



浙江工业大学

实 验 一：熟悉 CCS 集成开发环境

指导教师：吴春

班 级：电气 1701 班

姓 名：黄林志

学 号：201706060301

浙江工业大学

实验一：熟悉 CCS 集成开发环境

一、实验目的

1. 掌握 CCS5.4 的安装和配置步骤过程（安装配置在附录）；
2. 了解 DSP 开发系统和计算机与目标系统的链接方法；
3. 了解 CCS5.4 软件的操作环境和基本功能，了解 TMS320C28335 的开发过程：
 - 1) 学习创建工程和管理工程的方法；
 - 2) 了解基本的编译和调试功能
 - 3) 学习使用观察窗口；
 - 4) 了解图形功能的使用。

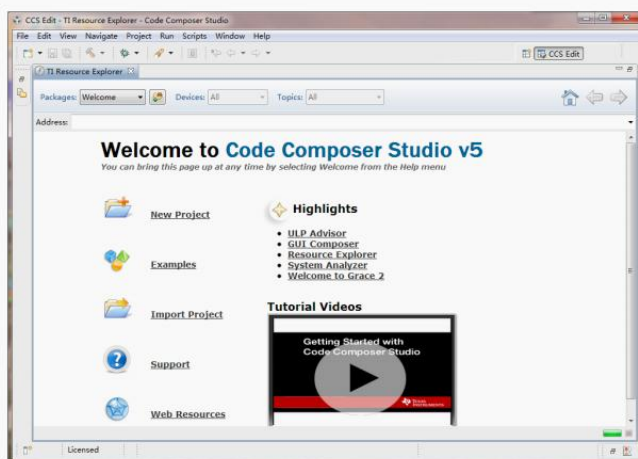
二、实验内容

1. 实验准备

- 1) 将 DSP 仿真器与计算机连接好；
- 2) 将 DSP 仿真器的 JTAG 插头与 SEED-DEC28335 单元的 J18 相连接；
- 3) 启动计算机，当计算机启动后，打开 SEED-DTK28335 的电源。观察 SEED-DTK_MBoard 单元的+5V，+3.3V，+15V，-15V 的电源指示灯及 SEED-DEC28335 的电源指示灯 D2 是否均亮；若有不亮，请断开电源，检查电源。

2. 新建一个 project 进行相关配置

- 1) 点击打开 CCS5.4，进入 CCS 画面，如图1-1：



- 2) CCS的开发都是以project为基础的，首先我们点击新建一个New project, 会出现如下图1-2 (a)所示的配置界面，添加项目名称、芯片、仿真器等：

- ① 输入新建工程名c_add；②选择TMS320F28335目标系统③点Finish

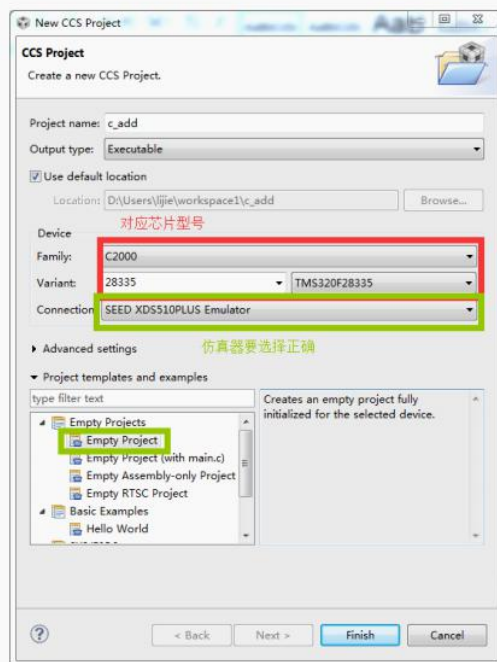


图1-2 (a)

至此，一个基本的project已经建立完成，进入如下图1-2 (b) 所示界面

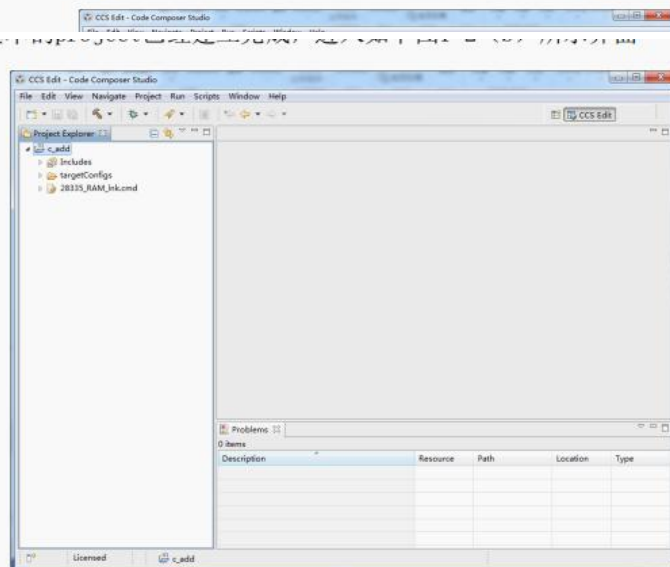


图1-2 (b)

- 3) 删除工程中自动生成的.cmd 文件(右键文件名->delete);
- 4) 右键工程选择 Add Files..., 添加工程所需文件;

- 5) 在弹出的对话框中添加源程序 add.c 添加到工程中;
- 6) 同样的方法可以添加文件 xxx.cmd、xxx.lib 等到工程中;
- 7) 设置编译与连接选项
 - a. 点击 Project 选择 properties
 - b. 在弹出的对话框如下图 1-3 所示中设置相应的编译参数, 一般情况下, 按默认值就可以;

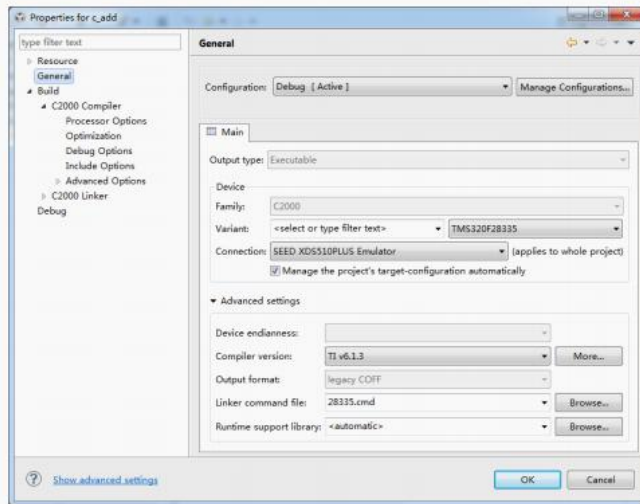


图1-3

- c. 点击 Basic Options 在弹出的对话框中选择连接的参数设置, 设置输出文件名 (可执行文件与空间分配文件), 堆栈的大小以及初始化的方式如下图 1-4 所示。

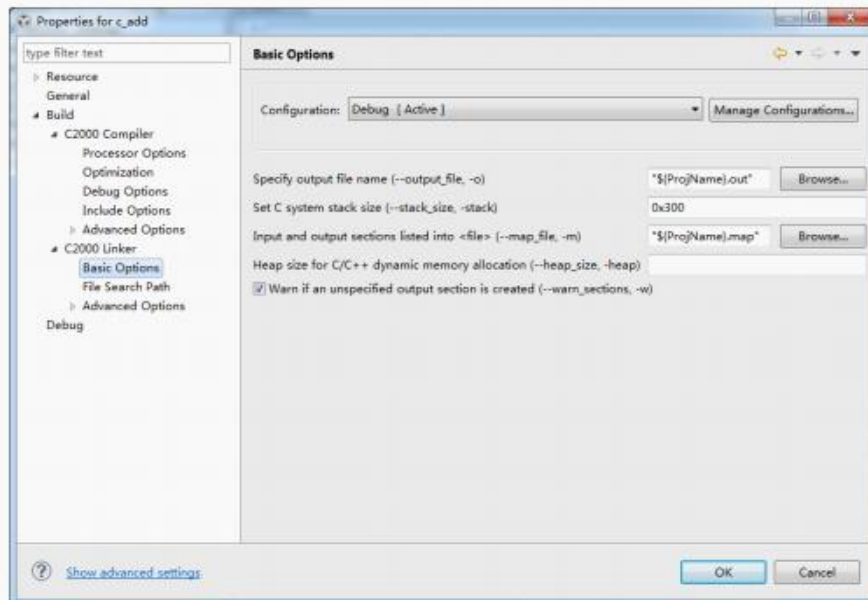


图1-4

- 8) 接下来进行相关仿真器的配置：工程名（c_add）> targetConfigs > TMS320F28335.CCXML；出现如下图1-5所示其实这里我们在上面新建工程时已经配置，此时可以修改和查看，若没有此文件则需要自己创建(参考下面第四部分，导入一个project工程)，根据实验是在硬件仿真还是软件仿真进行相应的修改)

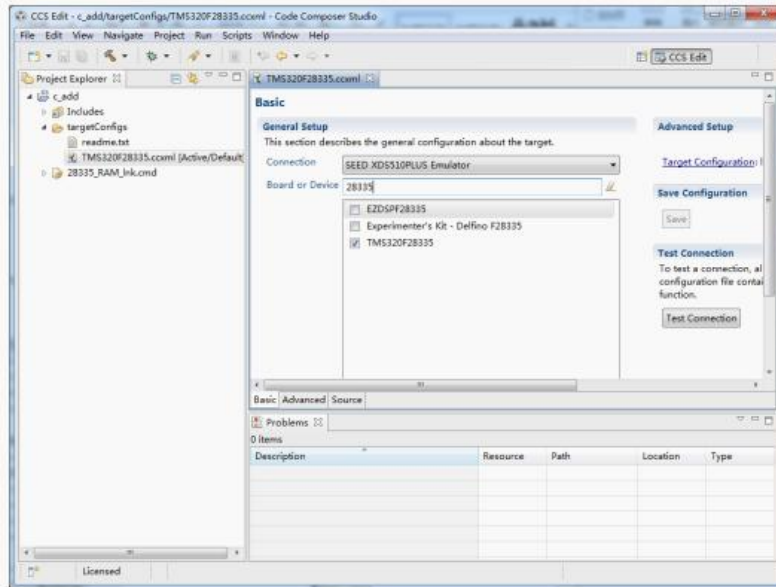


图1-5

- 9) ①选择正确的仿真器； ②选择TMS320F28335； ③点Save； ④确保pc与开发板正确连接⑤点test connection，如果出现图1-6所示，打印出仿真器相关信息，则说明仿真器配置正确，至此就可以开始编程学习DSP（此处会在最后报错，但程序能够正确下载）

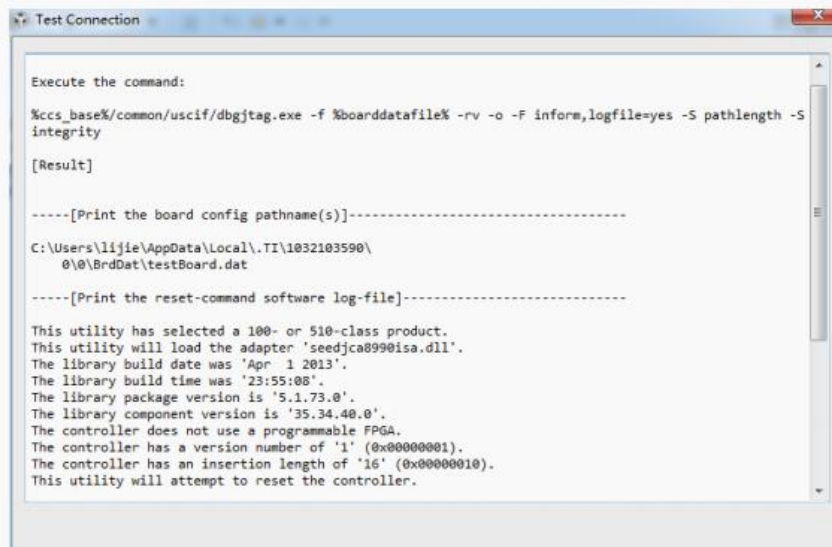


图1-6

3. 工程的编译与调试

- 1) 如图所示，点击Project→Build All,对整个工程进行编译（同理，如果要对整个工程进行重新编译，则先clean，然后在build all）在编译过程中，工程下面会有相对应的调试信息如下图1-7所示

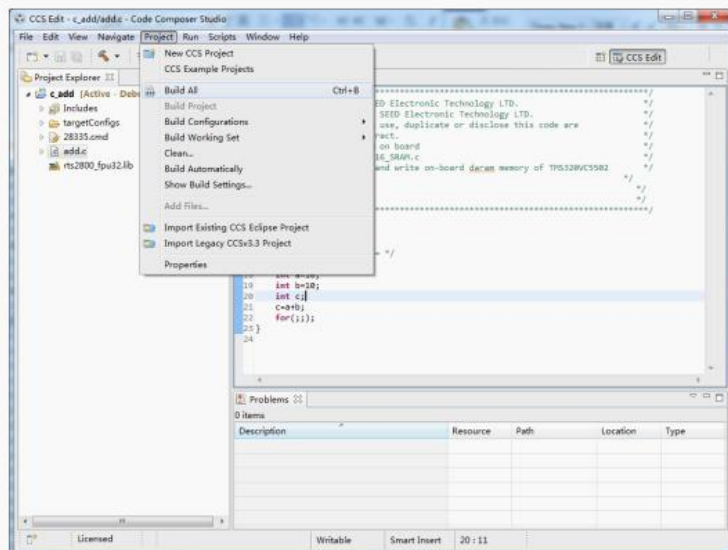


图1-7

- 2) 如果编译没有出错，则可进行debug调试（点击小爬虫）（下载程序进入实验箱）进入debug调试模式（如下图1-8所示）

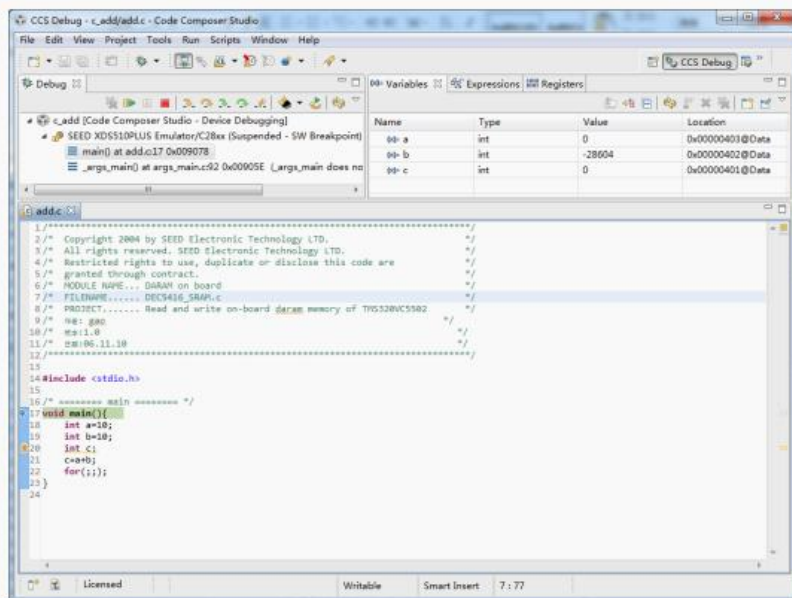


图1-8

- 3) 在代码中, for循环语句前, 蓝色区域双击, 从而产生一个断点, 然后点击运行按钮, 注意观察右上方变量区域的变化 (图1-9)

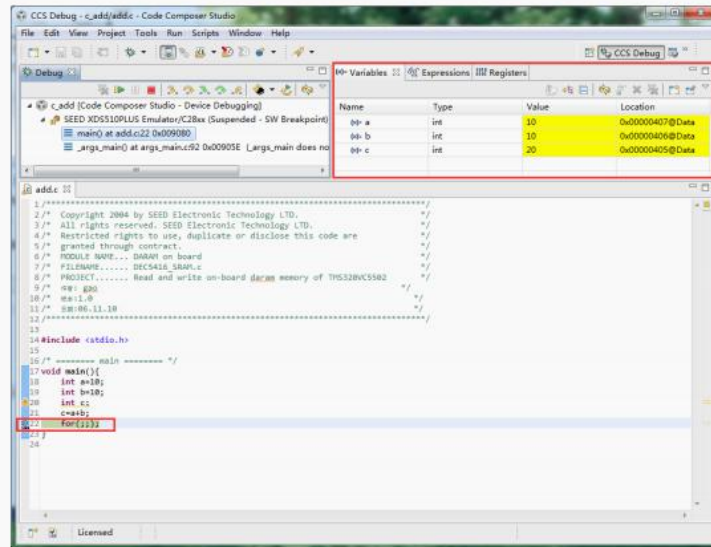


图1-9

至此, 我们对CCS的使用有了基本的了解, 同学们可以自己尝试debug界面的一些单步调试的功能

4. 导入一个已有 project 然后进行相关配置

- 1) 在导入工程之前, 因为我们前面创建了一个同名的工程, 因此在导入工程前先删除我们之前的工程 (右击工程名->delete), 参考附录中的实验经验部分
- 2) 在D盘新建一个test文件夹, 拷贝\SEED-DTK28335\03. Examples of Program\03. SEED-DTK28335 for CCS5.5目录下的3.1.2 c_add工程文件夹到test文件夹下
- 3) 如下图1-10所示, 点击project > Import Existing CCS Eclipse Projects > Browse

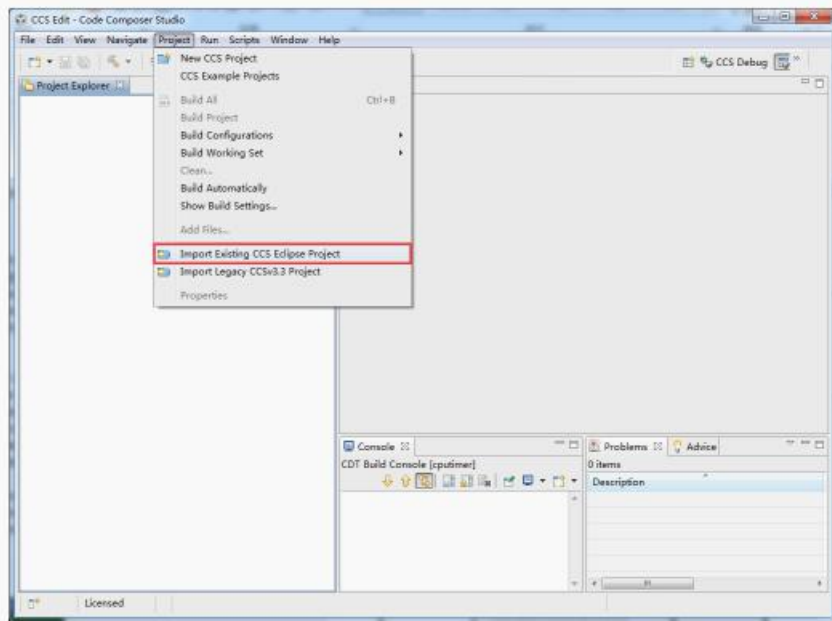


图1-10

- 4) 选择文件路径，找到3.1.2 c_add文件夹即可，点击打开，选择将其copy到我们的工作空间，然后finish

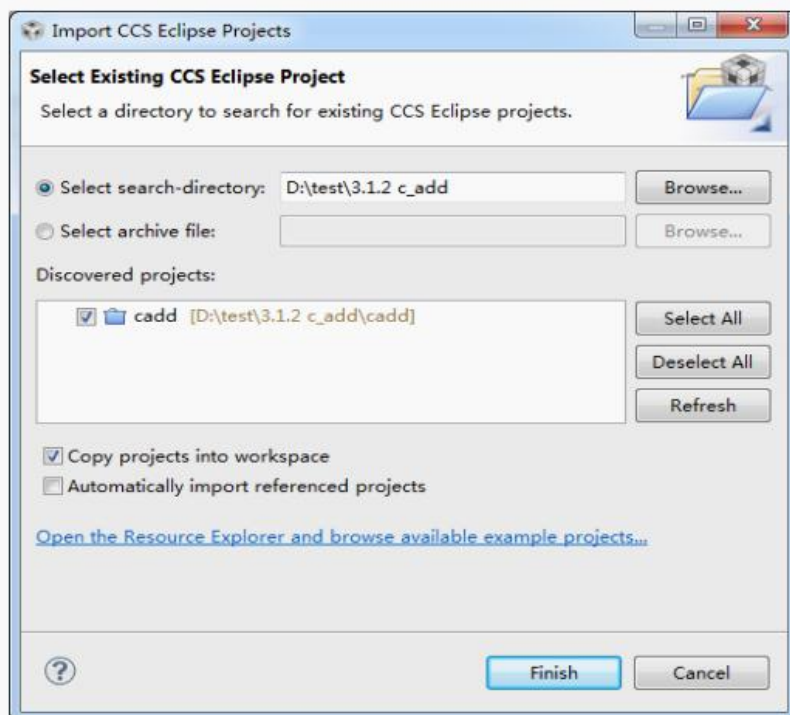


图1-11

- 5) 右击工程名 (cadd) > Add Files > 28335.gel(添加.gel文件到工程中)
- 6) 右击工程名 (cadd) > New > Targer configurition File

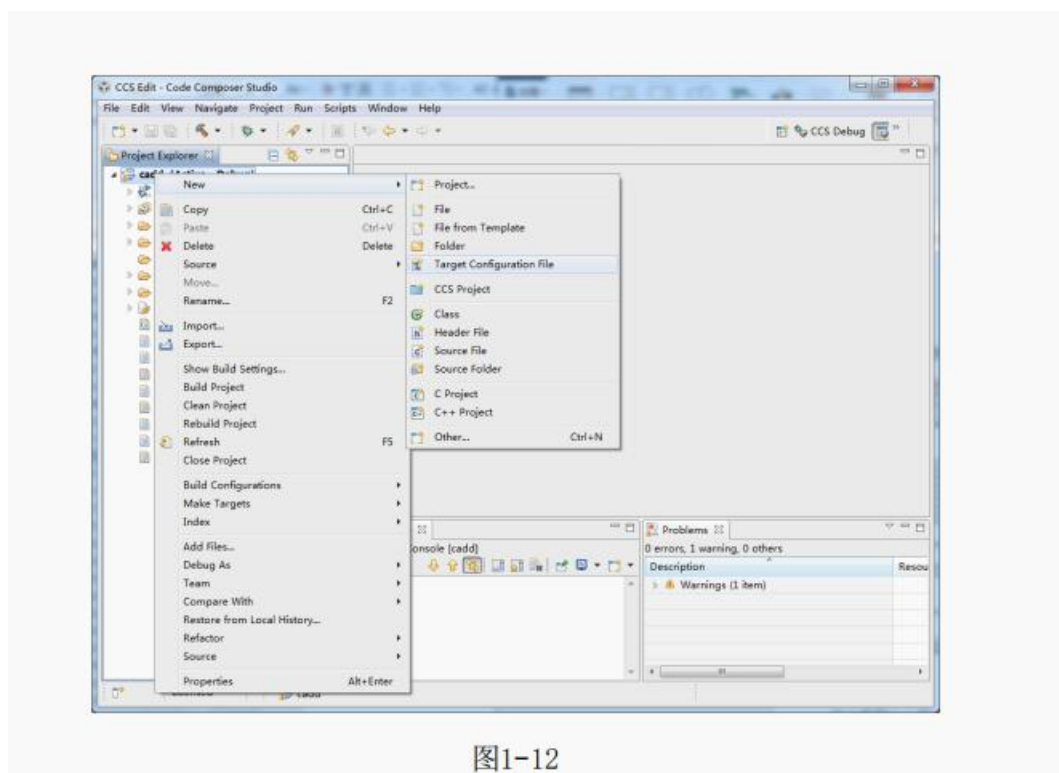


图1-12

- 7) 出现一个对话框，可修改文件名称(cadd.ccxml)，然后finish，随后出现如下图1-13所示

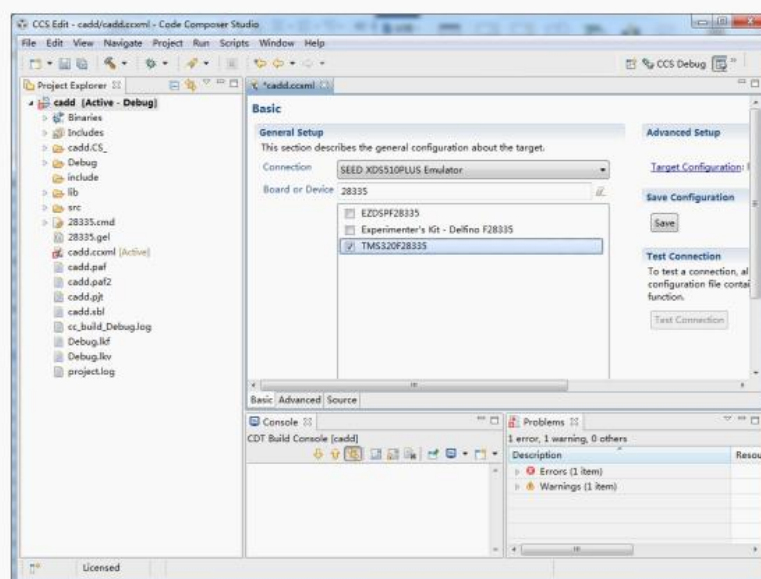


图1-13

至此界面与上面新建部分同样配置后，然后按照上面新建工程的调试步骤调试运行，发现是一样的效果

三、 实验结论

在这个试验中，我们学会了如何使用 CCS，CCS 是 TI 公司推出的一款具有环境配置、源文件编辑、程序调试、跟踪和分析等功能的集成开发环境，能够帮助用户在一个软件环境下完成编辑、编译、链接、调试和数据分析等工作。

我们用到了 debug 功能，在这个模式下，可以进行单步调试，打断点并全速运行到断点处，查看变量的变化情况，还能在全速运行状况下进行变量的在线观察，也就是 live watch。在这个状态下，我们可以通过调用栈查看函数使用情况和执行顺序，通过调用反汇编和内存，查看数据的真实变化情况，用来调试数组越界等非语法错误的问题。非常的好用。

我们还可以使用这个软件中的一些图形显示功能来查看变量的变化曲线等，这个对调试 adc 模块、pwm 模块非常有帮助。



浙江工业大学

实 验 二：DSP 数据存取实验

指导教师：吴春

班 级：电气 1701 班

姓 名：黄林志

学 号：201706060301

浙江工业大学

一、实验目的

1. 了解 TMS320F28335 的内部存储器空间的分配及指令寻址方式;
2. 学习用 Code Composer Studio 修改、填充 DSP 内存单元的方法;
3. 学习操作 TMS32028335 内存空间的指令。


二、实验内容

1. 读写 DSP 内存单元数据;
2. 复制内存单元的数据。

三、实验程序功能与结构说明

1. Memory.c: 实验的主程序, 包含了系统初始化, 读写、复制 DSP 内存单元等;
2. 28335.cmd: 声明了系统的存储器配置与程序各段的连接关系。
3. f28335.gel: 系统初始化

四、实验步骤:

1. 实验准备
 - 1) 首先将 03.SEED-DTK28335 for CCS5.5 目录下的 3.2.1 DEC28335_Memory 的文件夹拷贝到 D: 盘根目录下的 test 文件夹下。
 - 2) 将 DSP 仿真器与计算机连接好;
 - 3) 将 DSP 仿真器的 JTAG 插头与 SEED-DEC28335 单元的 J18 相连接;
 - 4) 打开 SEED-DTK28335 的电源。观察 SEED-DTK_MBoard 单元的 +5V, +3.3V, +15V, -15V 的电源指示灯以及 SEED_DEC28335 的电源指示灯 D2 是否均亮; 若有不亮的, 请断开电源, 检查电源。
2. 双击  图标, 打开 CCS, 进入 CCS 的操作环境。
3. 导入 CCS 工程 (参照实验一导入已有工程部分)
4. 设置仿真器 (参考实验一)
5. 阅读实验源代码, 分析整个程序运行过程
6. 进入 debug 调试模式, 打开 view-disassembly 反汇编窗口如下图 2-1 所示, 发现 main 函数的入口地址为 0x9046 (仔细比较一下汇编程序与 C 语言程序的对应关系)

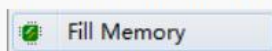


-
- Memory Browser [X] Disassembly
- Data 0x9046 Go New Tab
- Data:0x9046 <Memory Rendering 4> [X]
- 16-Bit Hex - TI Style
- | | | | | | |
|------------|------|------|------|------|------|
| 0x00009046 | FE08 | 8F00 | 0200 | A844 | 8F00 |
| 0x00009048 | 0210 | A846 | 2B41 | 0644 | 1E48 |
| 0x00009050 | 9241 | 5210 | 630B | 8A48 | 9241 |
| 0x00009055 | 96C4 | 0201 | 0A41 | 5601 | 0048 |
| 0x0000905A | 9241 | 5210 | 64F7 | 7625 | 2B41 |
| 0x0000905F | 1E48 | 9241 | 5210 | 630B | 8A48 |
| 0x00009064 | 28C4 | 1234 | 0201 | 0A41 | 5601 |
| 0x00009069 | 0048 | 9241 | 5210 | 64F7 | 7625 |
| 0x0000906E | 9241 | 5210 | 630E | 8A44 | 8346 |
| 0x00009073 | 92C4 | 96C5 | 0201 | 0A41 | 5601 |
| 0x00009078 | 0044 | 5601 | 0046 | 9241 | 5210 |
| 0x0000907D | 64F4 | 7625 | 28AB | FFFF | 28AA |
| 0x00009082 | FFFF | 28A9 | FFFF | 28A8 | FFFF |

图 2-2

8. 填充数据单元

在 Memory Browser 窗口，点击绿色图标旁下拉键，然后选择



，填写起始地址，以及填写长度，内容等参数（理解下图的意思，为何要这样的操作）

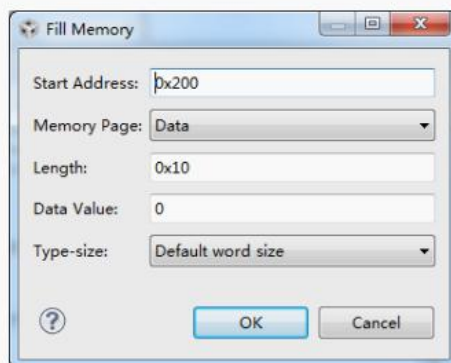


图 2-3

9. 运行程序观察结果

在 Memory.c 程序的” for (i=0, pz=py; i<16; i++, pz++)” , ” for (i=0; i<16; i++, px++, py++)” , ” for(;;);” 处设置断点。 运行程序，程序会停在断点处，此时可观察到 Data 窗口中前 16 个单元的值

运行至第一个断点处：

0x200 的存储情况如下：（思考为何是这些数）

0x00000200 0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 000A 000B 000C 000D 000E 000F

0x210 的存储情况如下：

0000 0001 0002 0003 0004 0005 0006 0007 0008 0009

运行至第二个断点处：

0x200 的存储情况如下：

0x00000200 0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 000A 000B 000C 000D 000E 000F

0x210 的存储情况如下：

1234 1234 1234 1234 1234 1234 1234 1234 1234 1234

运行至第三个断点处：

0x200 的存储情况如下：

0x0000200 0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 000A 000B 000C 000D 000E 000F

0x210 的存储情况如下：

0000 0001 0002 0003 0004 0005 0006 0007 0008 0009

整个实验到这个地方就结束了，此实验主要是验证 DSP 的存取，同学们需要思考一下整个数据搬运的过程

五、思考题

1. 在填充 0x200 的数据时，我们换成 0x01 会有什么样的效果呢？

0x01 区域为 BEGIN 和 BOOT 的区域，涉及到芯片的启动，若随意修改可能会导致芯片无法正常启动。

六、试验总结

这个实验让我们学会了如何调用并查看反汇编内容和内存里的数据。CCS 的这个功能还是很强大的，因为它显示的内容在旁边都有和 C 语言一一对应。通过查看地址，我们可以使用 fill memory 功能在线更改变量的值，这样我们可以实现一些特殊的功能，比如代码里如果有一个 switch(chooseTemp)case:..... 语句，通过更改 chooseTemp 变量的值，我们可以进行不同的操作。不得不说这个功能非常的有用。

当然，我们还学到了许多 C 语言的基础知识和这款 dsp 的一些特

点。比如这个 dsp 中的 int 是 16 位的，且存储器基础单位也是 16 位的，调用 sizeof (int) 和 sizeof(char) 时返回的都是 1。我认为他的 char 是只有 8 位的，但是他所占用的空间是 16 位的。所以会和 sizeof(int) 一样都返回 1。



浙江工业大学

实验三： 定时器实验

指导教师：吴春

班 级：电气 1701 班

姓 名：黄林志

学 号：201706060301

浙江工业大学

一、实验目的

1. 熟悉如何编写 28335 的中断服务程序
2. 掌握长时间间隔的定时器的处理
3. 掌握片内外设的设置方法

二、实验内容

1. 系统初始化
2. DSP 的初始设置
3. 定时中断的编写

三、实验背景知识

TMS320F28335 片上有 3 个 32-位 CPU 定时器，分别被称为 CPU 定时器 0、1 和 2。每个定时器中均有一个 32-位减计数器，当计数器减到 0 时，产生一个中断。其中，CPU 定时器 0 的中断 TINT0 为 PIE 中断，CPU 定时器 1 的中断 TINT1 直接连到 CPU 中断的 INT13，CPU 定时器 2 的中断 TINT2 直接连到 CPU 中断的 INT14。CPU 定时器 2 保留为实时操作系统（如 DSP BIOS）使用，而 CPU 定时器 0、1 则可被用户使用，SEED-DEC28335 未使用 CPU 定时器 0，用户可以根据应用的需要灵活使用。

CPU 定时器的原理框图和定时中断如下图 3-1、3-2 所示：

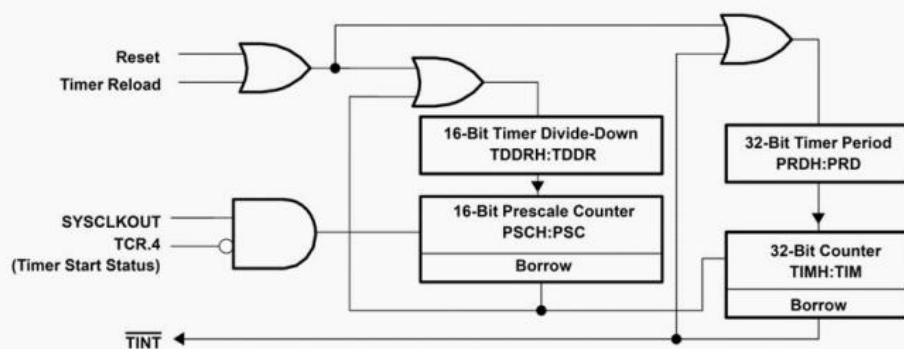


图 3-1

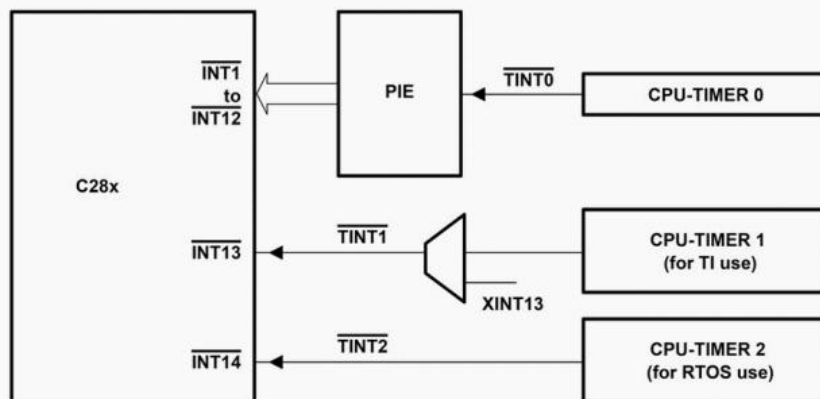


图 3-2

定时器在工作过程中，首先用 32 位计数寄存器（TIMH: TIM）装载周期寄存器（PRDH: PRD）内部的值。计数寄存器根据 SYSCLKOUT 时钟递减计数。当计数寄存器等于 0 时，定时器中断输出产生一个中断脉冲。

定时器计数器（TIMH: TIM）：TIM 寄存器保存当前 32 位定时器计数值的低 16 位，TIMH 寄存器保存高 16 位。每隔（TDDRH: TDDR+1）个时钟周期 TIMH: TIM 减 1，当 TIMH: TIM 递减到 0 时，TIMH: TIM 寄存器重新装载 PRDH: PRD 寄存器保存的周期值，并产生定时器中断 TINT 信号。

定时器周期寄存器（PRDH: PRD）：PRD 寄存器保存 32 位周期值的低 16 位，PRDH 保存高 16 位。当 TIMH: TIM 递减到零时，在下次定时周期开始之前 TIMH: TIM 寄存器重新装载 PRDH: PRD 寄存器保存的周期值；当用户将定时器控制寄存器（TCR）的定时器重新装载位（TRB）置位时，TIMH: TIM 也会重新装载 PRDH: PRD 寄存器保存的周期值。

定时器控制寄存器 TCR：主要有 TIF 位：定时器中断标志，当定时器计数器递减到 0 时，该位置 1。可以通过软件向 TIF 写 1 将 TIF 位清零，但只有计数器递减到 0 时才会将该位置位；TIE 位：CPU 定时器中断使能；如果定时器计数器递减到零，TIE 置位，定时器将会向 CPU 产生中断；TRB 位：定时器重新装载控制位，当向 TCR 的 TRB 位写 1 时，TIMH: TIM 会重新装载 PRDH: PRD 寄存器保存的周期值，并且预定标计数器（PSCH: PSC）装载定时器分频寄存器（TDDRH: TDDR）中的值；定时器停止状态位 TSS，要停止定时器 TSS 位置 1；

四、实验要求：

通过本实验，熟悉中断的结构及用中断程序控制程序流程，掌握定时器的应用。

五、实验程序功能与结构说明：

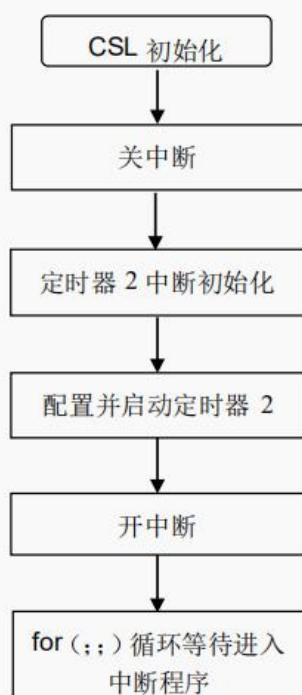
1. 定时器实验包含文件：

1) DSP2833X_CpuTimers.c：包含定时器初始化和配置函数。

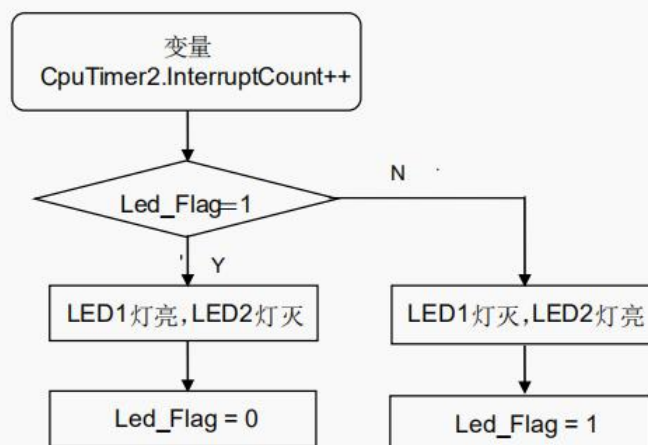
- 2) DSP2833X_DefaultIsr.c: 包含各中断默认的中断程序。
- 3) DSP2833X_GlobalVariableDefs.c: 定义各模块的全局变量。
- 4) DSP2833X_Gpio.c: Gpio 初始化。
- 5) DSP2833X_InitPeripherals.c: 包含各外设初始化。
- 6) DSP2833X_PieCtrl.c: 初始化各 PIE 控制寄存器。
- 7) DSP2833X_PieVect.c: PIE 中断向量表初始化。
- 8) DSP2833X_SysCtrl.c: 包含系统初始化函数等。
- 9) CpuTimer.c: 实验主程序, 包含系统初始化, 定时器中断初始化, 中断程序等。
- 10) DSP2833x_usDelay.asm: DSP 延时程序
- 11) DSP2833x_ADC_cal.asm: AD 寄存器设置
- 12) 28335.cmd: 声明了系统的存储器配置与程序各段的连接关系。
- 13) f28335.gel: 系统初始化
- 14) *.h: 各个源文件的头文件
- 15) rts2800_fpu32.l: 库函数文件

2. 程序流程图:

1) 主流程图



2) 中断流程图:



六、实验步骤

1. 实验准备

拷贝 cputimer 文件夹到 D 盘 test 目录下(参考前面实验)

2. 打开 CCS, 进入 CCS 的操作环境

3. 加载 cputimer 工程 (参照前面实验), 添加 28335.gel 文件

4. 设置仿真器

5. 阅读源代码, 理解整体程序运行流程

6. 编译、下载程序

7. 在 CpuTimer.c 程序 88 行 “Led_Flag= 0;” 和 93 行 “* Led_Flag= 1” 处设置断点, 如下图 3-3 所示

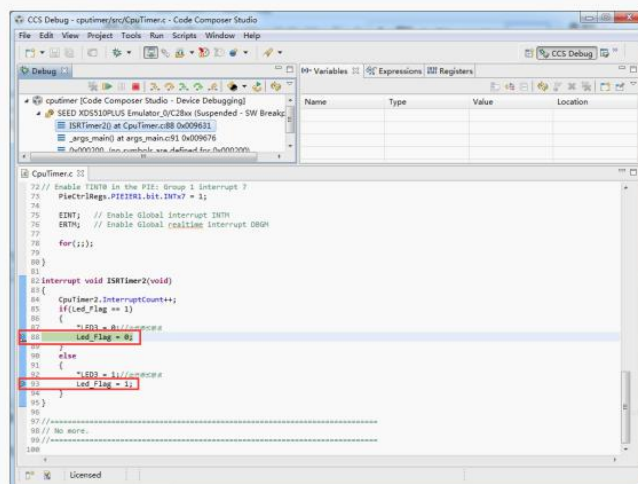


图 3-3

8. 运行程序, 观察结果

运行程序, 程序停在第一个断点处, 表明已进入定时器中断。观察指示灯 D3。

继续运行程序, 程序会停在第二个断点处。观察指示灯 D3。

若取消断点, 连续运行程序会看到 SEED_DEC28335 板卡上 D3 闪烁。

9. 实验者可根据自己的需要改变周期寄存器的值, 从而控制每次进中断的时间。

10. 退出 CCS

七、思考题

1、简述 CPU 定时器 2 中断的配置方法，

```
3 void ConfigCpuTimer(struct CPUTIMER_VARS *Timer, float Freq, float Period)
3 {
3     Uint32 temp;
1
2     // Initialize timer period:
3     Timer->CPUFreqInMHz = Freq;
4     Timer->PeriodInUsec = Period;
5     temp = (long) (Freq * Period);
5     Timer->RegsAddr->PRD.all = temp;
7
3     // Set pre-scale counter to divide by 1 (SYSCLKOUT):
3     Timer->RegsAddr->TPR.all = 0;
3     Timer->RegsAddr->TPRH.all = 0;
1
2     // Initialize timer control register:
3     Timer->RegsAddr->TCR.bit.TSS = 1;    // 1 = Stop timer, 0 = Start/Restart Timer
4     Timer->RegsAddr->TCR.bit.TRB = 1;    // 1 = reload timer
5     Timer->RegsAddr->TCR.bit.SOFT = 0;
5     Timer->RegsAddr->TCR.bit.FREE = 0;    // Timer Free Run Disabled
7     Timer->RegsAddr->TCR.bit.TIE = 1;    // 0 = Disable/ 1 = Enable Timer Interrupt
3
3     // Reset interrupt counter:
3     Timer->InterruptCount = 0;
1 }
2
```

2、在 30MHZ 的晶振频率下(ossclk)，如何设置各相关寄存器，使 CPU 定时器定时 2MS?

答：TDDRH:TDDR=0（分频值） sysCtrlRegs.PLLCR.bit.DIV=10(PLL 分频)

SysCtrlRegs.PLLSTS.bit.DIVSEL=2，即 PLL 倍频为 10/2=5

PRDH:PRD 为 150*2000=300000（周期值）

八、试验总结

这个试验，让我们学会了如何使用定时中断，主要是了解这款 DSP 的定时周期是如何配置的。其中 CPU TIMER0 是需要使用 PIE 中断的，而 TIMER2 是留给操作系统用的，当然也可以当做普通定时器使用。不过这个例程里面，他是把所有的中断都映射到 PIE 中断中的，所以我们使用定时器 2 的时候也初始化了 PIE 中断列表。



浙江工业大学

实验四：步进电机控制实验

指导教师：吴春

班 级：电气 1701 班

姓 名：黄林志

学 号：201706060301

浙江工业大学

一、实验目的

1. 了解步进电机驱动的原理；
2. 了解步进电机的控制原理；
3. 熟悉使用 PWM 控制步进电机的运行。

二、实验内容

1. DSP 的初始化；
2. EV 初始化与配置
3. 步进电机的驱动程序。

三、实验背景知识

1. 步进电机的驱动：图 4-1 是单极性步进电机翻译/驱动的典型电路，图中的方块为驱动开关。针对 SEED-DEC 中直流电机系统的动作要求，步进电机驱动电路设计思路如下：
 - 1) 电机采用 15V 直流电源供电；
 - 2) 4 路控制信号由 SEED-DTK_MBoard 提供，信号为 CMOS 标准电平，通过排线接入并下拉；
 - 3) 使用达灵顿管 TIP31C 代替 IRL540 作为电机驱动开关，基级串接 100Ω 电阻；
 - 4) 使用快速二极管 1N4007 完成保护功能，以免电机换向时烧毁电机；
 - 5) 使用 50Ω 限流电阻（半步运行时电流约为 $0.2A$ ，小于电机电源额定电流）；
 - 6) 电机电源地之间跨接电容，电机地与数字地之间采用磁珠连接共地；

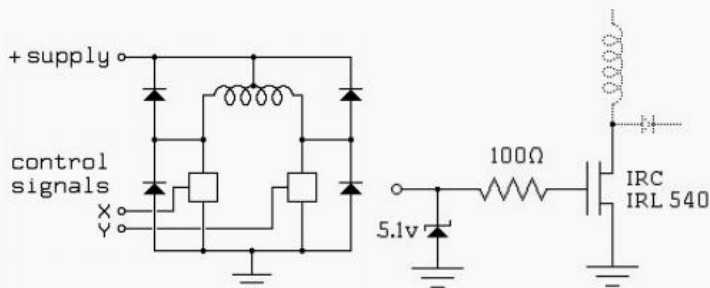


图 4-1

步进电机在这个实验中选择的是 M35SP-7N，其步进角为 7.5° ，是一种单极性的步进电机。它的结构如下图 4-2 所示：

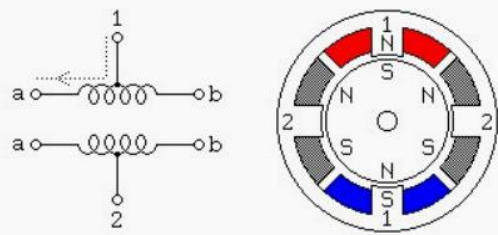


图 4-2

实际使用时，公共端 1 与 2 是短接在一起做为电源输入，一共五个抽头。控制每个绕组的两个抽头来实现对步进电机的控制。

步进电机的控制一般分为四相四拍与四相八拍两种方式，其中前者称为全步，后者称为半步。

2. 步进电机的驱动接口： 步进电机的控制接口为 EVA 的 PWM1、PWM2、PWM3、PWM4 引脚。它们分别为步进电机四相的控制端。按一定的频率循环置高电平即可使步进电机转动。引脚与步进电机的线圈对应关系如下图 4 所示：

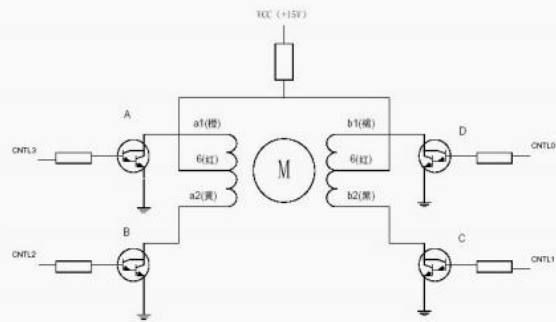


图 4-3

在步进电机为四相四拍时，其正转顺序为 $A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$ ，在控制器中 CNTL[3:0] 的输出依次为：

$0x8 \rightarrow 0x4 \rightarrow 0x2 \rightarrow 0x1$ 。

其反转的顺序为 $A \rightarrow D \rightarrow C \rightarrow B \rightarrow A$ ，在控制器中 CNTL[3:0] 的输出依次为：

$0x8 \rightarrow 0x1 \rightarrow 0x2 \rightarrow 0x4$ 。

四、实验要求

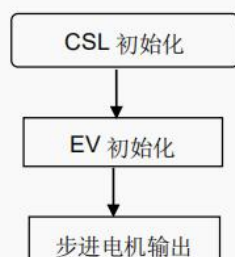
通过电机实验，了解对步进电机的驱动的基本原理。

五、实验程序功能与结构说明：

1. 步进电机实验包含文件：

- 1) stepmotor.c: 实验主程序，包含了系统初始化，步进电机控制，机调速等；
- 2) DSP2833X_EV.c: 包含了事件管理器初始化。
- 3) DSP2833X_DefaultIsr.c: 包含了异步串口接收中断服务程序。
- 4) DSP2833X_GlobalVariableDefs.c: 各个外设全局变量定义。
- 5) DSP2833X_PieCtrl.c: PIE 中断初始化。
- 6) DSP2833X_PieVect.c: PIE 中断矢量表初始化。
- 7) DSP2833X_SysCtrl.c: 系统初始化。
- 8) 28335.cmd: 声明了系统的存储器配置与程序各段的连接关系。
- 9) f28335.gel: 系统初始化
- 10) *.h: 各个源文件的头文件
- 11) rts2800_fpu32.l: 库函数文件

2. 程序流程图



六、实验步骤

1. 实验准备

拷贝 stepmotor 文件夹到 D 盘 test 目录下(参考前面实验)

2. 启动 CCS，进入 CCS 的操作环境。

3. 加载 stepmotor 工程，添加 28335.gel 文件

4. 设置仿真器
5. 阅读源程序，理解整个程序运行过程
6. 编译、下载程序
7. 运行程序，观察结果
8. 尝试将程序中的 0x3fff 修改成其他值（0x6fff, 0x1fff）观察结果，如下图 4-4 所示猜想为什么会有这种情况？

```

204 crm1Regs.IDCLVAL.LCLVAL = ID_VAL0; // SHOW JUST IN OBSERVE ON THE SCOPE
205 // Setup compare
206 EPwm1Regs.CMPA.half.CMPA = 0x3fff;
207 // EPwm1Regs.CPS = 0x3fff;
208 // Set actions
209 EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET; // Set PwM1A on Zero
210 EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;
211 EPwm1Regs.AQCTLB.bit.PR0 = AQ_SET; // Set PwM1A on Zero
212 EPwm1Regs.AQCTLB.bit.CAD = AQ_CLEAR;
213 }
214 void InitEPwm2Example()
215 {
216 EPwm2Regs.TBPRD = 0x7fff; // Set timer period
217 EPwm2Regs.TBPHS.half.TBPHS = 0; // Phase is 0
218 // Setup TBCLK
219 EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Count up
220 EPwm2Regs.TBCTL.bit.PHSEN = TB_ENABLE; // Disable phase loading
221 EPwm2Regs.TBCTL.bit.PRDL = TB_SHADOW; // Disable phase loading
222 EPwm2Regs.TBCTL.bit.SYNCSEL = TB_SYNC_IN;
223 EPwm2Regs.CMPCTL.bit.SHOWMODE = CC_SHADOW; // Load registers every ZERO
224 EPwm2Regs.CMPCTL.bit.SHOWMODE = CC_SHADOW;
225 EPwm2Regs.CMPCTL.bit.LOADMODE = CC_CTR_ZERO;
226 EPwm2Regs.CMPCTL.bit.LOADMODE = CC_CTR_ZERO;
227 EPwm2Regs.CMPCTL.bit.LOADMODE = CC_CTR_ZERO;
228 EPwm2Regs.TBCTL.bit.HSPCLKDIV = TB_DIV8; // Clock ratio to SYSCLKOUT
229 EPwm2Regs.TBCTL.bit.CLKDIV = TB_DIV8; // Slow just to observe on the scope
230 // Setup compare
231 EPwm2Regs.CMPA.half.CMPA = 0x3fff;
232 // EPwm2Regs.CPS = 0x3fff;
233 // Set actions
234 EPwm2Regs.AQCTLA.bit.PR0 = AQ_CLEAR; // Set PwM2A on Zero
235 EPwm2Regs.AQCTLA.bit.CAU = AQ_SET;
236 EPwm2Regs.AQCTLB.bit.ZRO = AQ_CLEAR;
237 EPwm2Regs.AQCTLB.bit.CAD = AQ_SET; // Set PwM2A on Zero
238 }

```

图 4-4

9. 退出 CCS。

五、实验结果与分析

可以看到，步进电机的转速随我们修改值而改变

六、思考题

尝试将程序中的 0x3fff 修改成其他值（0x6fff, 0x1fff）观察结果，如下图 4-4 所示猜想为什么会有这种情况？

答：步进电机的转速随之改变。因为步进电机是 1 个脉冲转动一个步距角的，当我们改变频率的时候，就相当于改不了，单位时间产生的脉冲的个数，所以频率越高，转速也就越快了

七、实验总结

这次实验中，我们尝试了使用 PWM 来控制步进电机的转速，其实并不是通过 PWM 的占空比来改变步进电机的转速，而是通过改变 PWM 的频率来改变他的转速。Pwm 意为脉冲宽度调制，我们这种控制方法应该算是脉冲频率控制的。

当然，在烧入代码之前，我还复习了一些步进电机的知识：

- 1、步进电机可以实现电机转速和位置的精确控制，一般步进电机的精度为步进角的 5%之内，且不累积误差。
- 2、步进电机必须加驱动才可以运转，驱动信号必须为脉冲信号，没有脉冲的时候，步进电机静止，如果加入适当的脉冲信号，就会以一定的角度（称为步距角）转动。转动的速度和脉冲的频率成正比。
- 3、改变脉冲的顺序，可以方便的改变转动的方向。
- 4、步进电机低速时可以正常运转，但若高于一定速度就无法启动，并伴有啸叫声。步进电机有一个技术参数：空载启动频率，即步进电机在空载情况下能够正常启动的脉冲频率，如果脉冲频率高于该值，电机不能正常启动，可能发生丢步或堵转。在有负载的情况下，启动频率应更低。如果要使电机达到高速转动，脉冲频率应该有加速过程，即启动频率较低，然后按一定加速度升到所希望的高频（电机转速从低速升到高速）。



浙江工业大学

实验五：直流电机控制实验

指导教师：吴春

班 级：电气 1701 班

姓 名：黄林志

学 号：201706060301

浙江工业大学

一、实验目的

1. 了解直流电机驱动的原理；
2. 了解 PWM 对直流电机的驱动原理；
3. 了解使用 PWM 控制直流电机的实现过程。

二、实验内容

1. DSP 的初始化；
2. PWM 产生的定时中断服务程序。

三、实验背景知识

1. PWM 简介

脉宽调制 (PWM) 是利用微处理器的数字输出来对模拟电路进行控制的一种非常有效的技术，广泛应用在从测量、通信到功率控制与变换的许多领域中。PWM 是一种对模拟信号电平进行数字编码的方法。通过高分辨率计数器的使用，方波的占空比被调制用来对一个具体模拟信号的电平进行编码。

PWM 有如下特点：

- 16 位寄存器
- 可编程死区，最小死区宽度为一个 CPU 时钟周期
- 可直接改变 PWM 频率
- 每个 PWM 周期内或周期结束后都可以改变 PWM 脉宽，最小脉宽和调整最小量都是一个 CPU 时钟周期
- 外部可屏蔽的功率和驱动保护
- 脉冲生成电路可以用来产生可编程的不对称、对称以及 8 个空间矢量的 PWM 波形
- 自动装载比较和周期寄存器减少 CPU 开销 PDPINTx 可直接屏蔽 PWM 输出

F28335 有六个增强型脉宽调制模块，原理框图如下图所示：

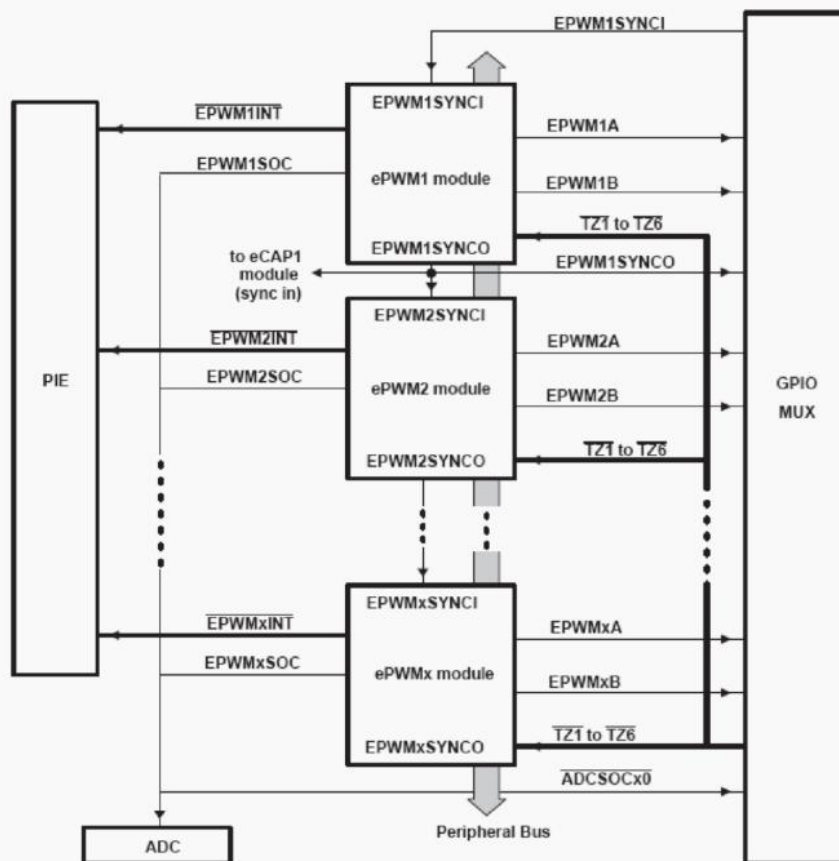


图 5-1

图 5-1

2. PWM 的管脚和信号

EPWM1A (O)	J1-1	J1-2	EPWM1B (O)
EPWM2A (O)	J1-3	J1-4	EPWM2B (O)
EPWM3A (O)	J1-5	J1-6	EPWM3B (O)
EPWM4A (O)	J3-1	J3-2	EPWM4B (O)
EPWM5A (O)	J3-3	J3-4	EPWM5B (O)
EPWM6A (O)	J3-5	J3-6	EPWM6B (O)

F28335 的脉宽调制模块的详细说明和编程操作请参考《*TMS320x28xx, 28xxx Enhanced Pulse Width Modulator (ePWM) Module Reference Guide*》。

3. 直流电机控制

1) 直流电机驱动

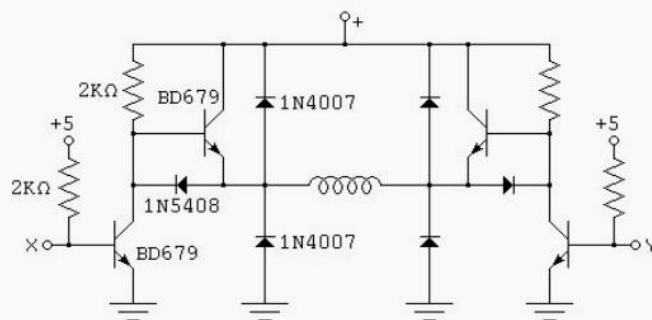


图 5-2

图 1 是直流电机翻译/驱动的典型电路的一个变种，采用这种电路不但能够完成直流电机驱动的动作，而且可以避免典型 H 桥电路潜在的短路危险。针对 SEED-DEC 中直流电机系统动作要求和电机的特点，电机驱动电路设计思路如下：

- a 电机采用 15V 直流电源供电，串接 50Ω @3W 电阻限流并分压；
- 2 路控制信号 X、Y 由 SEED-DTK_MBoard 提供，信号为 CMOS 标准电平；
- b 使用达灵顿管 TIP31C 代替 BD679 作为电机驱动开关，基级串接 100Ω 电阻；
- c 使用快速二极管 1N4007 完成保护功能，以免电机换向时烧毁电机；

电机电源/地之间跨接电容，电机地与数字地之间采用磁珠连接共地；

2) 直流电机驱动接口

本实验箱通过控制 EVA 的 EPWM3A 与 EPWM3B 引脚实现对直流电机的控制。

四、实验要求

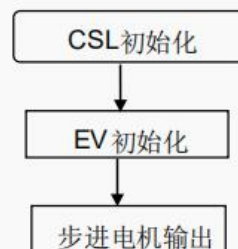
通过电机实验，了解对直流电机的驱动的基本原理，了解 PWM 波形的产生与调节

五、实验程序功能与结构说明

- 1. 直流电机实验，包含文件：

- 1) MOTOR.c: 实验主程序, 包含了系统初始化, 直流电机控制, 机调速等。
- 2) DSP2833X_EV.c: 包含了事件管理器初始化。
- 3) DSP2833X_DefaultIsr.c: 包含了异步串口接收中断服务程序。
- 4) DSP2833X_GlobalVariableDefs.c: 各个外设全局变量定义。
- 5) DSP2833X_PieCtrl.c: PIE 中断初始化。
- 6) DSP2833X_PieVect.c: PIE 中断矢量表初始化。
- 7) DSP2833X_SysCtrl.c: 系统初始化。
- 8) 28335.cmd: 声明了系统的存储器配置与程序各段的连接关系。
- 9) f28335.gel: 系统初始化
- 10) *.h: 各个源文件的头文件
- 11) rts2800_fpu32.l: 库函数文件

2. 程序流程图



六、实验步骤

1. 实验准备

拷贝 DCmotor 文件夹到 D 盘 test 目录下 (参考前面实验)

2. 打开 CCS, 进入 CCS 的操作环境

3. 加载 DCmotor 工程 (参照前面实验), 添加 28335.gel 文件

4. 设置仿真器

5. 阅读源代码, 理解整体程序运行流程

6. 编译、下载程序

7. 运行程序, 观察结果

七、实验结果

通过实验可以发现, 直流受控改变转速和方向。

八、实验总结

直流电机的控制是通过改变 **PWM** 来实现的，当频率大于 **10Khz** 之后，噪音会有显著减小。不同的 **PWM** 值会产生不同的电压值，从而改变负载电流，产生电磁转矩，使得电机旋转起来，说到底，其实是电流控制的转速。

Pwm 改变->电压改变->电流改变->转矩改变。



浙江工业大学

实验六：A/D 采集实验

指导教师：吴春

班 级：电气 1701 班

姓 名：黄林志

学 号：201706060301

浙江工业大学

实验六：A/D 采集实验

四、 实验目的

4. 了解 TMS320F28335 片上外设 AD;
5. 熟悉片上 AD 的使用;
6. 利用片上 AD 进行数据采集。

五、 实验内容

1. 实验步骤

1) 实验准备

- a) 拷贝 AD 文件夹到 D 盘 test 目录下
- b) （参考前面实验）本实验和 FFT 实验，数字滤波实验都需要设置实验箱信号源。通过液晶屏和键盘，设置信号源：当液晶屏上出现“通讯自检不成功，请复位系统”时，按下“Enter”键，进入“信号发生器设置”。在“信号发生器设置”这一菜单下：“通道”设为“0”；
“信号类型”可根据需要任意选择；
“信号频率”和“信号振幅”可在屏幕下方“有效输入”限定的范围内任意输入，建议振幅设为 1000 左右，频率”设为 300 左右；
“信号发生器开关”设为“开启”。此时便有信号输入到 ADCINA6 通道。

2) 启动 CCS，进入 CCS 的操作环境。

3) 加载 AD 工程，添加 28335.gel 文件

4) 设置仿真器

5) 阅读源程序，理解整个程序运行过程

6) 编译、下载程序

7) 运行程序观察结果

单击“Debug”菜单，“Run”项，运行程序；

8) 打开观察窗口

- a) 选择菜单 Tools->Graph->Single Time 做如下设置，然后单击“OK”按钮
 - b) 停止运行，观察“Single Time”窗口中的图形显示；
 - c) 适当修改信号发生器的输出，按 F8 键再次运行，停止后观察图形窗口中的显示。
- 9) 退出 CCS

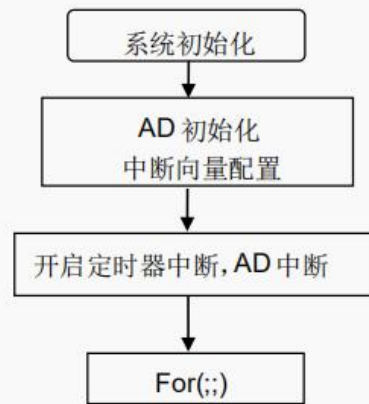
三、实验程序功能与结构说明：

1. AD 实验包含文件：

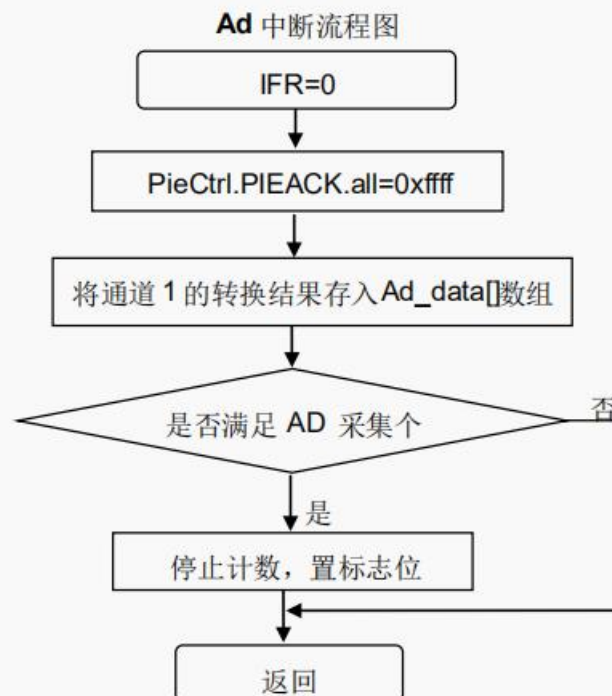
- 1) AD.c: 实验主程序，包含了系统初始化，AD 初始化，AD 中断初始化，Timer 中断初始化，A/D 采样及控制其采样频率的 Timer 中断程序。
- 2) DSP2833X_Adc.c: 包含 AD 模块初始化。
- 3) DSP2833X_CpuTimers.c: 包含定时器初始化和配置函数。
- 4) DSP2833X_DefaultIsr.c: 包含各中断默认的中断程序。
- 5) DSP2833X_GlobalVariableDefs.c: 定义各模块的全局变量。
- 6) DSP2833X_PieCtrl.c: 初始化各 PIE 控制寄存器。
- 7) DSP2833X_PieVect.c: PIE 中断向量表初始化。
- 8) DSP2833X_Sci.c: 包含 SCI 模块初始化和操作函数。
- 9) DSP2833X_SysCtrl.c: 包含系统初始化函数等。
- 10) 28335.cmd: 声明了系统的存储器配置与程序各段的连接关系。
- 11) f28335.gel: 系统初始化
- 12) *.h: 各个源文件的头文件
- 13) rts2800_fpu32.l: 库函数文件

2. 程序流程图

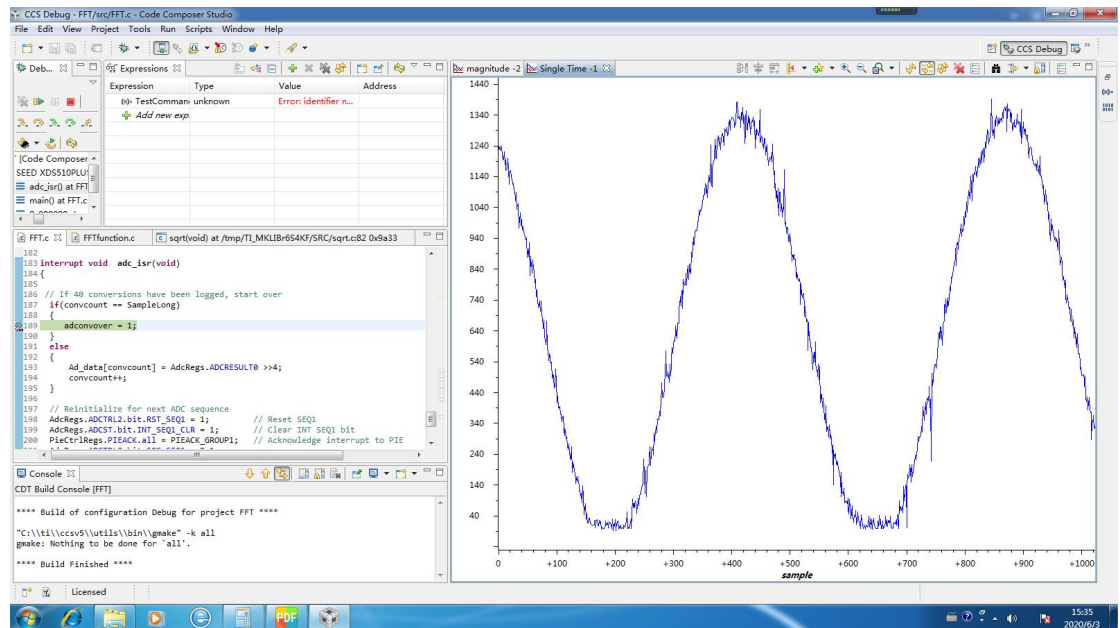
1) 主程序流程图



1) 中断流程图



四、实验结论



观察到正弦波。

四、思考题

1、如果 F28335 的片内 ADC 无法满足要求，如何扩展 ADC 芯片。

答： 可以通过 SPI 接口或外部扩展接口 (XINTF) 扩展串行或并口接口的 ADC 芯片。

2、试针对某一数字信号处理或数字控制系统，简述 A/D 转换器的功能？

答：水箱液位控制系统。通过水箱中的压力传感器（那种 U 型管，根据气压大小检测），压力传感器把检测到的压力转换成电压（模拟量-》数字量-》模拟量），主控制器采集电压值，转换成数字量，然后通过 PID 算法来得到一个控制量（数字量），并输出 PWM，进行液位的调节，控制。

五、实验总结

我们都知道自然界中存在的是模拟量，而我们计算机，最后产生的代码都是转换成了 0 和 1 的数字量，所以如果我们需要控制或者是采集模拟量，那么必定要对他们进行转换。而根据香农采样定理，我们对模拟量进行采样时，采样频率必须是信号最高频率的 2

倍以上。采样频率越高，越逼近原来的模拟量，但是比不是说采样频率越高越好的，因为每款芯片的 IO 都是有最高反应速度的，过高的频率，会导致 IO 来不及反应，而且会增加系统的功耗。所以我们决定采样频率大小时，应该时根据自己的需求和算法来进行调整的。

这次实验我还学到了不少关于 AD 的知识。Eg:

1、采样窗是用来计算采样保持的时间的；2、AD 采样的采样频率是可以用 PWM 频率来进行修改的（当然不是所有芯片都行）。



浙江工业大学

实验七：交通灯实验

指导教师：吴春

班 级：电气 1701 班

姓 名：黄林志

学 号：201706060301

浙江工业大学

六、 实验目的

7. 掌握 DSP 扩展数字 I/O 口的方法;
8. 了解 SEED-DEC28335 的硬件系统。

七、 实验内容

1. 实验步骤

10) 实验准备

- a) 拷贝 AD 文件夹到 D 盘 test 目录下 (参考前面实验)

11) 打开 CCS, 进入 CCS 的操作环境。

12) IO 工程, 添加 28335.gel 文件

13) 设置仿真器

14) 阅读源程序, 理解整个程序运行过程

15) 打开 IO.c 文件, 到第 25 行, 修改 TESTCOMMAND 的宏定义;

TESTCOMMAND 是交通灯操作控制选项。可以为 1、2、3、4、5 这 5 个数

1 为自动运行; 2 为夜间模式; 3 为交通灯东西通; 4 为交通灯南北通; 5 为禁行。

16) 编译、下载程序

17) 运行, 观察。在程序运行过程中, 暂停之后可直接在 Watch Window 里修改 TestCommand 的值, 即将每一种运行方式所对应宏定义的值直接赋值给 TestCommand, 即可改变运行方式。例如在程序运行过程中, 若想将运行方式改为夜间模式, 就请将 TestCommand 赋值为 0xAA16 (关于各种方式的宏定义已在第 33 行到第 37 行给出) 即可。

(点击 view->Expressions 打开窗口, 然后点击 add new expression 添加变量, 然后 debug 之后, 可以右击 value 选择以 hex 显示数据)

三、实验背景知识

1. DSP 系统中数字 I/O 的实现

DSP 系统中一般只有少量的数字 I/O 资源，而一些控制中经常需要大量的数字量的输入与输出。因而，在外部扩展 I/O 资源是非常有必要的。在扩展 I/O 资源时一般占用 DSP 的 I/O 空间。其实现方法一般有两种：其一为采用锁存器像 74LS273、741S373 之类的集成电路；另一种是采用 CPLD 在其内部做锁存逻辑，我们采用的是后者。

‘F28335 为 32-位浮点 DSP，其外部存储器接口只支持 16-位、异步存储器访问，所以 SEED-DEC28335 上的存储器扩展总线只实现 16-位、异步存储器接口，其 4-个存储空间 CE0~CE3 映射到’ F28335 外部存储空间 Zone 7 上，而’ F28335 的 Zone 7 存储空间的大小只有 1M×16-位。在此，我们采用分页技术将 4 个 1M×16-位的扩展总线存储空间 CE0~CE3 映射到’ F28335 的 Zone 7 存储空间中。也即将 4M×16-位的扩展总线存储空间分为 4 个 1M×16-位的存储空间页，2 位页地址由扩展总线存储器空间页地址寄存器（映射在 Zone 0 的 0x00,4060，只写）产生，其控制位的定义如下：

D7~D2 D1 D0

保留	PA21	PA20
----	------	------

PA[21:20]: 2 位页地址，用于选择 4 个 1M×16-位扩展总线存储空间

00: 选择扩展总线的 CE0 空间

01: 选择扩展总线的 CE1 空间

02: 选择扩展总线的 CE2 空间

03: 选择扩展总线的 CE3 空间

实验箱 I/O 控制部分映射到 SEED-DEC28335 模板的 CE3 空间，接口方式为 16-位。地址映射关系如下：

实验箱 I/O 板对应的起始地址为：0x 200000（字地址）；

SEED-DEC2812 板卡控制 Mboard 板 CPLD 的地址为 0x200005；向此地址写 1 使能控制功能。

TRAFFIC LED 的偏移地址为：0x0000；即 TRAFFIC LED 的地址为：0x200000；

2. 数字 I/O 所占的资源

交通灯控制口地址为：0x80000(I/O 空间)；其说明如下：

D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
SR	SY	SG	WR	EG	EY	WY	ER	WG	NR	NY	NG

NG: 方向北的绿灯控制位;

NY: 方向北的黄灯控制位;

NR: 方向北的红灯控制位;

WG: 方向西的绿灯控制位;

ER: 方向东的红灯控制位;

WY: 方向西的黄灯控制位;

EY: 方向东的黄灯控制位;

EG: 方向东的绿灯控制位;

WR: 方向西的红灯控制位;

SG: 方向南的绿灯控制位;

SY: 方向南的黄灯控制位;

SR: 方向南的红灯控制位; 当以上各位置“1”时, 点亮各控制位所代表的交通灯状态的 LED 灯。

四、实验要求

通过本实验, 了解 DSP 对 I/O 口的操作, 完成交通灯的控制。熟练使用 CCS 对程序进行调试

五、实验程序功能与结构说明

1. 数字 IO 实验包含文件:

- 1) IO.c: 这是实验的主程序, 包含了系统初始化, 并完成控制交通灯按照所选择的模式输出显示
- 2) DSP2833X_GlobalVariableDefs.c: 定义各模块的全局变量。

3) DSP2833X_SysCtrl.c: 系统初始化函数。

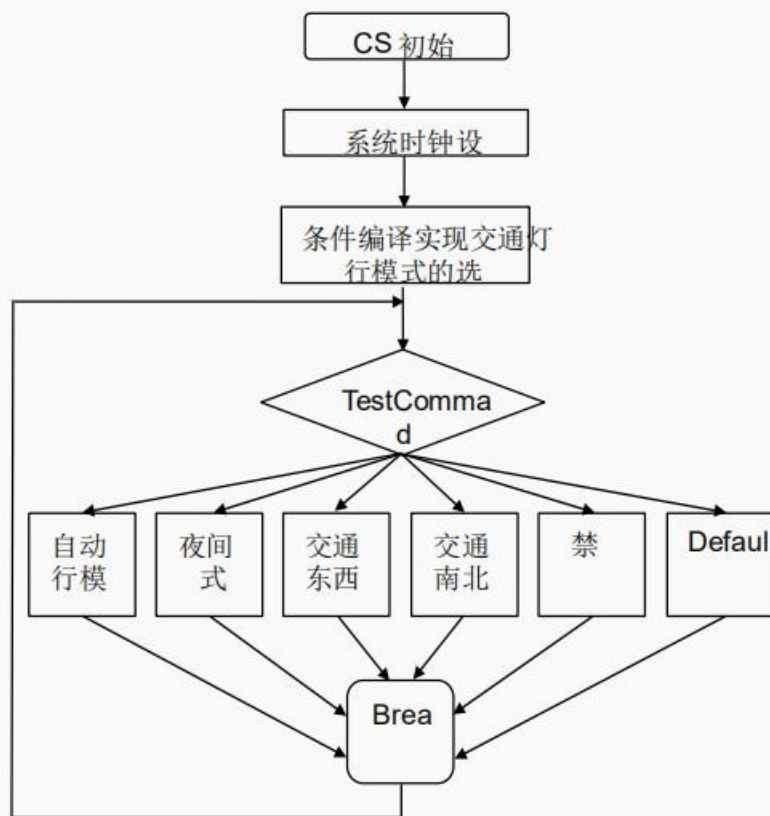
4) 28335.cmd: 声明了系统的存储器配置与程序各段的连接关系。

5) f28335.gel: 系统初始化

6) *.h: 各个源文件的头文件

7) rts2800_fpu32.l: 库函数文件

2. 程序流程图



六、实验步骤

1. 实验准备

拷贝 AD 文件夹到 D 盘 test 目录下（参考前面实验）

2. 打开 CCS, 进入 CCS 的操作环境。

3. I/O 工程, 添加 28335.gel 文件

4. 设置仿真器

5. 阅读源程序, 理解整个程序运行过程

6. 打开 I0.c 文件, 到第 25 行, 修改 TESTCOMMAND 的宏定义;

TESTCOMMAND 是交通灯操作控制选项。可以为 1、2、3、4、5 这 5 个数（如下图 7-1 所示）。

1 为自动运行；2 为夜间模式；3 为交通灯东西通；4 为交通灯南北通；5 为禁行。
SEED-DTK_MBoard 单元的 Traffic Lamp 处将显示结果；

```

17 /*交通灯操作宏定义*/
18 #define EASTWEST 0x88c //交通灯东西通（南北禁行）
19 #define SOUTHNORTH 0x311 //交通灯南北通（东西禁行）
20 #define IOCHANGE 0x462 //交通灯各方向黄灯亮
21 #define ALLFORBIN 0x914 //交通灯各方向绿灯行
22 //宏选择：
23 //选择TESTCOMMAND:1为自动运行,2为夜间模式,3为交通灯东西通,
24 // 4为交通灯南北通,5为禁行
25 #define TESTCOMMAND 1 //交通灯操作命令选择
26
27 /*定义扩展寄存器空间页地址寄存器地址为0x004060*/
28 volatile unsigned int* p_cselect=(volatile unsigned int *)0x004060;
29 /*定义交通灯IO口的地址为0x20000*/
30 volatile unsigned int* p_trafficaddr=(volatile unsigned int *)0x200001;
31 /*定义交通灯IO口的地址为0x20005*/
32 volatile unsigned int* p_ioenable=(volatile unsigned int *) 0x200005;
33 ///////////////
34 unsigned int TestCommand;
35 unsigned int iostatus=0;
36
37 /*****

```

图 7-1

7. 编译、下载程序

8. 运行，观察。在程序运行过程中，暂停之后可直接在 Watch Window 里修改 TestCommand 的值，即将每一种运行方式所对应宏定义的值直接赋值给 TestCommand，即可改变运行方式。例如在程序运行过程中，若想将运行方式改为夜间模式，就请将 TestCommand 赋值为 0xAA16（关于各种方式的宏定义已在第 33 行到第 37 行给出）即可。（点击 view->Expressions 打开窗口，然后点击 add new expression 添加变量，然后 debug 之后，可以右击 value 选择以 hex 显示数据）如下图 7-2 所示：

Variables Expressions Registers			
Expression	Type	Value	Address
TestCommand	unsigned int	0xAA14 (Hex)	0x0000C000@Data
Add new expression			

图 7-2

七、实验结果

观察试验箱上交通灯部分的情况，尝试修改交通灯的时间

八、试验总结

这个实验，是通过 **XINTF** 和 **CPLD** 芯片来操作那一堆的灯的。**CPLD** 可以实现锁存等功能，**XINTF** 映射到固定的三个区域，他们可以配置为不同的等待状态数，建立及保持时序。通过拉低不同的片选信号，可以操作不同的区域。而且在该款芯片中，外设寄存器所在的存储区域都设有相应的硬件保护，防止顺序颠倒，也就是“其后紧跟读的写操作流水线保护”。

这个试验，我们还做了一些拓展，也就是在线改变灯的模式：先进入 **debug** 模式，并全速运行；接着把 **switch** 中的那个选择模式的变量放到在线查看的窗口中，并查看他的地址，然后根据这个地址在反汇编

和 **memory** 窗口中找到相应的内容，最后利用 **fill memory** 来在线更改内存中的值，这样就可以在线更改灯的模式了。



浙江工业大学

实验八： 快速傅里叶变换(FFT)算法

指导教师： 吴春

班 级： 电气 1701 班

姓 名： 黄林志

学 号： 201706060301

浙江工业大学

实验八： 快速傅里叶变换(FFT)算法

八、 实验目的

9. 加深对 DFT 算法原理和基本性质的理解;
10. 熟悉 FFT 的算法原理和 FFT 子程序的算法流程和应用;
11. 学习用 FFT 对连续信号和时域信号进行频谱分析的方法

九、 实验内容

1. 实验步骤

18) 实验准备

拷贝 FFT 文件夹到 D 盘 test 目录下(参考前面实验)

19) 打开 CCS, 进入 CCS 的操作环境。

20) FFT 工程, 添加 28335.gel 文件

21) 设置仿真器

22) 阅读源程序, 理解整个程序运行过程

23) 打开 FFT.c 文件, 到第 34 行, 可以修改宏定义 SAMPLELONG。

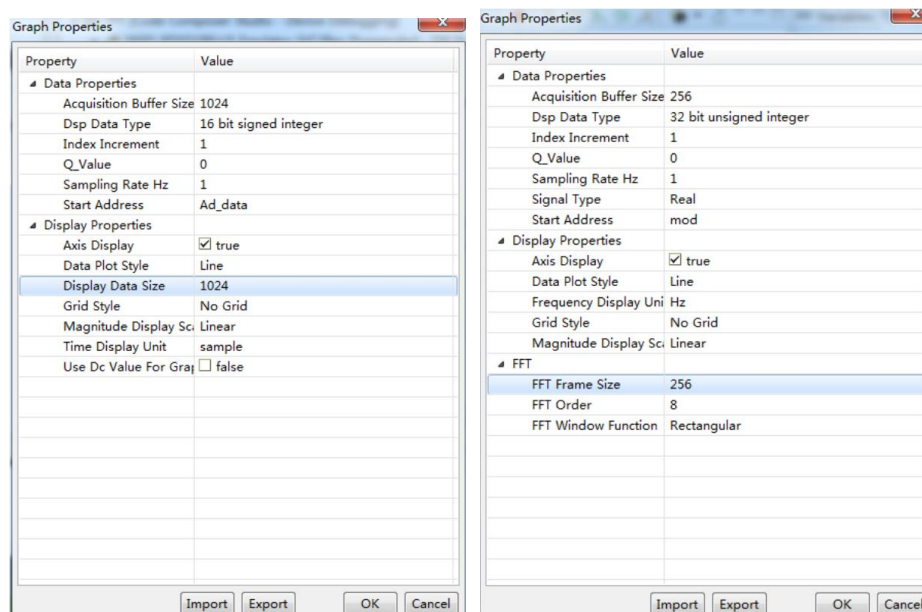
SAMPLELONG 是采样长度选择, 有 3 个选择 1、2、3。1 表示 256, 2 表示 512, 3 表示 1024。

24) 本实验和 AD 实验, 数字滤波实验都需要设置实验箱信号源。通过液晶和按键, 设置信号源。菜单路径为: “系统设置” — “信号发生器设置”。在 “信号发生器设置” 这一菜单下:

- a) “通道” 设为 “0”;
- b) “信号类型” 可根据需要任意选择; 本实验选择标准正弦波;
- c) “信号频率” 和 “信号振幅” 可在屏幕下方 “有效输入” 限定的范围内任意输入, “信号振幅” 设为 1000 左右, “信号频率” 设为 300 左右; “信号发生器开关” 设为 “开启”。

8) 运行程序。当程序执行到断点时, 可以观察收到的数据和显示的图像。

其中图像显示设置对话框中 Start address: 起始地址; Acquisition Buffer Size: 输入数据个数; Display Data Size: 显示数据个数。



2. 代码修改

三、实验背景知识

傅立叶变换是一种将信号从时域到频域的变换形式，是声学、语音、电信和信号处理等领域中的一种重要分析工具。离散傅立叶变换（DFT）是连续傅立叶变换在离散系统中的表现形式，由于 DFT 的计算量很大，因此在很长时间内其应用受到很大的限制。快速傅立叶变换（FFT）是离散傅立叶变换的一种高效运算方法。FFT 使 DFT 的运算大大简化，运算时间一般可以缩短一至两个数量级，FFT 的出现大大提高了 DFT 的运算速度，从而使 DFT 在实际应用中得到广泛的应用。在数字信号处理系统中，FFT 作为一个非常重要的工具经常使用，它甚至成为 DSP 运算能力的一个考核因素。对于有限长离散数字信号 $\{x[n]\}$, $0 \leq n \leq N-1$, 其离散谱 $\{x[k]\}$ 可以由离散付氏变换（DFT）求得。

DFT 的定义为

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{-j(\frac{2\pi}{N})nk} \quad k = 0, 1, \dots, N-1$$

可以方便的把它改写为如下形式：

$$X(k) = \sum_{n=0}^{N-1} x[n] W_N^{nk} \quad k = 0, 1, \dots, N-1$$

$$X(k) = \sum_{n=0}^{N-1} x[n] W_N^{nk} \quad k = 0, 1, \dots, N-1$$

$$W_N = e^{-j2\pi/N}$$

即 称为蝶形因子式旋转因子。

对于旋转因子 W_N 来说，有如下的对称性和周期性：

对称性： $W_N^k = -W_N^{k+N/2}$ ；

周期性： $W_N^k = W_N^{k+N}$ 。

FFT 就是利用了旋转因子的对称性和周期性来减少运算量的。

FFT 算法将长序列的 DFT 分解为短序列的 DFT。N 点的 DFT 先分解为两个 N/2 点的 DFT，每个 N/2 点的 DFT 又分解为两个 N/4 点的 DFT 等等，最小变换的点数即基数，基数为 2 的 FFT 算法的最小变换是 2 点 DFT。

一般而言，FFT 算法分为时间抽选（DIT）FFT 和频率抽选（DIF）FFT 两大类。时间抽取 FFT 算法的特点是每一级处理都是在时域里把输入序列依次按奇/偶一分为二分解成较短的序列；频率抽取 FFT 算法的特点是在频域里把序列依次按奇/偶一分为二分解成较短的序列来计算。

DIT 和 DIF 两种 FFT 算法的区别是旋转因子 W_N^k 出现的位置不同，（DIT）FFT 中旋转因子 W_N^k 在输入端，（DIF）FFT 中旋转因子 W_N^k 在输出端，除此之外，两种算法是一样的。在本设计中实现的是基 2 的频率抽取 FFT 算法，具体的实现过程可参见源程序及其注释。

四、实验程序功能与结构说明

1. FFT 实验包含文件:

- 1) FFT.c: 实验主程序, 包含系统初始化, AD 初始化 A/D 采样, FFT 变换, 以及将 FFT 变换结果做取模运算;
- 2) DSP2833X_Adc.c: 包含了 AD 初始化。
- 3) DSP2833X_DefaultIsr.c: 包含了异步串口接收中断服务程序。
- 4) Cfft32c.asm、cfft32i.asm、rfft32br.asm、rfft32m.asm、rfft32s.asm、rfft32w.asm: TI 源代码。
- 5) DSP2833X_CpuTimers.c: 包含了 DSP 定时器的初始化。
- 6) DSP2833X_GlobalVariableDefs.c: 各个外设全局变量定义。
- 7) DSP2833X_Gpio.c: GPIO 初始化。
- 8) DSP2833X_InitPeripherals.c: 外设初始化函数。
- 9) DSP2833X_PieCtrl.c: PIE 中断初始化。
- 10) DSP2833X_PieVect.c: PIE 中断矢量表初始化。
- 11) DSP2833X_SysCtrl.c: 系统初始化。

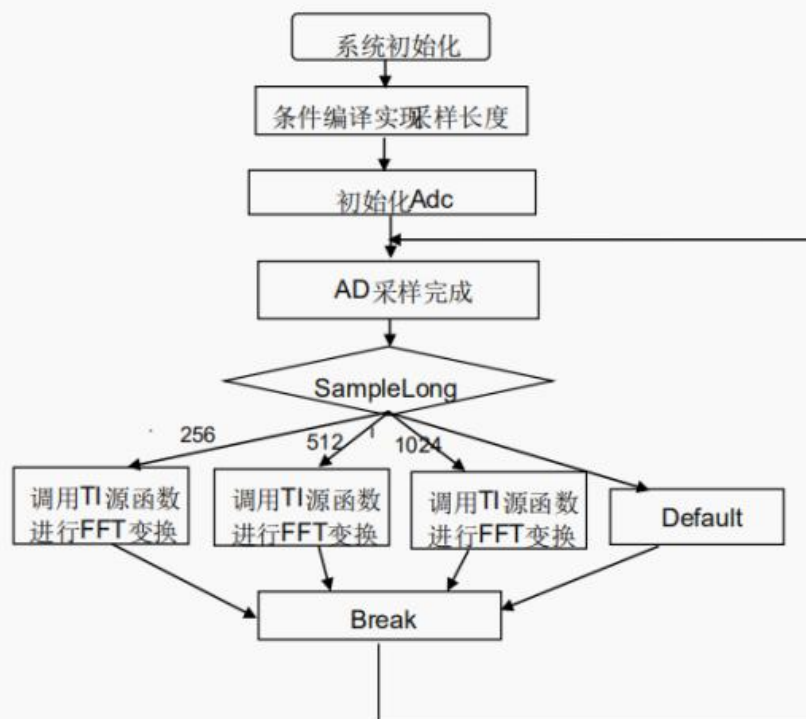
12) 28335.cmd: 声明了系统的存储器配置与程序各段的连接关系。

13) f28335.gel: 系统初始化

14) *.h: 各个源文件的头文件

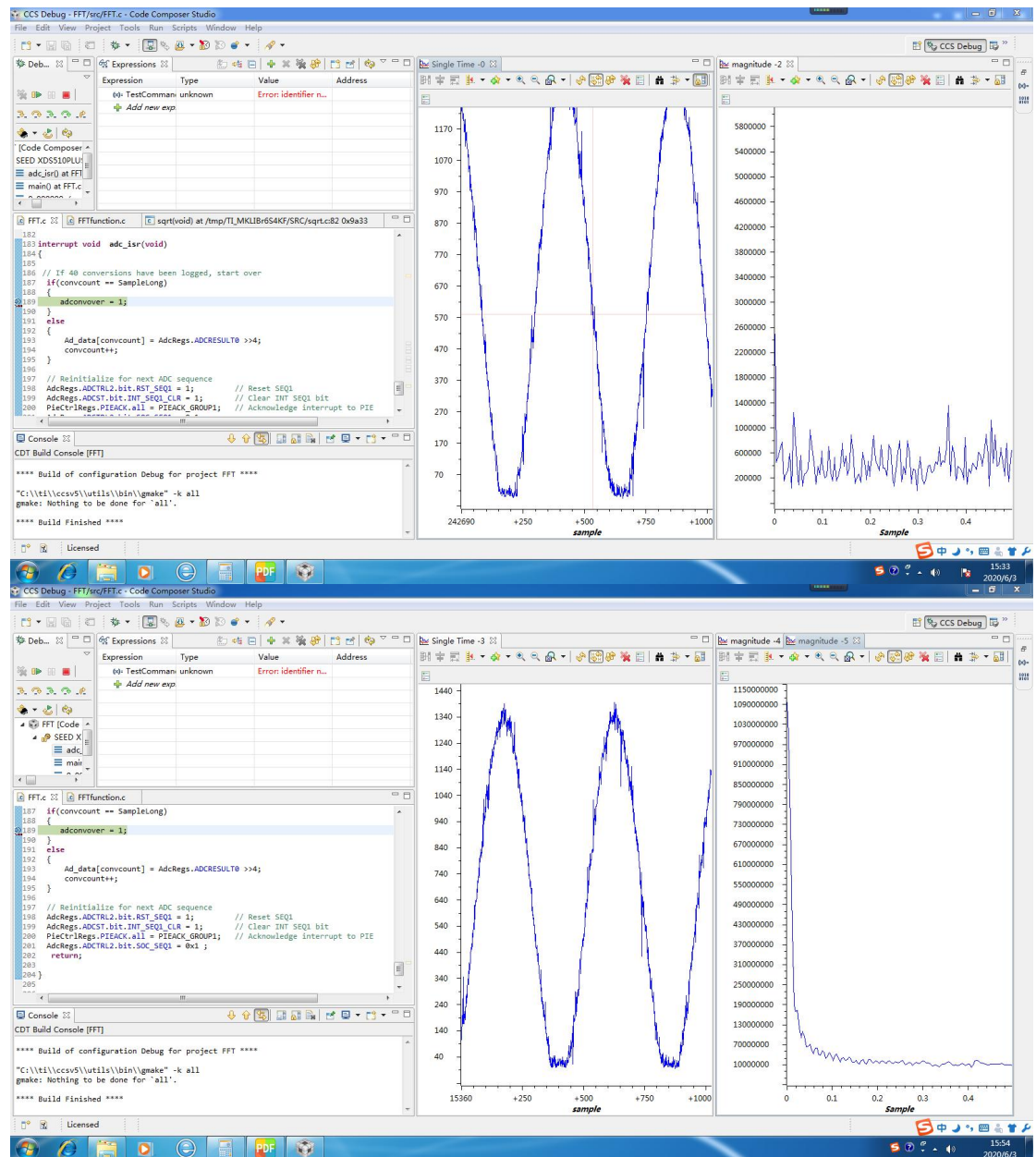
15) rts2800_fpu32.l: 库函数文件

2. 程序流程图



五、实验结论

如图所示



六、试验总结

在这个试验中我了解到了 FFT 的一些知识, 其实之前是没有怎么接触过这方面知识的。FFT: 快速傅里叶变化, 将信号从时域或空间域变换到频域, 也可以逆变换使用。把 DFT (离散傅里叶) 分解为短序列 DFT (矩阵分解为稀疏矩阵), 复杂度从 $O(n^2)$ 降到 $O(n \log n)$, 其中 n 为数据大小, FFT 和之间用 DFT 计算, 得到的结果是相同的, 甚至 FFT 还更加的准确。

从老师的提问中, 我还意识到了采样窗的大小的含义是什么, 其实是和采样保持的保持时间有关的。而且代码中的采样方式是没有采样频率的, 而是不断的进行采样, 采样并转换完成, 就会触发新的中断, 中断处理的末尾进行开启转换功能。

