南京理工大学

NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

电子工程与光电技术学院

实验报告

课程名称：DSP 应用技术

实验名称：DSP 开发基础实验

班　　级：9151040G02

姓　　名：傅　超

学　　号：9151040G0216

指导老师：李彧晟

2018 年 11 月 20 日

目　录

1 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。

2 实验仪器

计算机，C2000 DSP 教学实验箱，XDS510 USB 仿真器

3 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。

4 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 软件开发流程如下所示。

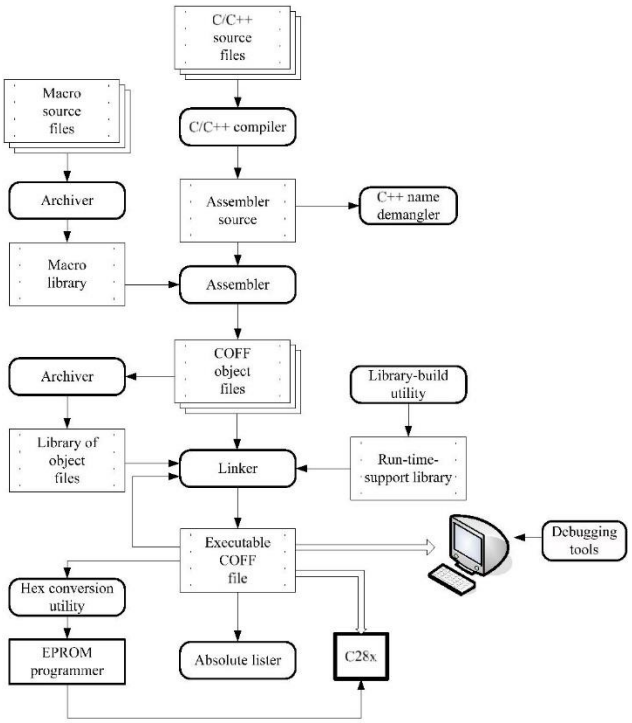


图 1：TMS320C28x 软件开发流程

• 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。

• 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。

• 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。

• 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）。

5 实验步骤

1、设备检查

检查仿真器、C2000 DSP 实验箱、计算机之间的连接是否正确，打开计算机和实验箱电源。

2、启动集成开发环境

点击桌面 CCS 2（C2000）快捷方式，启动 CCS。

3、新建工程

在主菜单中单击“Project → New”命令，弹出“Project Creation”对话框。在第一项 Project Name 中输入新建的工程名称，在第二项 Location 中选择工程所在目录，第三项 ProjectType 中选择输出文件格式“Executable（.out）”，在第四项 Target Family 中选择与当前 DSP 芯片吻合的 TMS320C28XX。

4、添加工程文件

在主菜单中单击“Project → Add Files to Project”命令，在弹出的对话框中依次选择当前工程目录下 sine.c、sinewave.cmd 以及 rts2800_ml.lib 文件，添加到当前工程中。

5、查阅代码

在 build 工程之前，先阅读一下源代码，明白各文件的内容。

6、建立工程（Build 工程）

建立工程（build）是指对 asm、c 源程序文件进行编译（Compile）、汇编（Assemble），并结合配置命令文件对工程进行链接（Link），输出可执行程序（.out）。

9. 加载程序

在主菜单下，选择“File → Load Program”，选中该工程的输出可执行.out 程序即可。

10. 程序的运行

对 C 语言编写的程序，载入程序以后，DSP 的程序计数器（PC）会自动指向_c_int00，这是 C 程序链接以后的入口地址（在反汇编窗口上有一个绿色箭头标识）。它是运行支持库自动在用户 C 程序之前添加的一段初始化程序。如果想让程序计数器指向 C 语言的起始处，在菜单中选择“Debug → Go Main”命令，让程序从主函数开始执行（在源程序窗口有一个黄色箭头标识）。

11. 程序的调试

添加结构体变量 currentBuffer 到变量观察窗口，观察 currentBuffer.output 和 currentBuffer.input 的地址以及数值。添加 dataIO( )到变量窗口，查看该子程序的入口地址。

全速运行程序，或者动画执行程序，查看以上存储空间的数值变化。

6 实验结果

6.1 实验箱测试

1、将信号源的输出信号接至 INPUT1，将实验箱的 OUTPUT3 接至示波器的通道一输入端口；

2、打开示波器和信号发生器，调节信号发生器的输出，控制幅度峰峰值在 1V 左右；

3、打开实验箱电源，检查实验箱电源指示灯是否正常指示；

4、通过仿真器将实验箱与 PC 机连在一起，点击 PC 机上的 CCS2 配置程序，配置完成后成功打开 F2812 集成开发环境；

5、创建工程，导入测试文件后重新编译生成 test.out 文件，加载到 DSP 中并全速运行，检查实验箱上数码管、LED、示波器波形等；

6、最终观察到示波器上的波形和信号发生器产生的波形一致，且实验箱上 LED 等不断闪烁，数码管显示“12345678”，由此判断实验箱正常工作，可以进行接下来的实验。

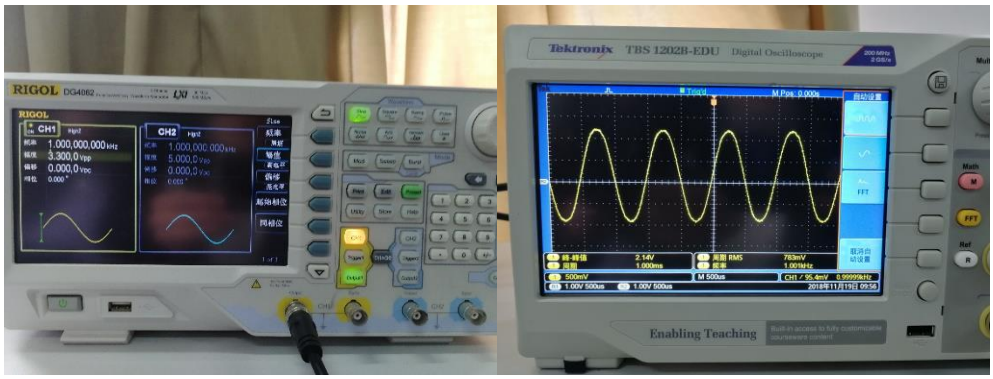
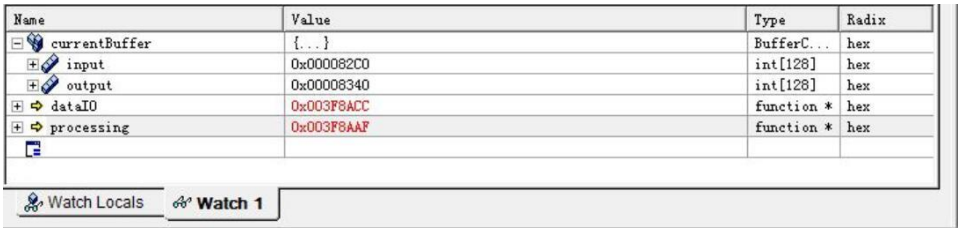


图 2：信号发生器配置图&示波器波形图

6.2 例程调试

1、记录 dataIO()、processing()子程序的入口地址



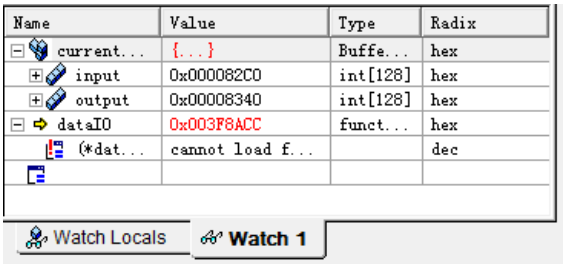
Name	Value	Type	Radix
currentBuffer	{...}	BufferC...	hex
input	0x000082C0	int[128]	hex
output	0x00008340	int[128]	hex
dataIO	0x003F8ACC	function *	hex
processing	0x003F8AAF	function *	hex

Watch Locals | Watch 1

图 3：dataIO()、processing()子程序的入口地址图

从图中可以看到，dataIO()子程序的入口地址为 0x003F8ACC，processing()子程序的入口地址为 0x003F8AAF。

2、记录 currentBuffer.input 和 currentBuffer.output 所在存储器地址



Name	Value	Type	Radix
current...	{...}	Buffe...	hex
input	0x000082C0	int[128]	hex
output	0x00008340	int[128]	hex
dataIO	0x003F8ACC	funct...	hex
(*dat...	cannot load f...		dec

Watch Locals | Watch 1

图 4：currentBuffer.input 和 currentBuffer.output 所在存储器地址图

从图中可以看到，currentBuffer.input 的地址为 0x000082C0，currentBuffer.output 的地址为 0x00008340。

3、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。

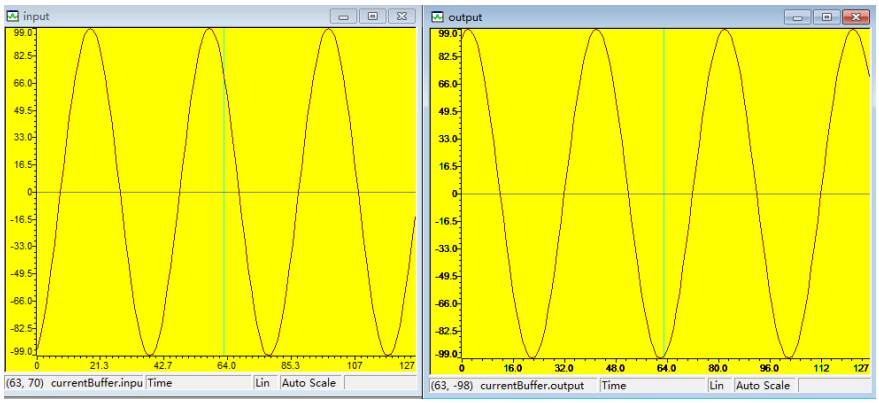


图 5：缓存器数据波形图

通过调用 CCS 中的图形显示方式，将两个数据空间中的数据进行了绘制，发现两者的波形一致，但是存在延迟，这是由于 DSP 内部计算导致的，所以这符合正常现象。

4、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   003f8080   00000adc
             003f8080   00000232   rts2800_ml.lib : lowlev.obj (.text)
             003f82b2   000001f8                  : trgdrv.obj (.text)
             003f84aa   000001d5                  : memory.obj (.text)
             003f867f   000000dd                  : fopen.obj (.text)
             003f875c   0000009a                  : fputs.obj (.text)
             003f87f6   0000007c                  : ankmsg.obj (.text)
             003f8872   0000006e                  : _io_perm.obj (.text)
             003f88e0   00000061                  : setvbuf.obj (.text)
             003f8941   0000004e                  : fflush.obj (.text)
             003f898f   0000004b                  : exit.obj (.text)
             003f89da   00000046                  : boot.obj (.text)
             003f8a20   00000040                  : memcpy.obj (.text)
             003f8a60   0000003b                  : fclose.obj (.text)
             003f8a9b   00000032   sine.obj (.text)
             003f8acd   00000030   rts2800_ml.lib : fseek.obj (.text)
             003f8afd   00000018                  : strncpy.obj (.text)
             003f8b15   0000000e                  : memchr.obj (.text)
             003f8b23   0000000a                  : strlen.obj (.text)
             003f8b2d   00000009                  : _lock.obj (.text)
             003f8b36   00000009                  : memset.obj (.text)
             003f8b3f   00000009                  : strchr.obj (.text)
             003f8b48   00000009                  : strcpy.obj (.text)
             003f8b51   00000008                  : strcmp.obj (.text)
             003f8b59   00000003                  : remove.obj (.text)

.sysmem  1   00000000   00000400   UNINITIALIZED
             00000000   00000400   --HOLE--

.bss     1   00000400   00000000   UNINITIALIZED

.stack   1   00000400   00000400   UNINITIALIZED
             00000400   00000400   --HOLE--

.ebss    1   00008000   000006c0   UNINITIALIZED
             00008000   00000280   rts2800_ml.lib : defs.obj (.ebss)
             00008280   00000140   sine.obj (.ebss)
             000083c0   00000110   rts2800_ml.lib : lowlev.obj (.ebss)
             000084d0   00000008                  : memory.obj (.ebss)
             000084d8   00000004                  : _lock.obj (.ebss)
             000084dc   00000024   --HOLE--
             00008500   00000108                  : trgdrv.obj (.ebss)
             00008608   00000038   --HOLE--
             00008640   00000080                  : exit.obj (.ebss)

.econst  1   000086c0   0000001c
             000086c0   0000001a   sine.obj (.econst:.string)
             000086da   00000002   rts2800_ml.lib : fputs.obj (.econst)

.cio     1   00008700   00000120   UNINITIALIZED
             00008700   00000120   rts2800_ml.lib : ankmsg.obj (.cio)
```

图 6：.map 文件部分截图

Section	Page	Origin	Length	Station
.text	0	003f8080	00000adc	片外存储空间的数据存储空间
.bss	1	00000400	00000000	内部存储空间数据存储空间 00000400

```
     name          origin    length     used     unused   attr    fill
---------------------- --------  ---------  --------  --------  ----  --------
PAGE 0:
  BOOT              003f8000  00000080  00000000  00000080  R
  PROG              003f8080  00001f80  00000c22  0000135e  R
  RESET             003fffc0  00000002  00000000  00000002  R

PAGE 1:
  M0RAM             00000000  00000400  00000400  00000000  RW
  M1RAM             00000400  00000400  00000400  00000000  RW
  L0L1RAM           00008000  00002000  000007fc  00001804  RW
```

图 7：.cmd 文件部分截图

5. 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。

```
******************************************************************************
              TMS320C2000 Linker PC v5.2.1
******************************************************************************
>> Linked Mon Nov 19 11:31:02 2018

OUTPUT FILE NAME:   <./Debug/fc.out>
ENTRY POINT SYMBOL: "_c_int00"  address: 003f995a


MEMORY CONFIGURATION

         name            origin    length     used     unused   attr    fill
----------------------  --------  ---------  --------  --------  ----  --------
PAGE 0:
  BOOT                  003f8000  00000080  00000000  00000080  R
  PROG                  003f9000  00001f80  00000c22  0000135e  R
  RESET                 003fffc0  00000002  00000000  00000002  R

PAGE 1:
  M0RAM                 00000000  00000400  00000400  00000000  RW
  M1RAM                 00000400  00000400  00000400  00000000  RW
  L0L1RAM               00008000  00002000  000007fc  00001804  RW


SECTION ALLOCATION MAP

 output                                  attributes/
section   page    origin      length      input sections
--------  ----  ----------  ----------  ----------------
.pinit    0     003f9000    00000000    UNINITIALIZED

.text     0     003f9000    00000adc
                003f9000      00000232    rts2800_ml.lib : lowlev.obj (.text)
                003f9232      000001f8                   : trgdrv.obj (.text)
```

```
/*###########################################################################*/
/*   sinewave.cmd - Sample linker command file for F28xx devices             */
/*                                                                           */
/*   Description: This file is a sample F2812 linker command file that can  */
/*                be used for linking programs built with the TMS320C2000    */
/*                C Compiler. Use it as a guideline; you may want to change  */
/*                the allocation scheme according to the size of your        */
/*                program and the memory layout of your target system.       */
/*###########################################################################*/

MEMORY
{
   PAGE 0 : BOOT(R)      : origin = 0x3f8000, length = 0x80
   PAGE 0 : PROG(R)      : origin = 0x3f9000, length = 0x1f80
   PAGE 0 : RESET(R)     : origin = 0x3fffc0, length = 0x2


   PAGE 1 : M0RAM(RW)    : origin = 0x000000, length = 0x400
   PAGE 1 : M1RAM(RW)    : origin = 0x000800, length = 0x400
   PAGE 1 : L0L1RAM(RW)  : origin = 0x008000, length = 0x2000
}
```

图 8：修改的 map 截图&修改后的 cmd 文件截图

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。

```
/* 22-bit program sections */
.reset   : > RESET, PAGE = 0, TYPE = DSECT
.pinit   : > PROG,  PAGE = 0
.cinit   : > PROG,  PAGE = 0
.text    : > PROG,  PAGE = 0


/* 16-Bit data sections */
.const   : > M0RAM, PAGE = 1
.bss     : > M1RAM, PAGE = 1
.stack   : > M1RAM, PAGE = 1
.sysmem  : > M0RAM, PAGE = 1
```

```
.text     0     003f9000    00000ade
                003f9000      00000232    rts2800_ml.lib : lowlev.obj (.text)
                003f9232      000001f8                   : trgdrv.obj (.text)
                003f942a      000001d5                   : memory.obj (.text)
                003f95ff      000000dd                   : fopen.obj (.text)
                003f96dc      0000009a                   : fputs.obj (.text)
                003f9776      0000007c                   : ankmsg.obj (.text)
                003f97f2      0000006e                   : _io_perm.obj (.text)
                003f9860      00000061                   : setvbuf.obj (.text)
                003f98c1      0000004e                   : fflush.obj (.text)
                003f990f      0000004b                   : exit.obj (.text)
                003f995a      00000046                   : boot.obj (.text)
                003f99a0      00000040                   : memcpy.obj (.text)
                003f99e0      0000003b                   : fclose.obj (.text)
                003f9a1b      00000034    sine.obj (.text)
                003f9a4f      00000030    rts2800_ml.lib : fseek.obj (.text)
                003f9a7f      00000018                   : strncpy.obj (.text)
                003f9a97      0000000e                   : memchr.obj (.text)
                003f9aa5      0000000a                   : strlen.obj (.text)
                003f9aaf      00000009                   : _lock.obj (.text)
                003f9ab8      00000009                   : memset.obj (.text)
                003f9ac1      00000009                   : strchr.obj (.text)
                003f9aca      00000009                   : strcpy.obj (.text)
                003f9ad3      00000008                   : strcmp.obj (.text)
                003f9adb      00000003                   : remove.obj (.text)
```

图 9：.cmd 文件地址映射&.map 文件部分截图

7 实验感悟

7.1 实验中遇到的问题与解决方案

7.1.1 JTAG 的连接口中有一根断掉的针

由于仿真器上的排插没有凸口，所以很容易被反插损坏实验箱，而 JTAG 中有几根针是没有实际用处的，所以直接在接口中插入断针可以防止仿真器接口被反插。

7.1.2 程序无法正常启动

在实际过程中发现双击程序无法正常启动，经研究后发现首先打开配置程序，配置完成后即可成功打开程序。

7.1.3 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。

7.1.4 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 位符号数，即绘制出了正确的图像。

7.2 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。