

DSP开发实验基础实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| **作 者:** | 马子轩 | **学 号：** | 9161040G0826 |
| **学 院:** | 电子工程与光电技术学院 | | |
| **专业(方向):** | 电子信息工程 | | |
| **题 目:** | DSP开发实验基础 | | |

**指导者： 李彧晟**

**评阅者：**

2019年 11 月

目 次

[1 实验目的 2](#_Toc24665)

[2 实验仪器 2](#_Toc1190)

[3 实验内容 3](#_Toc405)

[4 实验准备 3](#_Toc28976)

[5 实验步骤 6](#_Toc17243)

[6 实验结果 6](#_Toc11611)

[6.1 试验箱测试 6](#_Toc17284)

[6.2 入口地址 7](#_Toc25465)

[6.3 输入与输出波形 7](#_Toc32638)

[6.4 查看.map文件 8](#_Toc3637)

[6.5 查看.cmd文件 9](#_Toc30313)

[7 实验感悟 10](#_Toc2737)

[7.1 实验中遇到的问题与解决方案 10](#_Toc2399)

[7.1.1 无法从在工程中添加文件 10](#_Toc13359)

[7.1.2 图形工具画出的波形错误 10](#_Toc29411)

[7.2 实验的收获与感受 10](#_Toc7754)

# 

# 实验目的

1. 了解DSP开发系统的基本配置

2. 熟悉DSP集成开发环境（CCS）

3. 掌握C语言开发的基本流程

4. 熟悉代码调试的基本方法

# 实验仪器

计算机，TMS320F28335 DSP教学实验箱，XDS510 USB仿真器



图 2-1 熟悉实验仪器设备框图



图 2-2 实验一设备框图

# 实验内容

建立工程，对工程进行编译、链接，载入可执行程序，在DSP硬件平台上进行实时调试，利用代码调试工具，查看程序运行结果。

# 实验准备

TI的CCS 5集成开发环境，不仅支持汇编的编译、链接，还支持对C/C++汇编、编译、链接以及优化。同时强大的IDE开发环境也为代码的调试提供了强大的功能支持，已经成为TI 各DSP系列的程序设计、制作、调试、优化的主流工具。

TMS320F283x软件开发流程如下图所示。

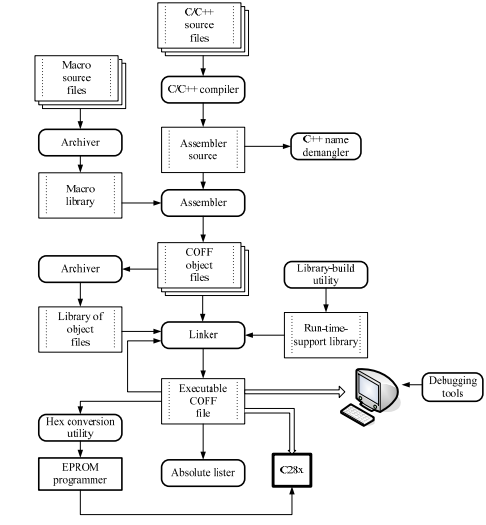


图 4-1 软件开发流程图

下面简单介绍各主要模块功能。

• C/C++ Compiler（C/C++编译器）

C/C++编译器把C/C++程序代码编译为基于DSP汇编指令集的汇编代码。这种转换并非一一对应，甚至会产生冗余的汇编代码，在某些场合需要使用优化器（Optimizer）来提高转换的效率，使得汇编代码长度尽可能的短小，程序所使用的资源尽可能的少。优化器是编译器的一部分。编程效率与编译器直接相关。

• Assembler（汇编器）

汇编器负责将汇编语言代码转换为符合公共目标格式（COFF）的机器语言，这种转换是一一对应的，每一条汇编指令都对应了唯一的机器代码。源文件中还包括汇编指令、伪指令和宏指令。这里的汇编代码包括了由C/C++编译器生成的汇编代码和直接编写的汇编代码。

• Linker（链接器）

链接器负责把可重定位的多个目标文件和目标库文件转换为一个DSP可执行程序，其中包含程序的机器代码、数据以及其他用来链接和加载程序所需要的信息。链接器必须依赖配置命令文件（CMD）的指令，实现对目标文件中各段的定位。

• Run-time-support library（运行支持库）

对于用C/C++语言中编写DSP程序中的某些功能（例如存储器的寻址定位、字符串转换等）并不属于C/C++语言所能描述对象，包含在C/C++编译器中的运行支持库却可以很好的支持这些算法的标准ANSI/ISO C函数描述。函数运行支持库包含有ANSI/ISO C的标准运行支持库函数、编译器功能函数、浮点算术函数和系统初始化子程序（这些函数都集成在汇编源文件rts.src中）。当对C/C++编写的DSP程序进行链接时，必须根据不同型号的DSP芯片添加相应的运行支持库到工程中。除此之外，在使用运行支持库中的函数时，必须在程序起始处用include语句包含相应的头文件（如使用数学运算sin、cos时，必须包含math.h）。而采用汇编语言编写程序时，却不需要这个运行支持库。因此C 语言编写的DSP 程序链接后，会产生大量的“冗余”汇编程序。

由此可见，用C/C++语言来开发DSP 程序，一般在工程中必须包含以下文件：

•.c或者.cpp：C 或C++程序，是主程序或函数，用于描述用户特定的算法功能；

•.cmd：配置命令文件，用于对编译生成的COFF 格式目标文件（.obj）定位，安排各段的物理存储空间；

•.lib：运行支持库文件，不同芯片有不同的运行支持库，必须根据具体芯片加以选择，例如TMS320F283x 的运行支持库文件命为rts2800\_fpu32.lib。（后缀fpu32 含义是支持32 位浮点运算）。

至于头文件（.h），只有当使用了运行支持库中相应的函数时，才需要在C 文件的主程序中用include语句指定相应的头文件（math.h、stdlib.h、float.h 等）。具体内容参见TI 公司的TMS320C28x Optimizing C/C++ Compiler User's Guide。

其次用户自定义函数、寄存器地址、常量定义等信息也可以编制到头文件中，使用时也同样需要在C 主程序中指定。

例如本实验中，需要的文件：

• main.c：C 语言主程序。

• 28335\_RAM\_lnk.cmd，DSP2833x\_Headers\_nonBIOS.cmd：配置命令文件。

• rts2800\_fpu32.lib：运行支持库。

• Sine.h：常量定义头文件。

• FPU.h：浮点运算库头文件。

• sine.dat：实验中需要的数据文件。

对于使用CCS以工程为单位进行DSP程序的项目开发时，一般为每个工程建立一个独立的目录，将项目中所需要的文件都存放在该工程目录下，便于程序的管理。rts2800\_fpu32.lib在TI的安装目录…\TI\c2000\cgtools\lib中可以找到。

# 实验步骤



# 实验结果

## 试验箱测试

（1）打开示波器，信号发生器，调节信号源输出幅度控制在±0.5V以内。

（2）将信号源输出端接至实验箱curINPUT1，将实验箱OUT2分别连接至示波器。

（3）打开实验箱电源，检查电源指示灯是否点亮。

（4）点击桌面CCS软件，进入集成开发环境。

（5）将学生实验资料中的LAB11工程导入到CCS工作环境中，并运行程序。

（6）在示波器上观察到输出波形如下两图所示。

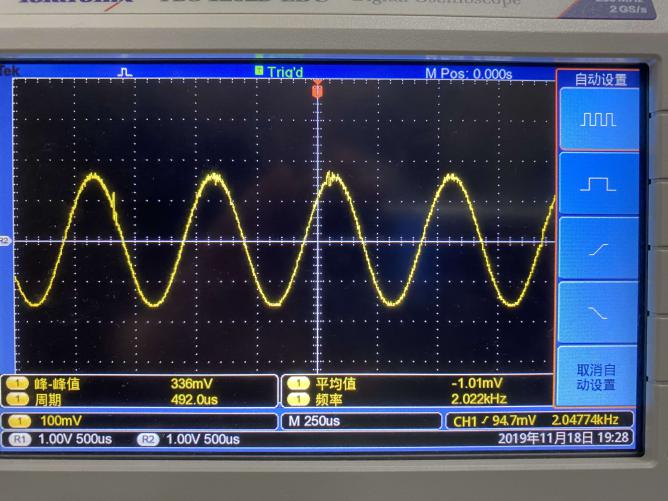
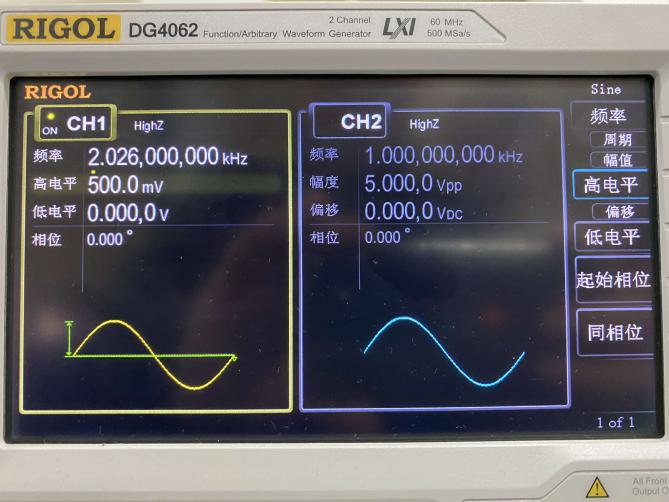


图 6-1 （a）信号发生器输出峰峰值500mv，正偏置250mv的正弦波；（b）在示波器上看到输出的波形

## 入口地址

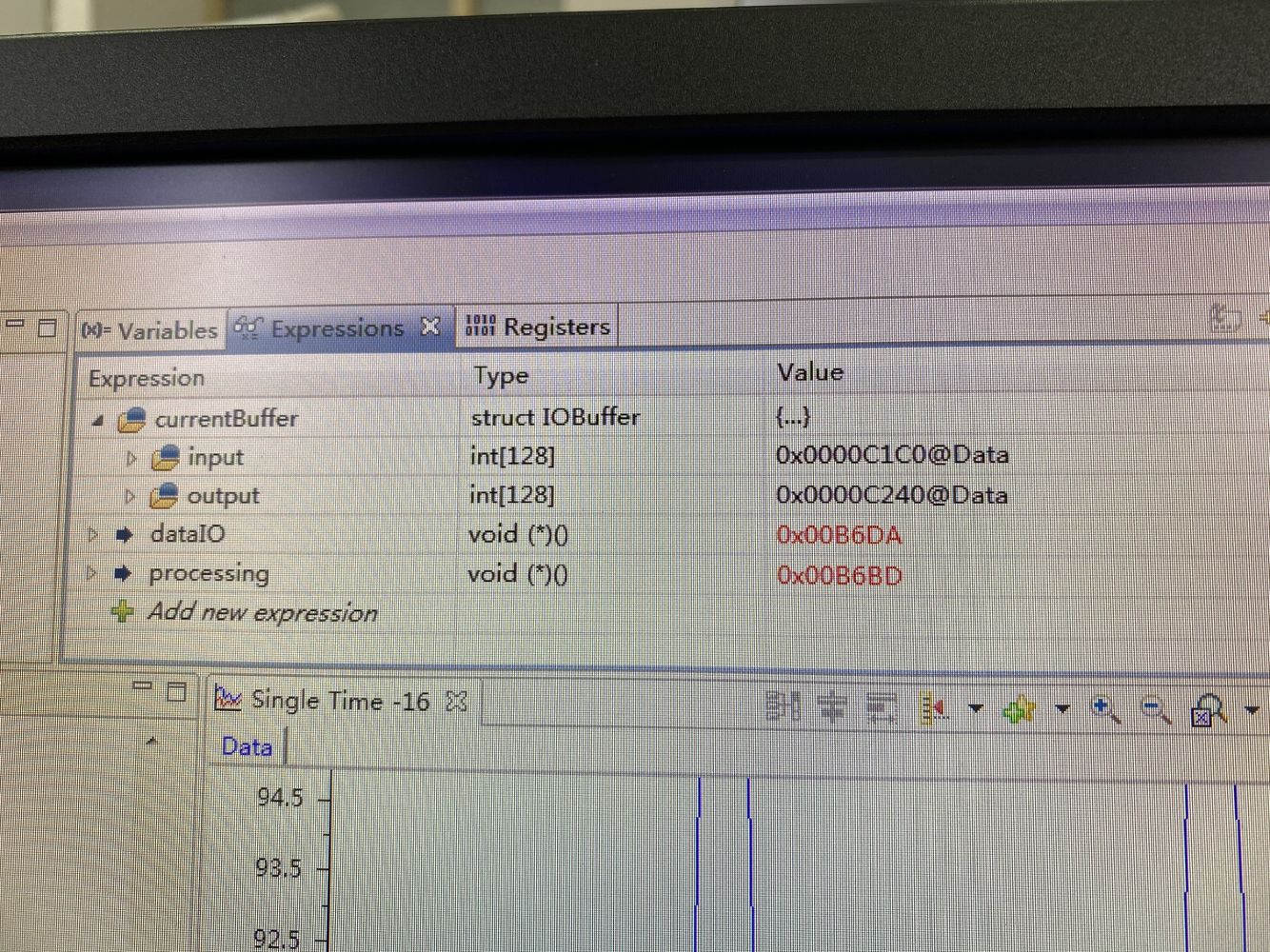


图 6-2入口地址

为更清楚展示实验结果，将实验要求的入口地址以表格形式列在下面：

|  |  |
| --- | --- |
| **名称** | **地址** |
| dataIO()子程序 | 0x00B6DA |
| Procession()子程序 | 0x00B6BD |
| currentBuffer.input | 0x0000C1C0 |
| currentBuffer.output | 0x0000C240 |

## 输入与输出波形

记录增益控制处理后，以图形方式显示数据空间currentBuffer.input和currentBuffer.output缓冲存储器中的波形，如图6-3所示。

【注意】，我们将增益由-4修改为-1，以至于形成了下面的这种反相效果。

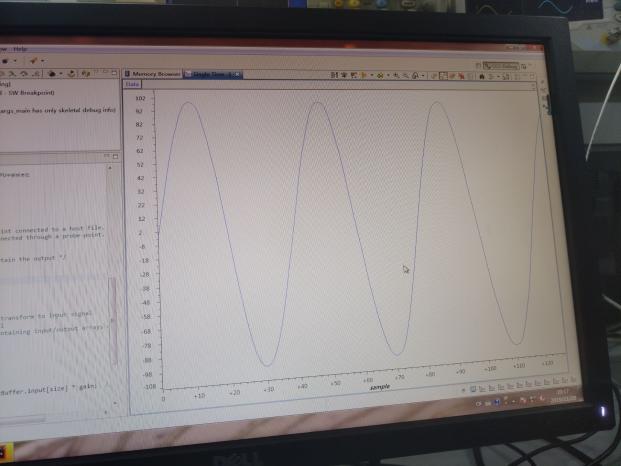
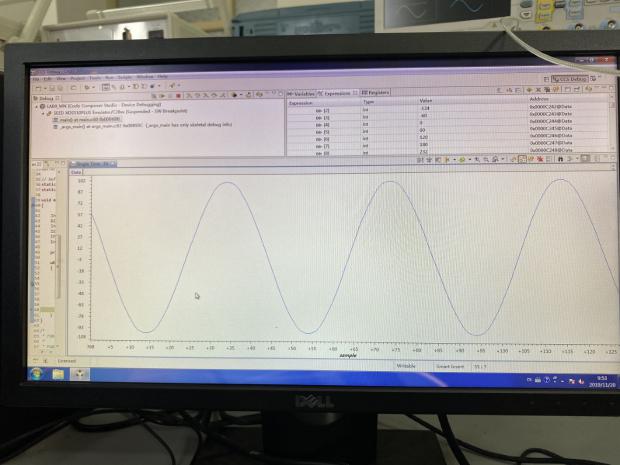


图 6-3 （a）currentBuffer.input示意图，（b）currentBuffer.output示意图

## 查看.map文件

打开工程的.map文件，查看所有的段在存储空间的地址、长度和含义，指出分别位于TMS320F28335的什么存储空间以及物理存储块名称，主程序中所用的变量分别属于什么段？

如下图所示，打开.map文件，可以看到如下内容。

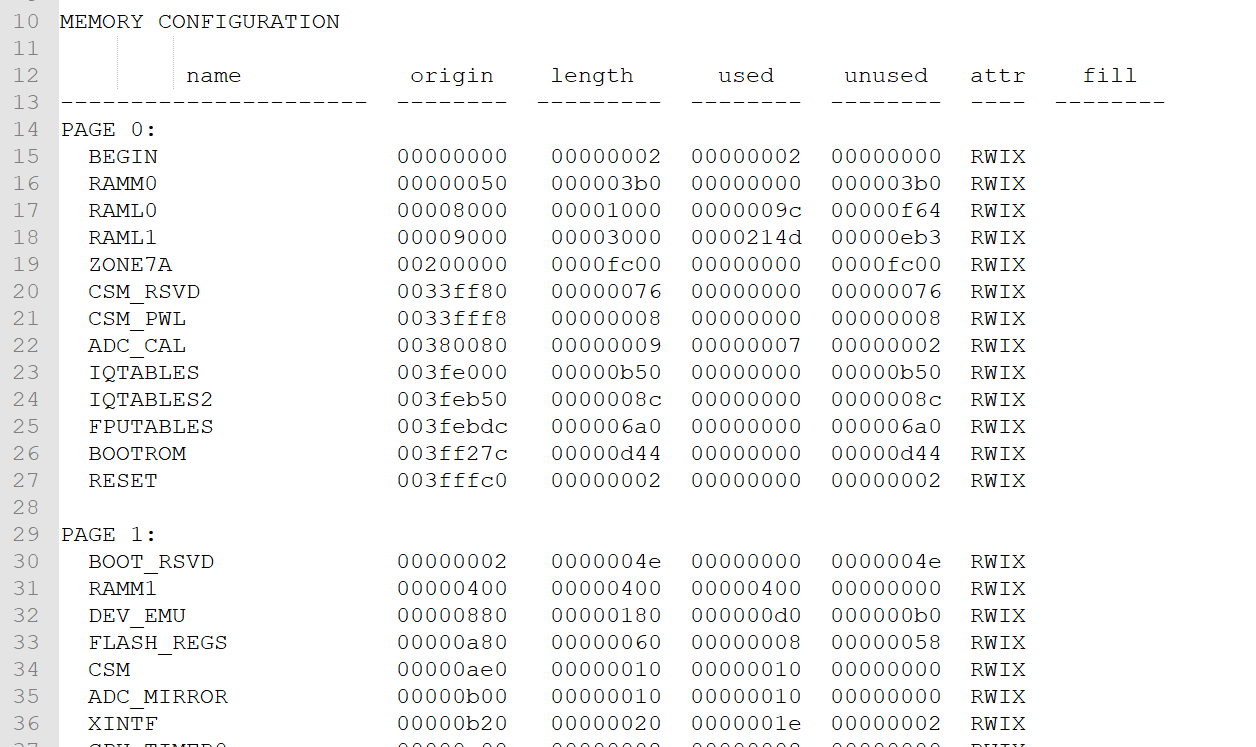


图 6-4 .map文件中的部分内容截图

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Section** | **page** | **origin** | **length** | **作用** | **物理存储块** |
| .pinit | 0 | 003f8090 | 00000000 | 调用全局对象构造函数的表 | FLASH |
| .cinit | 0 | 003f8b6c | 00000146 | 变量和常数初始化表 | FLASH |
| .reset | 0 | 003fffc0 | 00000002 |  | RAMI |
| .sysmem | 1 | 00000000 | 00000400 | 为动态存储分配保留 的空间 | RAM |
| .stack | 1 | 00000400 | 00000400 | 分配系统堆栈空间 | Lower 64Kw RAM |
| .ebss | 1 | 00008000 | 000006c0 | 为使用大寄存器模式 时的全局变量和静态 变量预留的空间 | RAM |
| .econst | 1 | 000086c0 | 0000001c | 字符串常量和数据（不  包括 volatile） | FLASH |
| .cio | 1 | 00008700 | 00000120 | 用于stdio函数 | RAM |

表 1 map文件中各字段相关信息

## 查看.cmd文件

查看.cmd 命令文件，比较其与上述.map 中的映射关系。试图修改.cmd 文件，再次编译链接，查看配置命令与各段的映射关系。

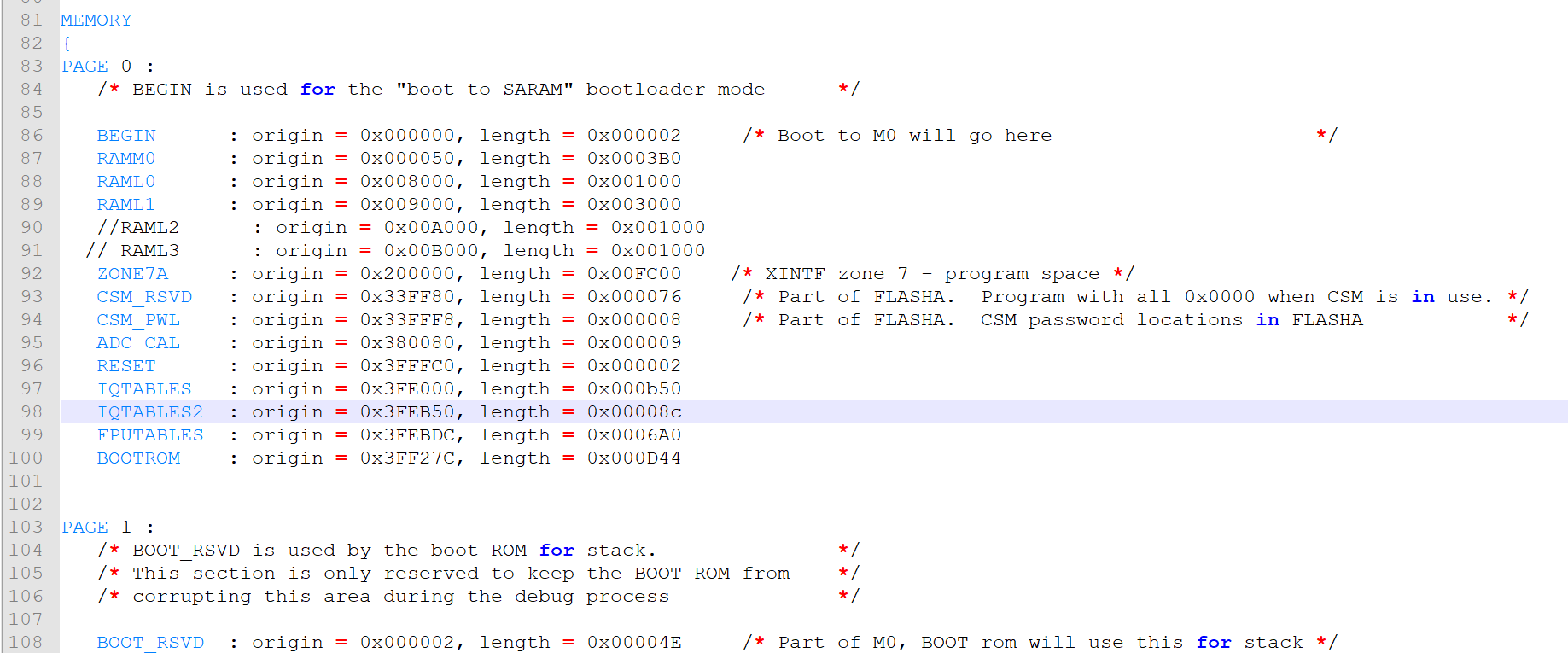


图 6-5 .cmd文件部分命令示意图

从上图和6-4图对比可看出，其与.map文件中的memory部分相对应。

修改了.cmd命令文件中的PROG起始地址为003f9000，重新编译链接后可以看到.map中的映射关系发生了改变。从下图中可以看到，.text的存储空间为PROG，所以在.map中可以看到.text的起始地址也同样发生了变化，同样的变化体现在.bss的存储地址上。

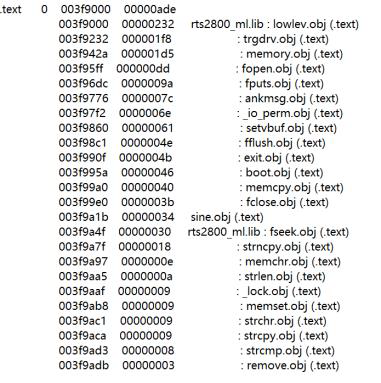
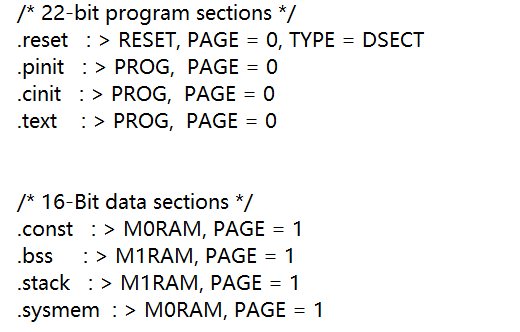


图 6-6 .cmd文件 地址 映射 &.map文件 部分截图

# 实验感悟

## 实验中遇到的问题与解决方案

### 无法从在工程中添加文件

为了测试实验箱完整性，需要添加外部文件，但是在添加时缺提示报错，后猜测由于路径中有中文名字导致无法添加，修改了文件路径后可以加入文件。

### 图形工具画出的波形错误

使用CCS中的图形工具，绘制出的图像波形前没有数据，波形后有杂乱的波形，经过研究发现是在绘制图象时将16位的数据误认为32位的数据，从而导致了图像错误，最终将图像数据选择为16位有符号数，即绘制出了正确的图像。

## 实验的收获与感受

这应该时第三次碰到这个实验箱了，但是前两次都没有自己对这个DSP芯片进行编程，所以没有深入了解DSP，在本次实验中，通过老师的讲解以及对于例程的阅读，对于使用CCS编程有了一个大致的了解。

之前有接触过STM32等一些MCU的程序编写，所以对于老师所给的一些例程的阅读无障碍，其实接触一个新的硬件或是软件都是大同小异的，难点在于整个开发环境的搭建，比如让实验箱可以成功的连接上PC机，可以让程序成功地下载到芯片上，这些基础工作完成后就可以通过编写一些基础程序快速地熟悉整一套设备。

除了基础环境的搭建之外，还需要懂得调试程序的方法，比如在本次实验中使用的图形绘制的方法，在写完程序之后可以现在PC机上做一个仿真，没问题后再下载到芯片上，可以防止出现一些意外。

在本次实验中，我还通过查看各个子程序的地址，第一次了解到程序从编译到链接的整个过程，之前在编程从来没有考虑过这个问题，从更底层了解整个程序的运行机理，可以让自己在编写程序时可以以一种更贴合机器计算的思路编写算法，从而提高整个算法运行的效率。