mcw”。请将 Debugger 设置为“MPLAB SIM”完成下面的实验内容。



（2） 符号扩展【请参考示例代码中的要求完成实验】

（3） 饱和逻辑【请参考示例代码中的要求完成实验】

（4） 舍入逻辑【请参考示例代码中的要求完成实验】

4.4.2 DSP 乘法器与 MAC 指令

（5） 通过 IDE 环境的主菜单“File”→“Open WorkSpace”打开示例代码的 Project 文件“example4_1.mcw”。请将 Debugger 设置为“MPLAB SIM”完成下面的实验内容。

（6） 学习数据换算指令的使用【请参考示例代码中的要求完成实验】

（7） 对比 MCU 乘法指令和 DSP 乘法指令【请参考示例代码中的要求完成实验】

（8） 理解 MAC 指令【请参考示例代码中的要求完成实验】

4.4.3 使用 PSV 访问程序存储器【*选做】

（9） 在 C30 下使用 PSV 功能。读懂示例“Save_Restore_PSVPAG”。

（10） 在 ASM 下使用 PSV 功能。读懂示例“example4_3_2.mcw”，结合上一实验内容进行分析。在实验报告上说明 PSV 访问的步骤；说明 PSVPAG 寄存器中存放的什么内容。

4.4.4 用 dsPIC 设计 FIR【*选做】

（11） 读懂示例“CE005_FIR_DSP_lib_Filter”。在实验报告上画出用 dsPIC 做 FIR 设计的基本步骤；分析 FilterOut、square1k、bandpassexampleFilter 分别存放在存储器空间的什么位置（结论写到实验报告上）？

4.4.5 用 dsPIC 做 FFT【*选做】

（12） 读懂示例“CE018_FFT_DSPlib”。在实验报告上画出主程序流程图。

4.5 思考题

（13） 使用 PSV 方式访问程序存储器和在实验内容 2.4.4 中采用表方式在应用方面存在哪些差异（即各自适合哪些场景）？

（14） DSP 与通用 CPU 的最本质区别是什么？【*选做】

（15） 数据换算（normalization (scaling)）指令如“FBCL”、“FF1L”、“FF1R”可能会在哪些场合使用？

实验5　简单任务管理与 FreeRTOS 使用

5.1　实验目的

（1）　理解 COFF 文件格式

（2）　理解操作系统调度的基本原理。

（3）　理解课程中进程间通信的基本原理。

（4）　了解 FreeRTOS 实现的功能。

5.2　预习内容

（1）　请复习如下理论课专题

《同芯不同样：dsPIC 内核与编程者模型》

《科学的管理：嵌入式 OS 与 RTOS 基础》

（2）　请提前预读《dsPIC 30F 程序员手册》并熟悉该文档的结构

5.3　示例说明

表 5-1　实验 5 示例代码功能介绍

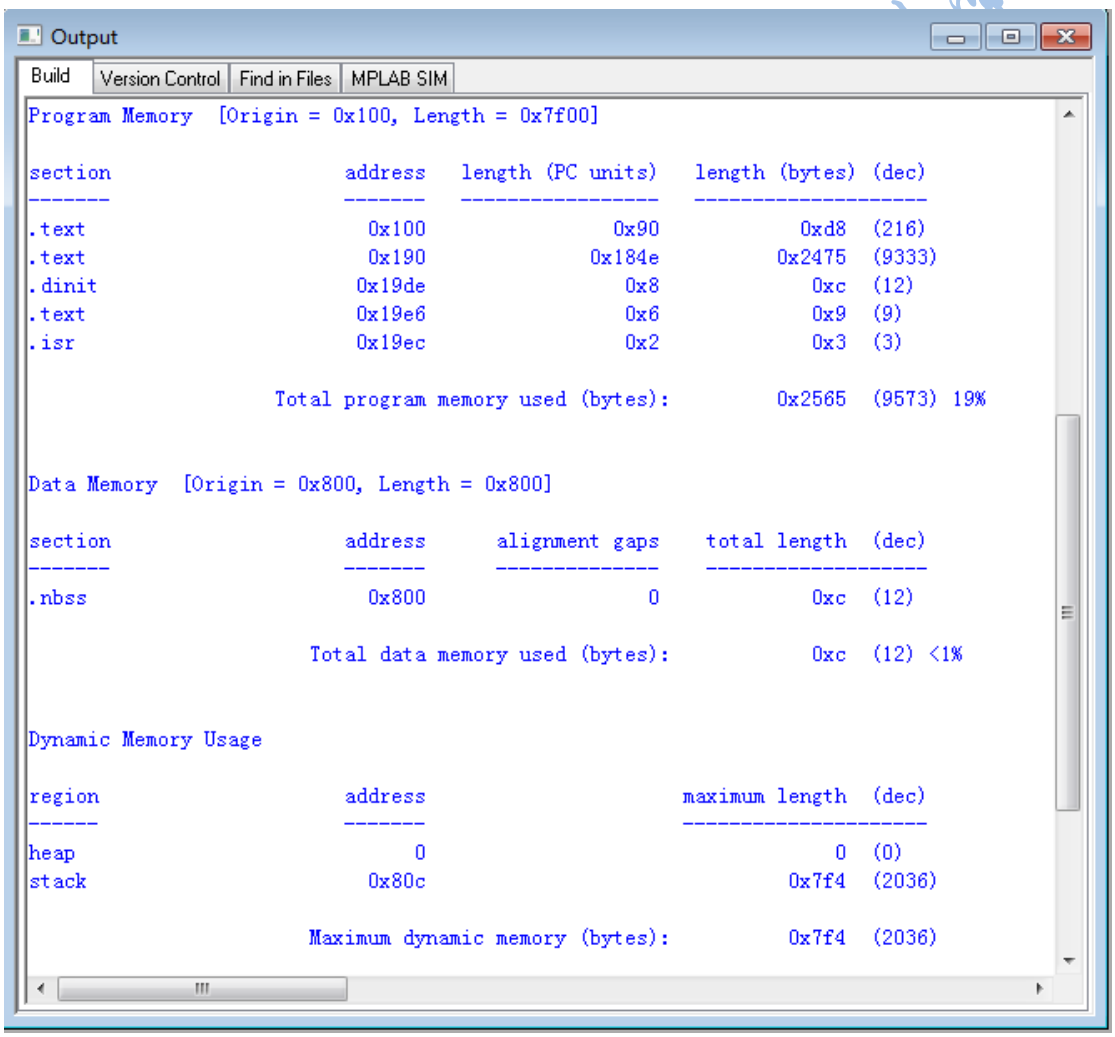
Project 名称	Project 中文件名称	功能描述
example5_1.mcw	sample5_1_main.c	用作调度器的主函数
	sample5_1_task1.s	控制 PORTE 的示例子任务 1
	sample5_1_task2.s	子任务 2（空函数）
	sample5_1_task3.s	子任务 3（空函数）
RTOSDemo.mcw	main.c	测试代码主程序
	lcd.c	测试代码：LCD 显示控制
	ParTest.c	测试代码：开发板 LED 控制
	timertest.c	测试代码：定时器功能测试
	tasks.c	FreeRTOS Kernel 代码：任务管理 API
	queue.c	FreeRTOS Kernel 代码：消息队列管理 API
	list.c	FreeRTOS Kernel 代码：list API

	RTOS 代码简介.doc	本示例代码的说明
	FreeRTOS_cn.pdf	FREERTOS 实时内核实用指南

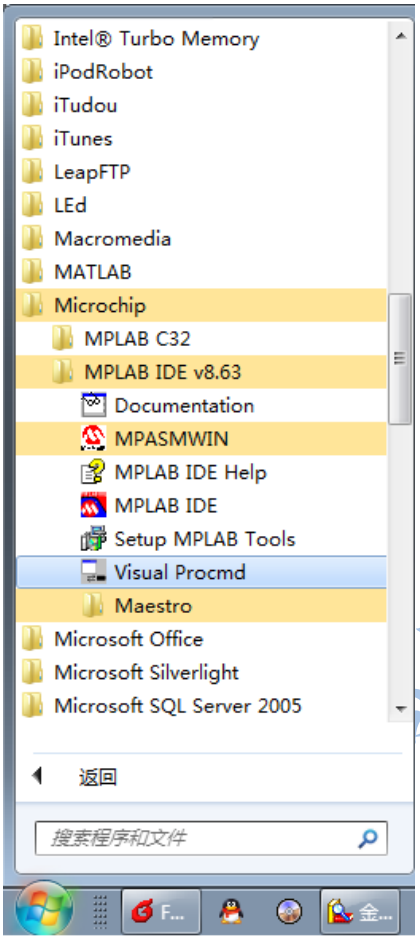
5.4 实验内容

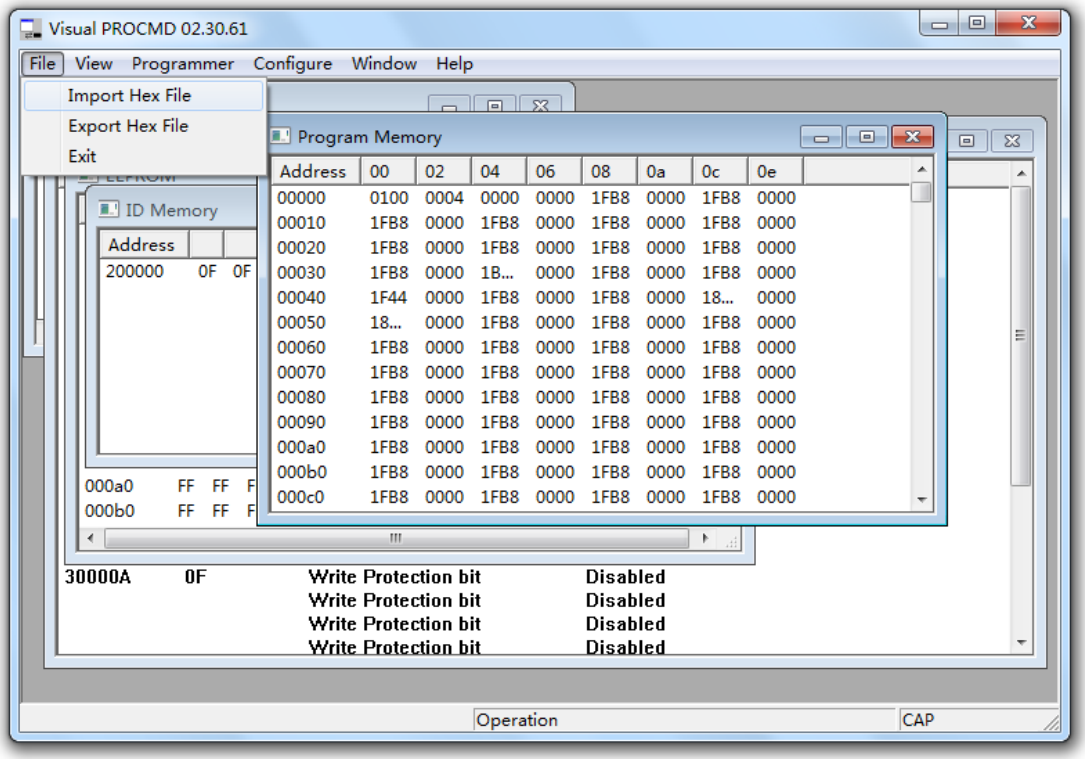
5.4.1 理解编译过程

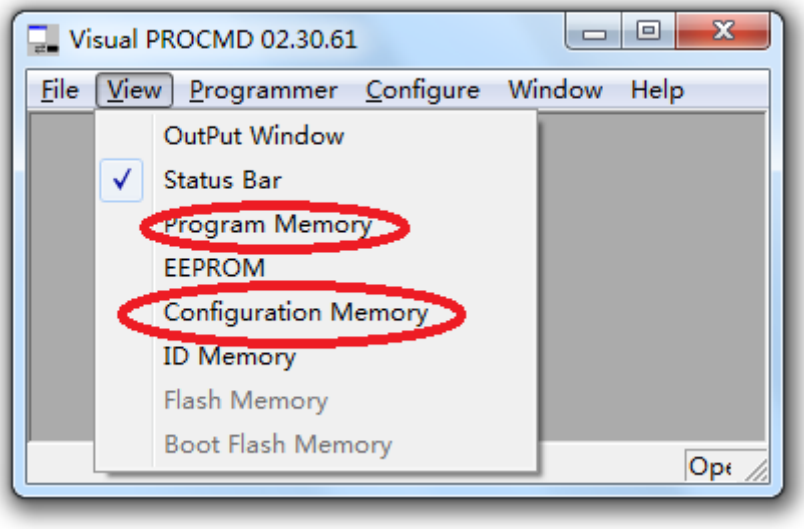
（1） 打开“example5_1.mcw”重新编译，根据编译时候看到的信息（如下图所示）分析生成的 COFF 文件中有哪些段？这些段分别存储到 dsPIC 片内存储器的哪些部分？



（2） 使用 Visual PROCMD 查看“example5_1.mcw”生成的 HEX 文件（example5_1.hex）的 Config memory 和 Program memory。Visual PROCMD 可在 Microchip 主菜单项中找到如下图。







（3） 使用文本编辑器（如 EditPlus、UltraEdit 或记事本程序）打开“example5_1.mcw”生成的 MAP 文件（example5_1.map），分析 HEX 文件烧写到芯片中后，“__task1”函数对应的机器码在 Program memory 中的具体存放位置（起始地址和结束地址）并记录。

5.4.2 顺序执行不同的子任务

（4） 请参考“example5_1.mcw”框架新建项目“lab5_1.mcw”，为项目编写三个子函数。三个子函数分别对应三项任务，功能如下。在主函数中顺序执行写好的三个子函数。

a) 函数 1（对应任务 1）：

i. 共 3 次循环

ii. 每个循环提内：点亮 D13 灭 D14 两秒钟、灭 D13 亮 D14 两秒钟

b) 函数 2（对应任务 2）：循环点亮 D6-D10；3 次循环

i. 共 3 次循环

ii. 每个循环提内：点亮 D1 灭 D2 两秒钟、灭 D1 亮 D2 两秒钟

c) 函数 3（对应任务 3）：循环点亮 D1-D5；3 次循环

i. 共 3 次循环

ii. 每个循环提内：点亮 D9 灭 D10 两秒钟、灭 D9 亮 D10 两秒钟

5.4.3 可控次序的逐个调度子任务

（5） 请参考“example5_1.mcw”中“sample5_1_main.c”类似的定义方式为三个任务安排不同的优先级。并按照优先级递增的顺序逐个调用子函数。

5.4.4 子任务间带简单同步的调度【*选做】

（6） 修改任务对应的函数

d) 函数 1（对应任务 1）：

i. 共 3 次循环

ii. 每个循环提内：点亮 D13 灭 D14 两秒钟、灭 D13 亮 D14 两秒钟

iii. 循环执行 1 次后查看是否有比自己优先级高的任务，如果没有继续循环，如果有责暂停，记录下已经循环的次数，返回主程序

e) 函数 2（对应任务 2）：循环点亮 D6-D10；3 次循环

i. 共 3 次循环

ii. 每个循环提内：点亮 D1 灭 D2 两秒钟、灭 D1 亮 D2 两秒钟

iii. 循环执行 1 次后查看是否有比自己优先级高的任务，如果没有继续循环，如果有责暂停，记录下已经循环的次数，返回主程序

f) 函数 3（对应任务 3）：循环点亮 D1-D5；3 次循环

i. 共 3 次循环

ii. 每个循环提内：点亮 D9 灭 D10 两秒钟、灭 D9 亮 D10 两秒钟

iii. 循环执行 1 次后查看是否有比自己优先级高的任务，如果没有继续循环，如果有责暂停，记录下已经循环的次数，返回主程序

5.4.5 FreeRTOS 示例测试效果再现

（7） 运行示例“RTOSDemo.mcw”，观察运行效果。分析“main.c”中如下三行的作用。

```
xTaskCreate( portControlTest1, ..... );
xTaskCreate( portControlTest2, ..... );
xTaskCreate( vLCDTask1, ..... );
```

5.4.6 FreeRTOS 核心代码阅读

（8） 阅读示例“RTOSDemo.mcw”中“\DSPic30F4011_RTOS\src\list.c”文件，根据在《C 语言》或《数据结构》课程中的知识分析其基本功能。

（9） 阅读示例“RTOSDemo.mcw”中“\DSPic30F4011_RTOS\src\tasks.c”文件，根据本课程关于 OS 的知识分析其基本功能。

（10） 阅读示例“RTOSDemo.mcw”中“\DSPic30F4011_RTOS\src\queue.c”文件，根据

本课程关于 OS 的知识分析其基本功能。

5.4.7 FreeRTOS 中的任务状态信息

（11） 阅读示例“RTOSDemo.mcw”中“\DSPic30F4011_RTOS\src\tasks.c”文件中对任务控制块（TCB,Task Control Block）数据结构的定义（如下代码所示），TCB 中定义了哪些信息？

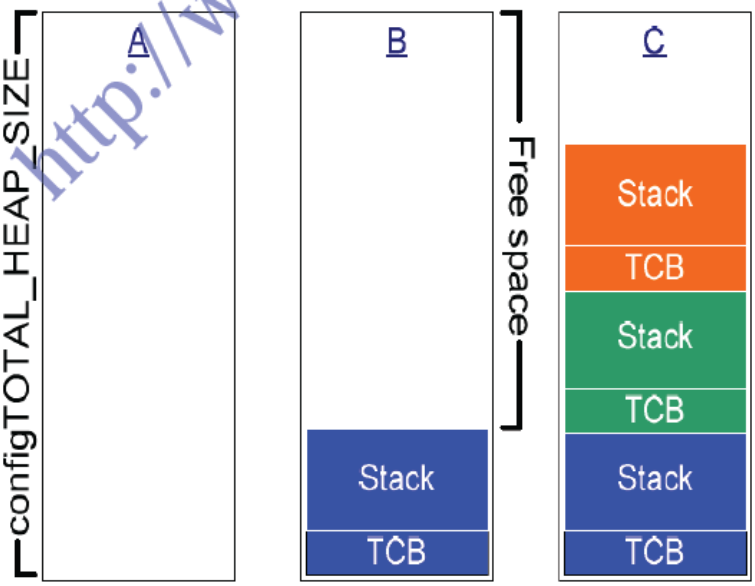
```
typedef struct tskTaskControlBlock
{
……
} tskTCB;
```

5.4.8 FreeRTOS 中的堆和栈

（12） 示例“RTOSDemo.mcw”中“\DSPic30F4011_RTOS\include\FreeRTOSConfig.h”文件中声明了 heap 的大小（如下所示），结合课堂知识分析 heap 和 stack 的区别。

```
#define configTOTAL_HEAP_SIZE                ( ( size_t ) 1540 )
```

（13） 下图显示了在创建任务后 configTOTAL_HEAP_SIZE 大小的 memory 空间的情况。



图中

A 表示数组在没有任何任务创建时的情形，这里整个数据是空的。

B 表示数组在创建了一个任务后的情形。

C 表示数组在创建了三个任务后的情形。

请解释 heap、stack 和 TCB 的联系。

5.5 思考题

（14） 单片机的 CPU 设计中一般提供哪些机制可供用户同时运行多项任务？

（15） 在单片机程序中管理多个不同程序的难点是什么？

（16） 操作系统在调度多项用户子任务的时候，如果子任务之间需要进行同步（即需要协调执行），至少应该为每个子任务保存哪些信息？