DSP 应用技术

实验报告〈一〉

题目：DSP 应用技术实验报告

院系：电子工程与光电技术学院

姓名（学号）：

指导教师：李彧晟

实验日期：2015 年 12 月 1 号

实验一 DSP 开发基础实验

一、 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。

二、实验仪器

计算机，C2000DSP 教学实验箱，XDS510USB 仿真器

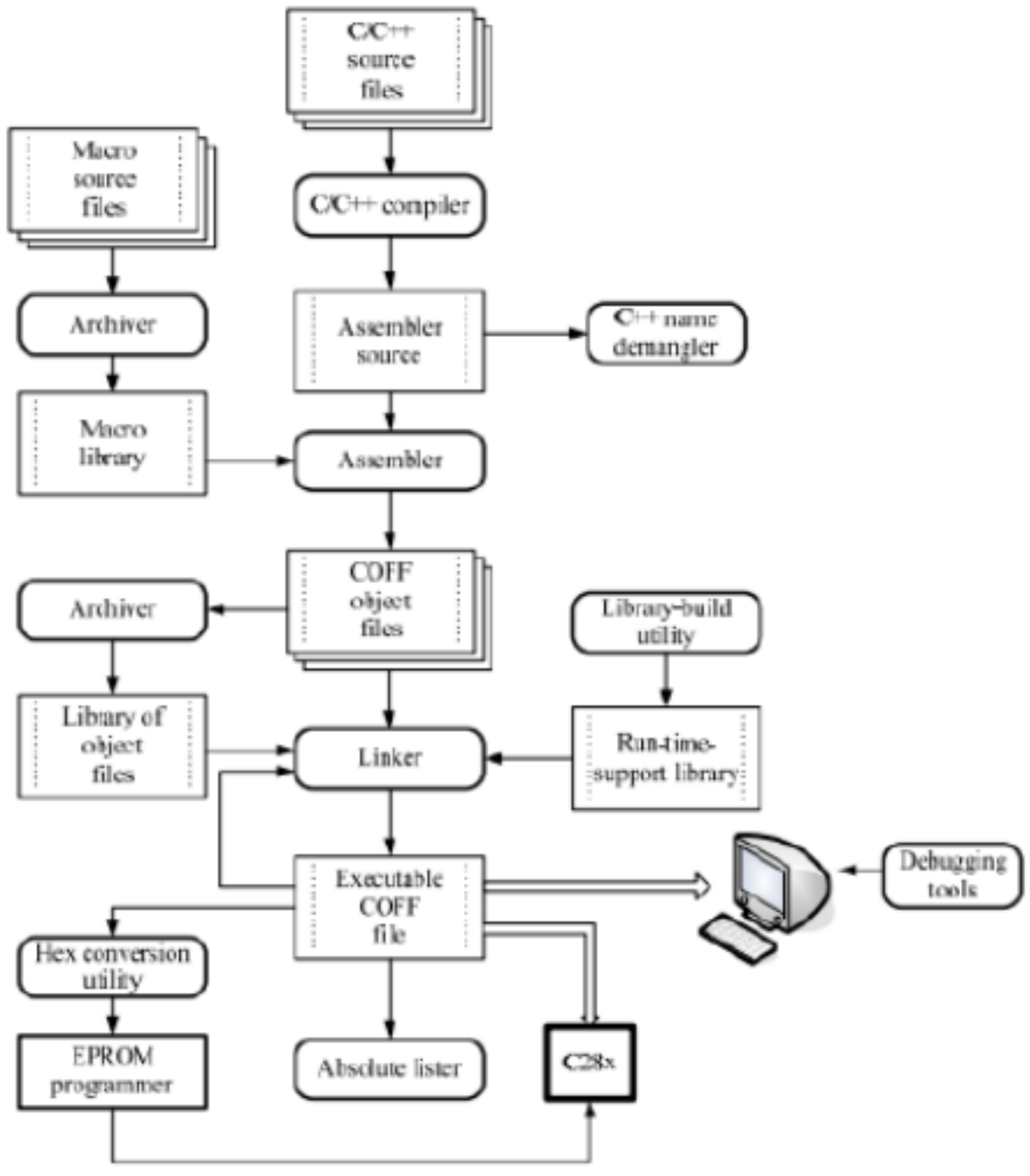
三、实验内容

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。

四、实验准备

CCS 2(C2000)这一集成开发环境,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。

TMS320C28x 软件开发流程如下图所示：



图一 TMS320C28x 软件开发流程

下面简单介绍各主要模块功能：

（1） C/C++ Compiler(C/C++编译器)

C/C++编译器把 C/C++程序自动转换成 C28x 的汇编语言源程序。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。

（2） Assembler(汇编器)

汇编器负责将汇编源程序转换为符合公共目标格式(COFF)的机器目标代码,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。

（3） 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。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。

（4） Run-time-support library(运行支持库)

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)。

五、实验步骤

（1）设备检查
（2）启动集成开发环境
（3）新建工程
（4）添加工程文件
（5）查阅代码
（6）建立工程(Build 工程)
（7）加载程序
（8）程序的运行
（9）程序的调试

六、实验结果

1、实验箱测试

（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 INPUT1，将实验箱 OUT3、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 CCS2（C2000），进入 F2812 的集成开发环境。

（5）加载 test.out 程序，并运行（Run），查看数码管显示、LED 闪烁、示波器两路输出信号。

最终用 CCS2（C2000）运行测试程序，观察到数码管显示数字“12345678”，LED 灯不断闪烁，同时示波器上显示出由信号源传来的两路输出信号，实验结果是实验箱工作完全正常，无异常现象。

2、C 程序调试

（1）记录 dataIO()、processing()子程序的入口地址,记录 currentBuffer.input 和 currentBuffer.output 所在存储器地址。

Name	Value	Type	Radix
processing	0x003F81D8	function *	hex
(*processing)		none	hex
dataIO	0x003F81F5	function *	hex
(*dataIO)		none	hex

Watch Locals | Watch 1

图一 dataIO()、processing()变量观察窗口

Name	Value	Type	Radix
currentBuffer	{...}	struct IOBuffer	hex
input	0x00008480	int[128]	hex
output	0x00008500	int[128]	hex
currentBuffer.output	0x00008500	int[128]	hex

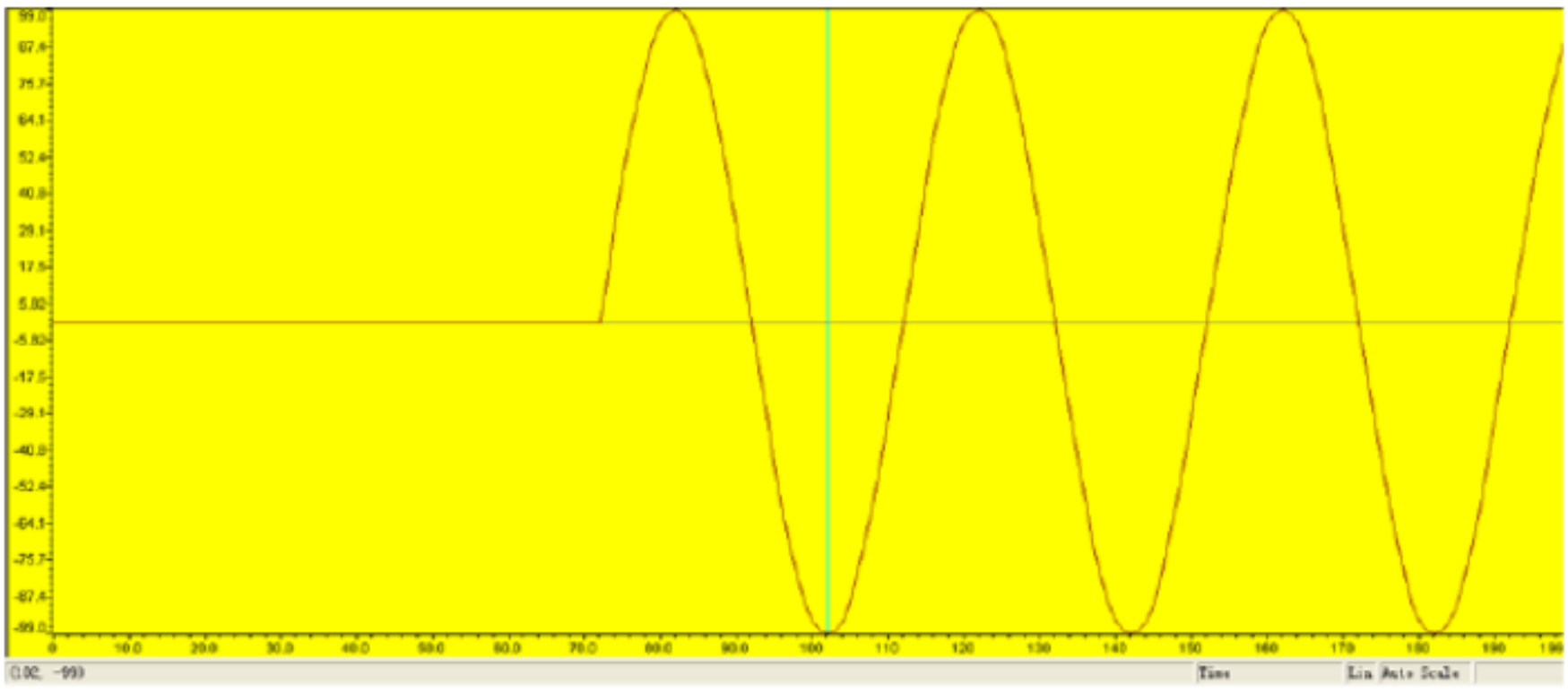
Watch Locals | Watch 1

图二 currentBuffer 变量观察窗口

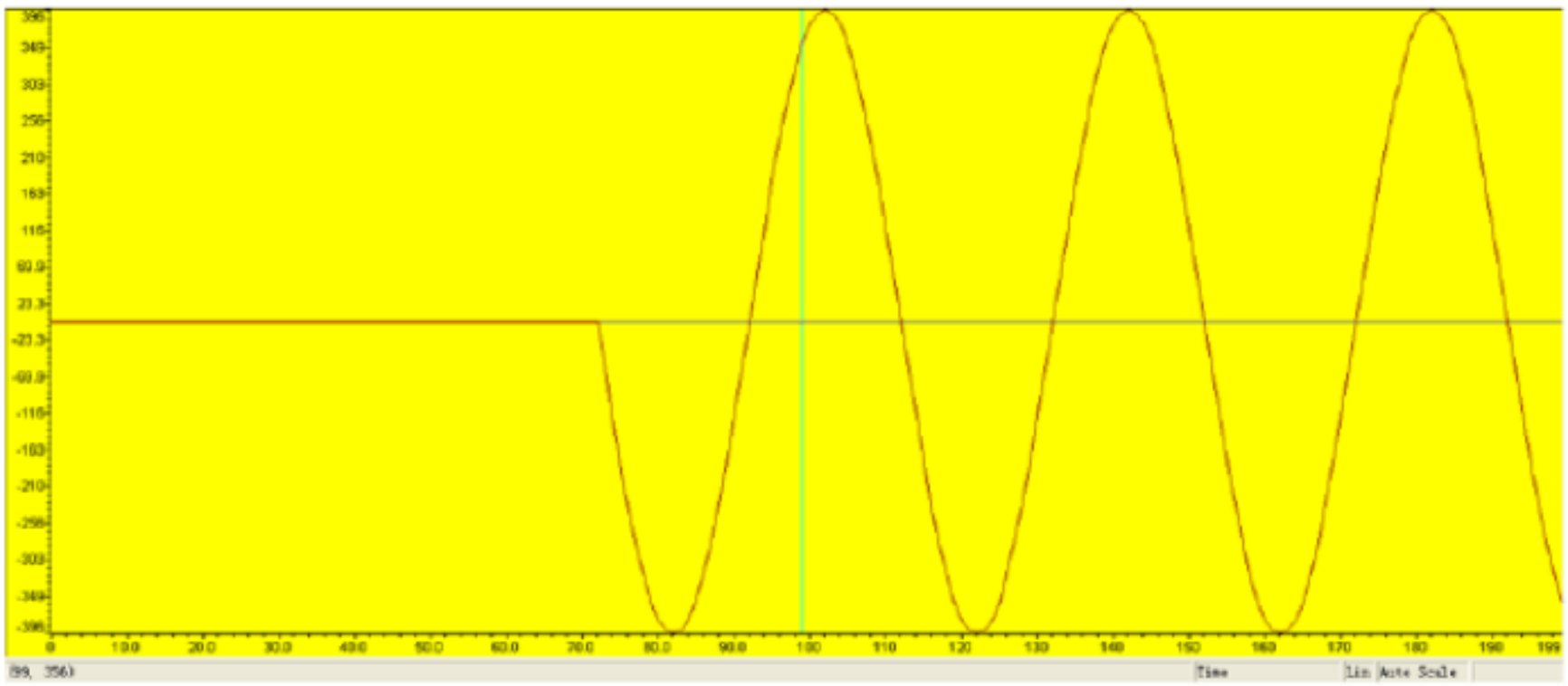
由 CCS 中的变量观察窗口可以清楚看到 dataIO()、processing()子程序的入口地址分别为 0x003F81F5 和 0x003F81D8， currentBuffer.input 和 currentBuffer.output 所在存储器地址分别为 0x00008480 和 0x00008500。

（2）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。

数据空间 currentBuffer.input 和 currentBuffer.output 缓冲存储器中的波形，通过 CCS 程序显示如下图所示：



图三 currentBuffer.input 缓冲存储器波形



图四 currentBuffer.output 缓冲存储器波形

（3）打开工程的.map 文件,查看.text、.data、.bss 段在存储空间的地址和长度,指出分别位于 TMS320F2812 的什么存储空间以及物理存储块名称。

```
.text       0    003f81c5   00000adb
                 003f81c5    00000031     sine.obj (.text)
                 003f81f6    00000046     rts2800_ml.lib : boot.obj (.text)
                 003f823c    0000004b                    : exit.obj (.text)
                 003f8287    0000009a                    : fputs.obj (.text)
                 003f8321    00000232                    : lowlev.obj (.text)
                 003f8553    0000000e                    : memchr.obj (.text)
                 003f8561    00000040                    : memcpy.obj (.text)
                 003f85a1    00000009                    : strchr.obj (.text)
                 003f85aa    00000008                    : strcmp.obj (.text)
                 003f85b2    0000000a                    : strlen.obj (.text)
                 003f85bc    00000018                    : strncpy.obj (.text)
                 003f85d4    000001f8                    : trgdrv.obj (.text)
                 003f87cc    0000006e                    : _io_perm.obj (.text)
                 003f883a    00000009                    : _lock.obj (.text)
                 003f8843    0000007c                    : ankmsg.obj (.text)
                 003f88bf    0000004e                    : fflush.obj (.text)
                 003f890d    00000030                    : fseek.obj (.text)
                 003f893d    00000061                    : setvbuf.obj (.text)
                 003f899e    00000009                    : strcpy.obj (.text)
                 003f89a7    000000dd                    : fopen.obj (.text)
                 003f8a84    000001d5                    : memory.obj (.text)
                 003f8c59    00000009                    : memset.obj (.text)
                 003f8c62    0000003b                    : fclose.obj (.text)
                 003f8c9d    00000003                    : remove.obj (.text)

.reset      0    003fffc0   00000002     DSECT
                 003fffc0    00000002       rts2800_ml.lib : boot.obj (.reset)

.const      1    00000000   00000000     UNINITIALIZED

.data       1    00000000   00000000     UNINITIALIZED

.sysmem     1    00000000   00000400     UNINITIALIZED

.bss        1    00000400   00000000     UNINITIALIZED

.stack      1    00000400   00000400     UNINITIALIZED

.esysmem    1    00008000   00000000     UNINITIALIZED
```

Section	page	origin	length	station
.text	0	003f81c5	00000adb	片外存储空间的数据存储空间
.data	1	00000000	00000000	内部存储空间数据存储空间的起始位
.bss	1	00000400	00000000	内部存储空间数据存储空间 00000040

（4）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。

由上图可以得到：

① PAGE 1：M1RAM（RW） origin=0x000500，length=0x400

将上面的 M1RAM（RW）的值改为 origin=0x000500 时，.bss 的起始位置变为如下所示：

.bss 1 00000500 00000000 UNINITIALIZED

即修正.bss 的存储空间应该在 M1RAM（RW）内。

② 由.cmd 命令文件中的语句：

PAGE0：PROG（R）： origin=0x3f8080，length=0x1f80

再由.text 0 003f81c5 00000adb 的地址及长度推断，修正.text 的存储空间为 PROG（R）。

七、实验总结

通过这次 DSP 实验我了解 DSP 开发系统的基本配置，并且熟悉 DSP 集成开发环境（CCS），掌握 C 语言开发的基本流程，熟悉代码调试的基本方法。通过实验箱的测试，我学会了建立、编译程序，并生成.out 文件，把程序加载到 DSP 芯片上。在 C 程序的调试中，我学会了查看子程序的入口地址和相关存储器的地址，并且通过 CCS 的图形显示功能显示缓冲存储器中的波形。接着我在助教的帮助下，学会查看.map 文件，并且能够找出其中.text、.data、.bss 段在存储空间的地址和长度。同时学会了查看.cmd 命令文件，并且比较与 map 中的映射关系。最后通过修改.cmd 文件，再次编译链接，观察配置命令与各段的映射关系。

总之，通过这次的 DSP 实验，让我对 DSP 开发中的软件和硬件有了大概的了解，任何事物的学习都是由浅入深，相信通过后期的学习和实验，自己在编程、程序调试和硬件测试方面的能力会进一步提升，并且能够独立地完成工程的建立、程序的建立、编译和调试。同时由于本次的实验是三人合作完成，因此通过这次实验，也加强了三人之间的默契程度和三人之间的合作能力。

通过这次的 DSP 实验我对 DSP 开发有了一定的认识，希望通过以后的 DSP 的实验，自己的能力也能够得到进一步提升。