

实验报告

课程名称: DSP 的原理与应用

实验名称: 实验 1.1 ICETEK-F28335-AF 教学系统简介

专业-班级: 电气 1 班 学号: 220330124 姓名: 舒晟超

实验日期: 2024 年 4 月 1 日

试验台号: _____

报告总分数: _____

教师评语:

助教签字: _____

教师签字: _____

日 期: _____

一、实验目的

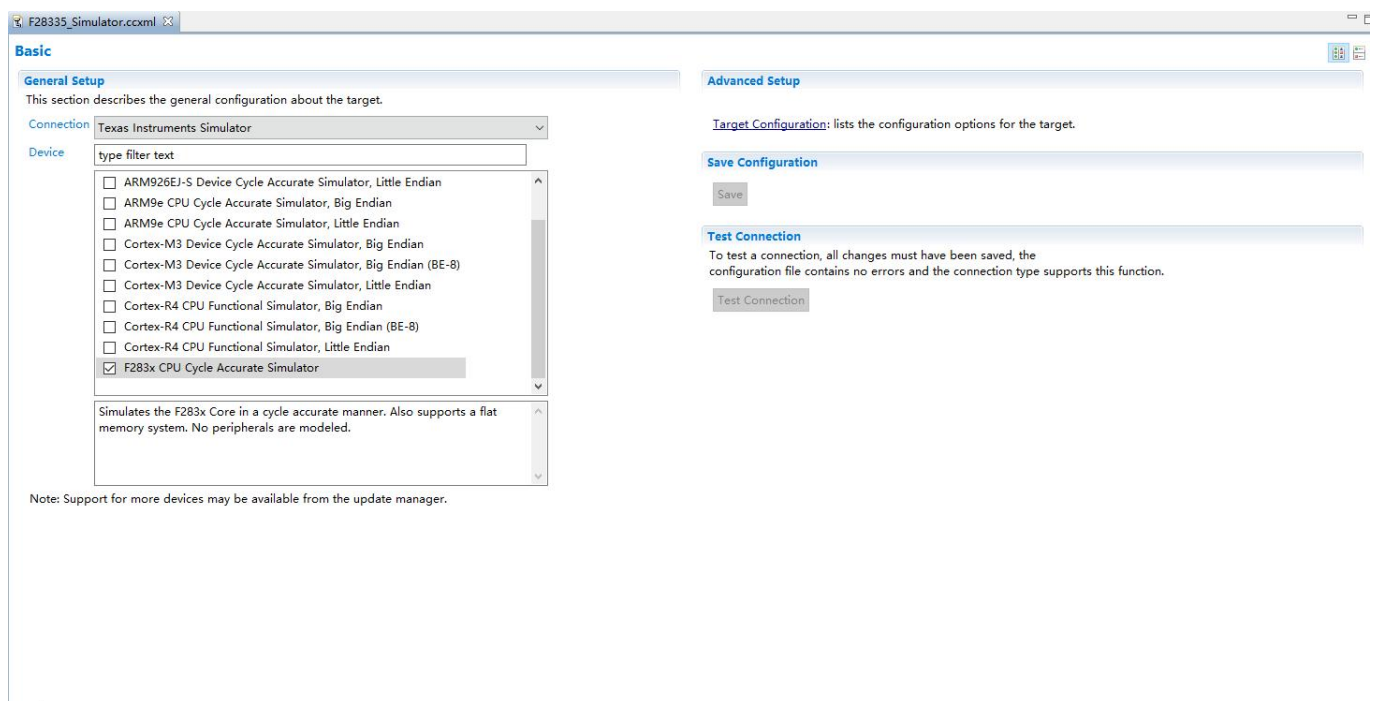
1. 学习配置 DSP 开发的软件环境，下载并设置 CCS 开发和仿真环境；
2. 学习配置硬件(Emulator)和软件(Simulator)仿真配置文件。

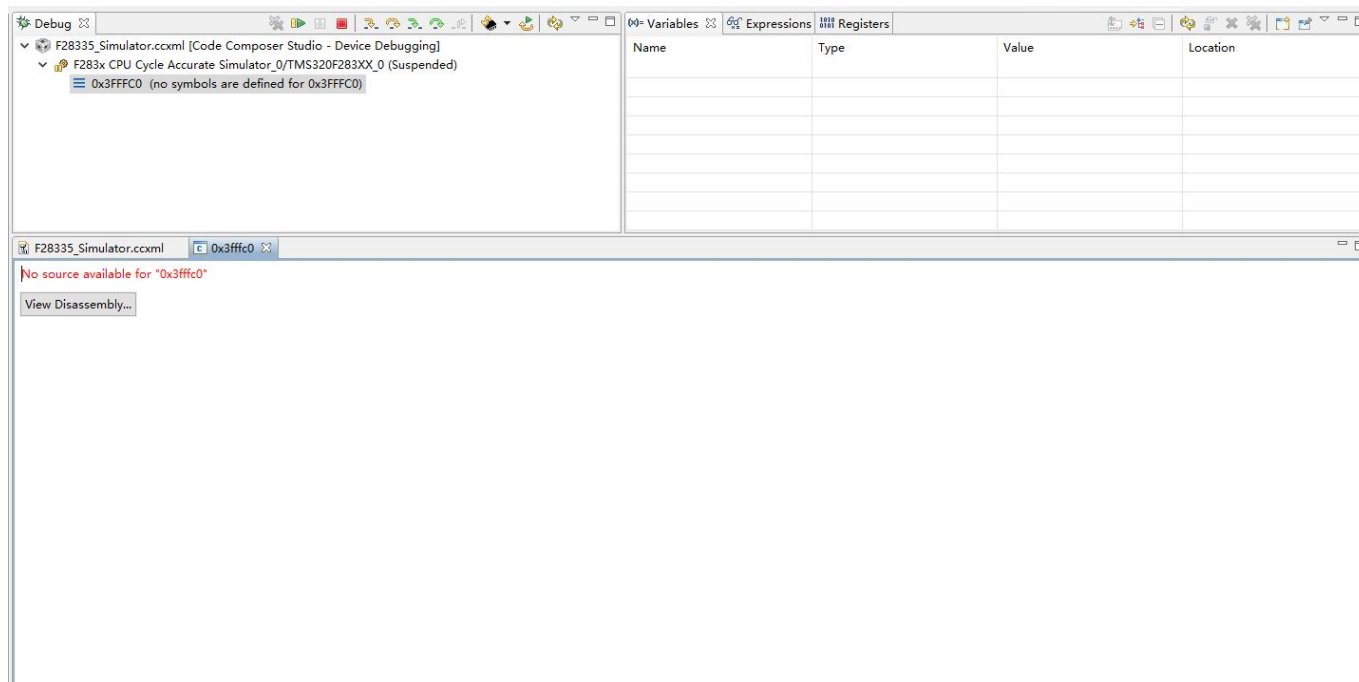
二、实验过程

- 1.正确连接 ICETEK-DSP 教学实验箱的电源和仿真器；
- 2.设置 CCS 工作空间；
- 3.配置 DSP 软件仿真文件 F28335_Simulator.ccxml 并运行；
- 4.配置 DSP 硬件仿真文件 F28335_Emulator.ccxml 并运行；

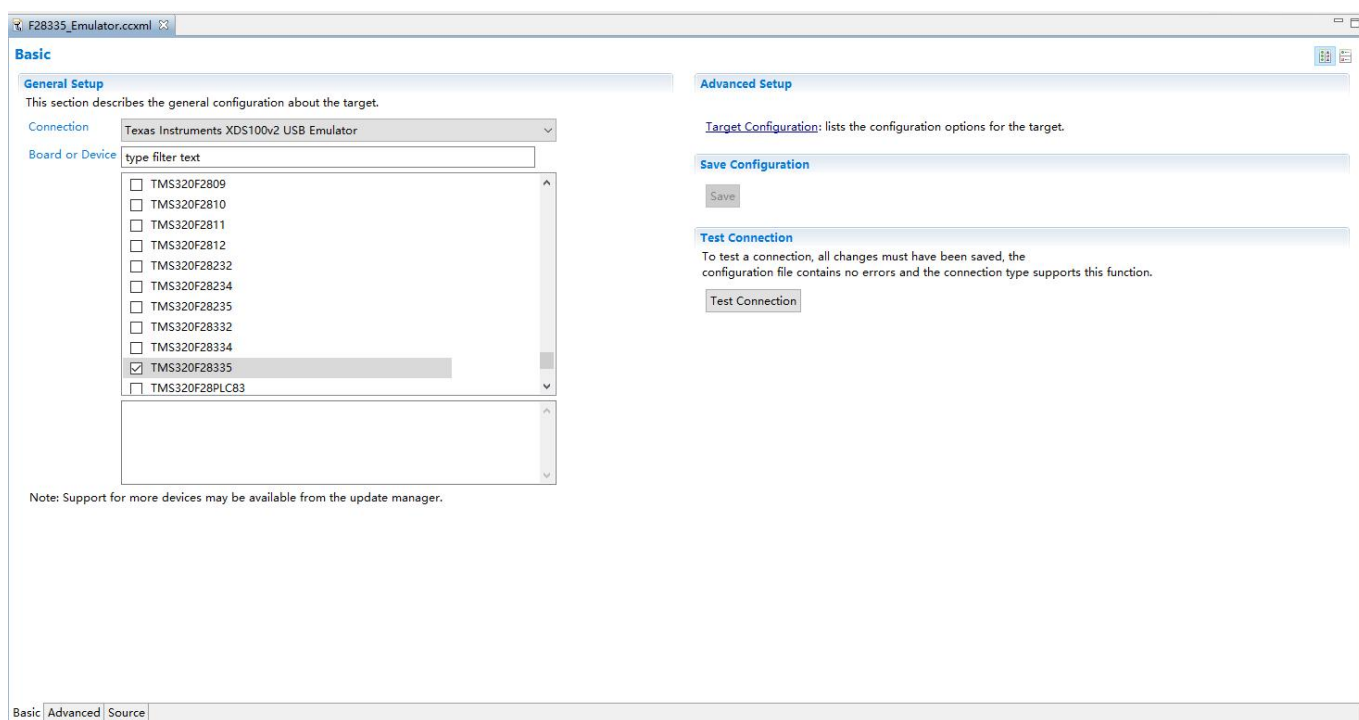
三、实验结果

- 1.软件仿真文件 F28335_Simulator.ccxml 的配置结果：

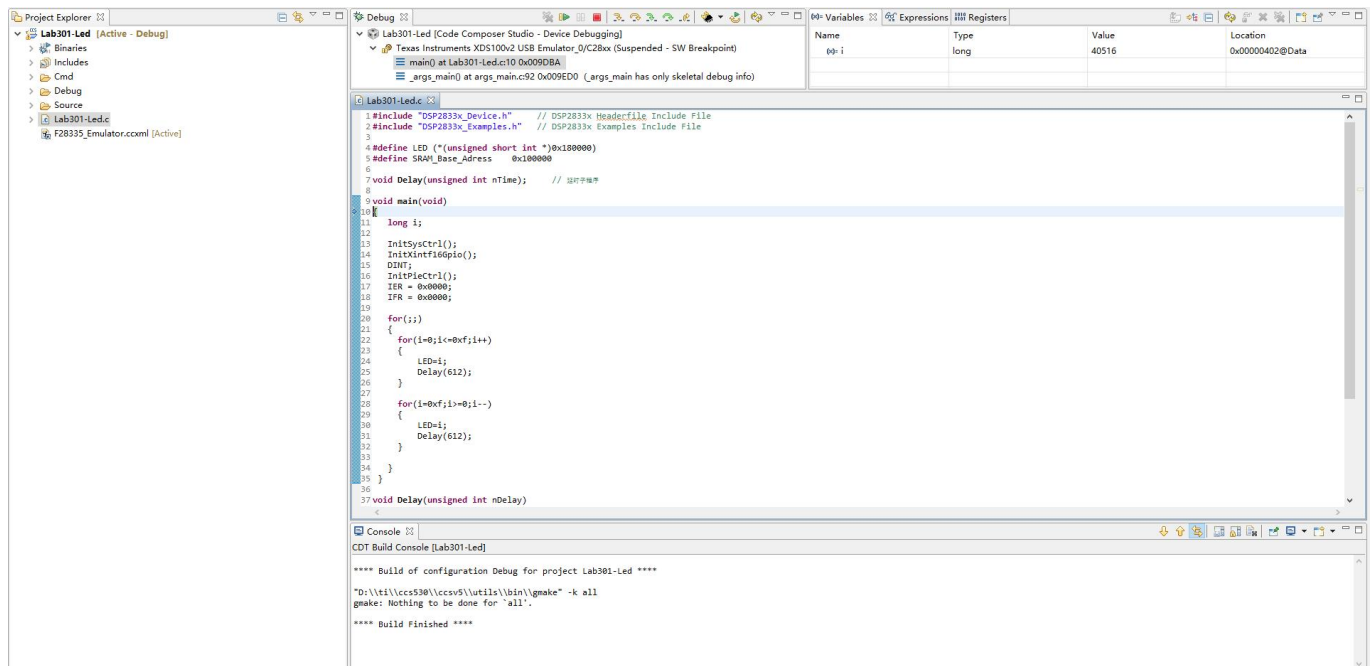




2. 硬件仿真文件 F28335_Emulator.ccxml 的配置结果



导入 Lab301-led 工程后将仿真文件替换为 F28335_Emulator.ccxml，编译完成后点击调试按钮得到以下调试界面：



点击运行按钮，可以观察到 DSP 开发板上的 4 个 LED 灯开始闪烁。同时观察到 XDS100 仿真器上的 RUN 指示灯亮起。

四、问题与思考

1、建立 Simulator Target Configuration File 和 Emulator Target Configuration File 的过程有什么不一样？这两种 Target Configuration Files 的作用分别是什么？（1 分）

建立 Simulator Target Configuration File:

1. 选择 TI 的软件仿真器 Texas Instruments Simulator
2. 选择仿真目标为 F283x CPU Cycle Accurate Simulator

建立 Emulator Target Configuration File:

1. 选择硬件调试器
2. 选择实际仿真目标 TMS320F28335

Simulator Target Configuration File 主要用于不需要硬件的调试，比如算法测试（Park 算法测试）等等，**Emulator Target Configuration File** 主要用于需要硬件的调试，比如外设调试。

不过 CCS6 及其以上版本均不支持软件调试，CCS9 及其以上版本不接受移植后的软件调试。

实验报告

课程名称: _____ DSP 的原理与应用 _____

实验名称: _____ 实验 1.2 Code Composer Studio 入门 _____

专业-班级: _____ 电气 1 班 _____ 学号: _____ 220330124 _____ 姓名: _____ 舒晟超 _____

实验日期: _____ 2024 年 4 月 1 日 _____

试验台号: _____

报告总分数: _____

教师评语:

助教签字: _____

教师签字: _____

日 期: _____

一、实验目的


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、实验过程

1. 创建一个新的 CCS 工程，向工程中添加文件；
2. 修改工程文件使其通过编译；
3. 添加断点，测试基本调试功能，使用观察窗口查看变量值；
4. 添加断点，测试文件数据流的输入和输出功能，使用图形窗口观察文件数据。

三、实验结果

1. 新建 MLab102_Create_Project 工程：

 New CCS Project

CCS Project
Create a new CCS Project.

Project name:

Output type:

☒ Use default location

Location:

Device

Family:

Variant:

Connection:

► Advanced settings







▼ Project templates and examples

type filter text

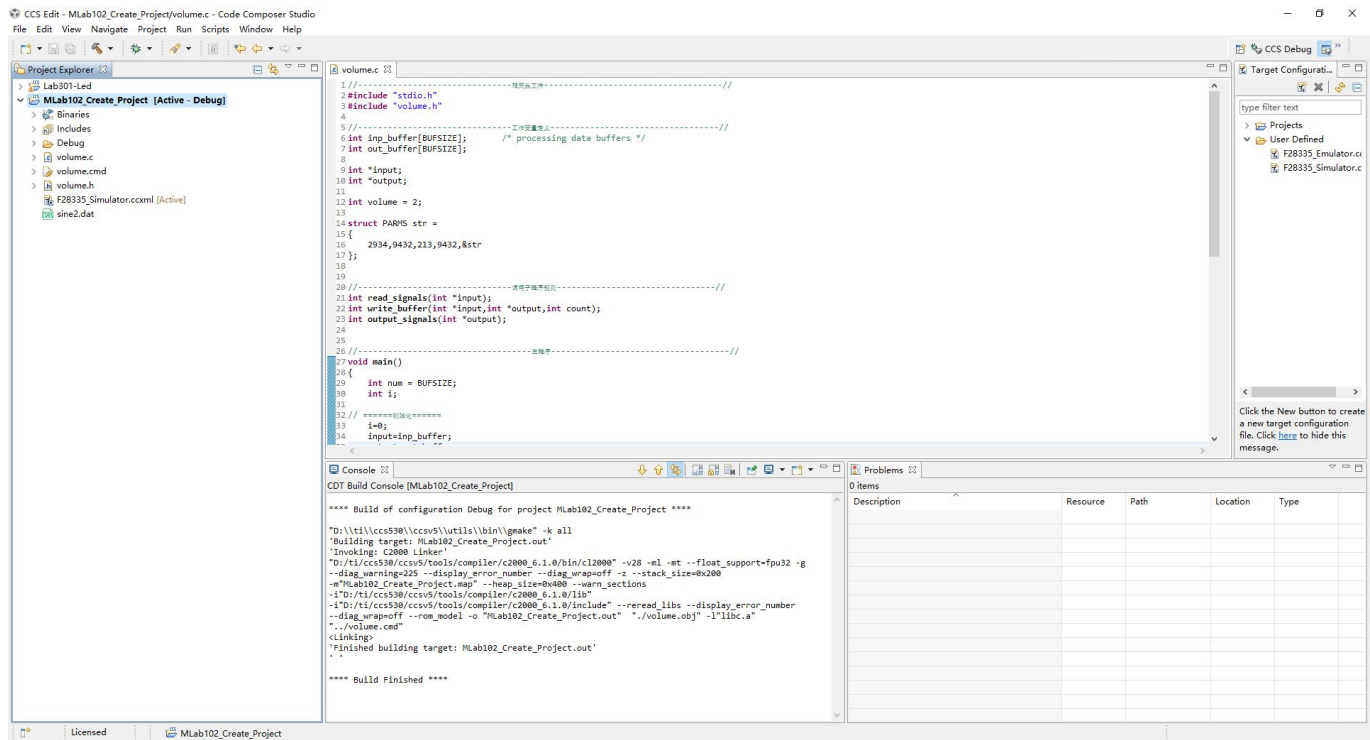
- ▼ Empty Projects
 - Empty Project
 - Empty Project (with main.c)
 - Empty Assembly-only Project
 - Empty RTSC Project
- ▼ Basic Examples
 - Hello World

Creates an empty project fully initialized for the selected device.

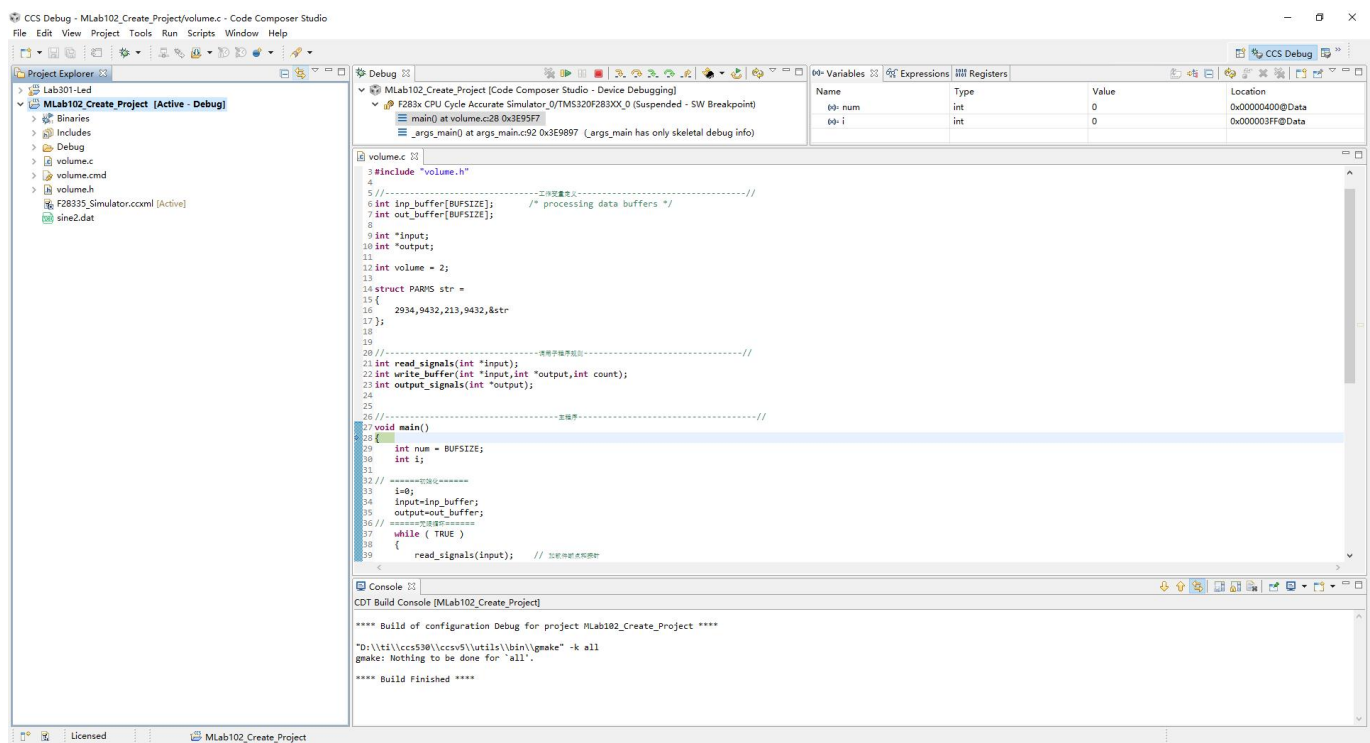
导入文件如下：

- ▼  **MLab102_Create_Project [Active - Debug]**
 - >  Includes
 - >  volume.c
 - >  volume.cmd
 - >  volume.h
 -  F28335_Simulator.ccxml [Active]
 -  sine2.dat

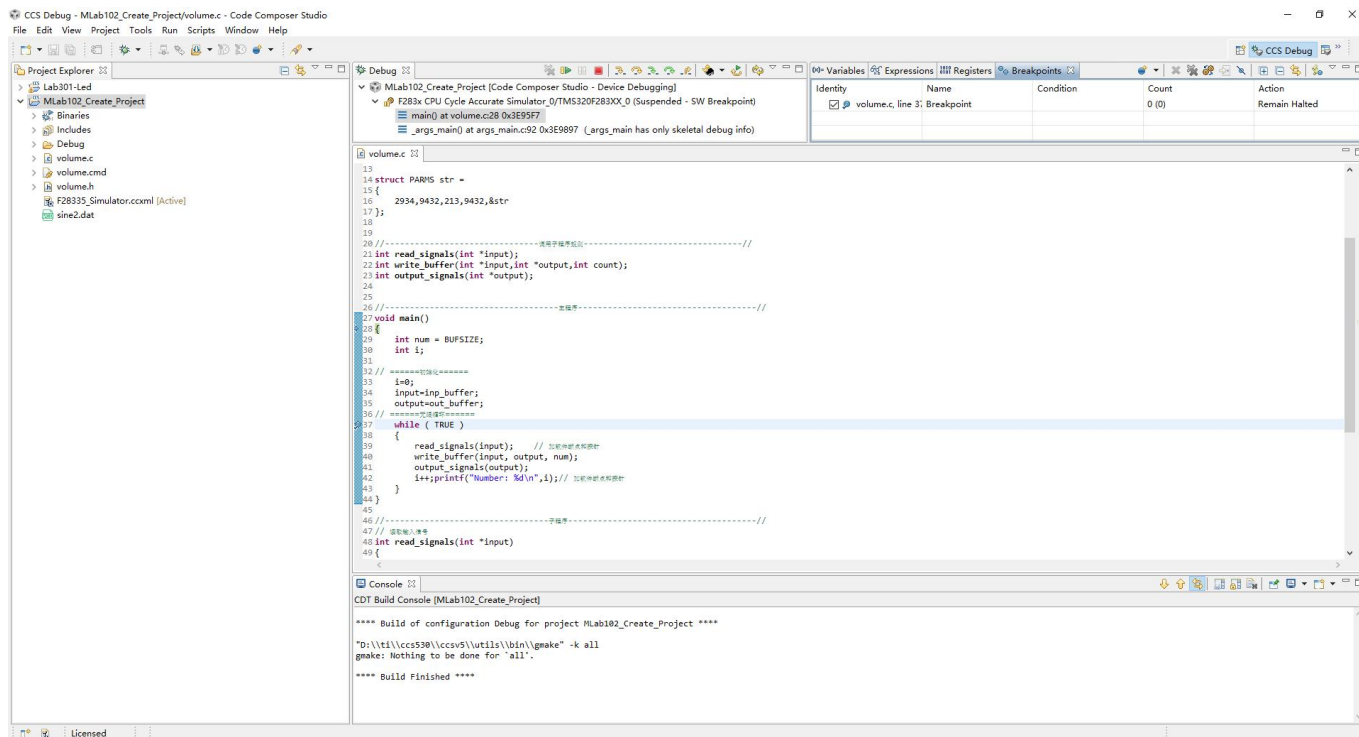
编译结果：



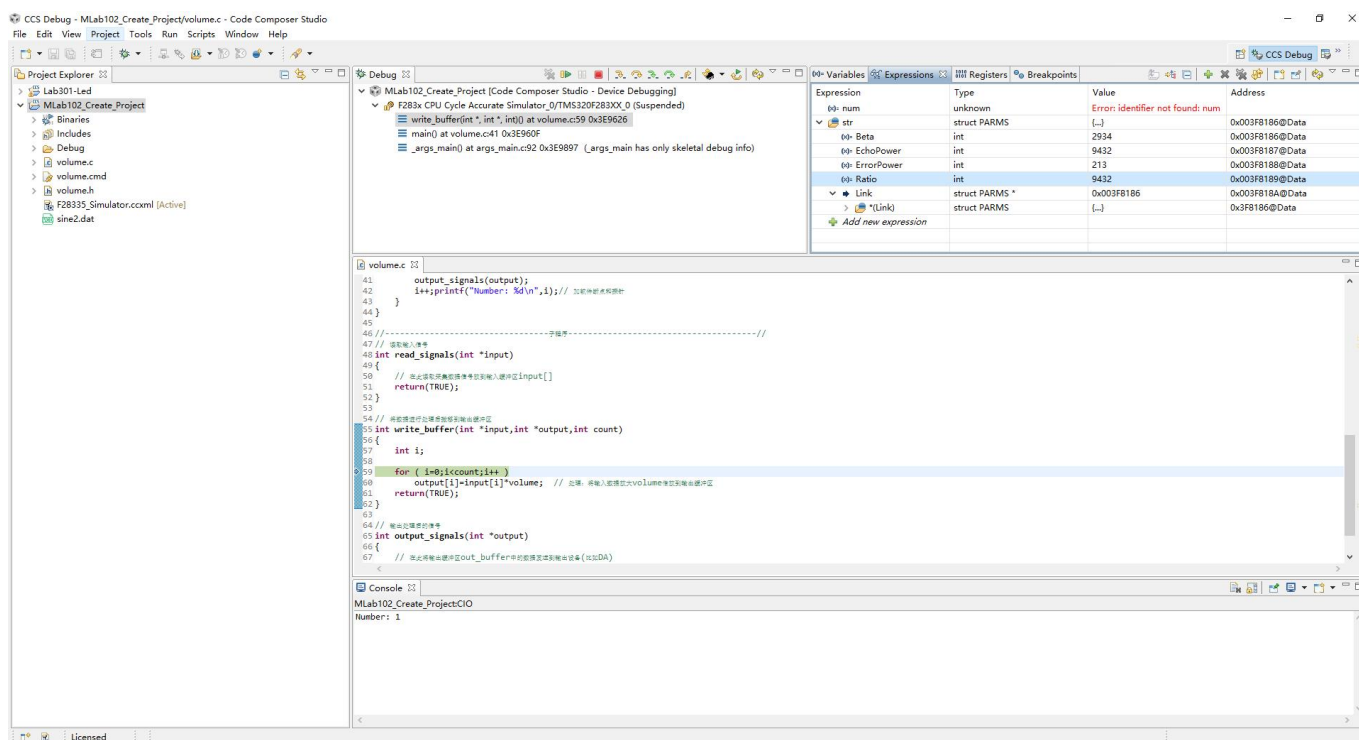
点击调试按钮开始调试：



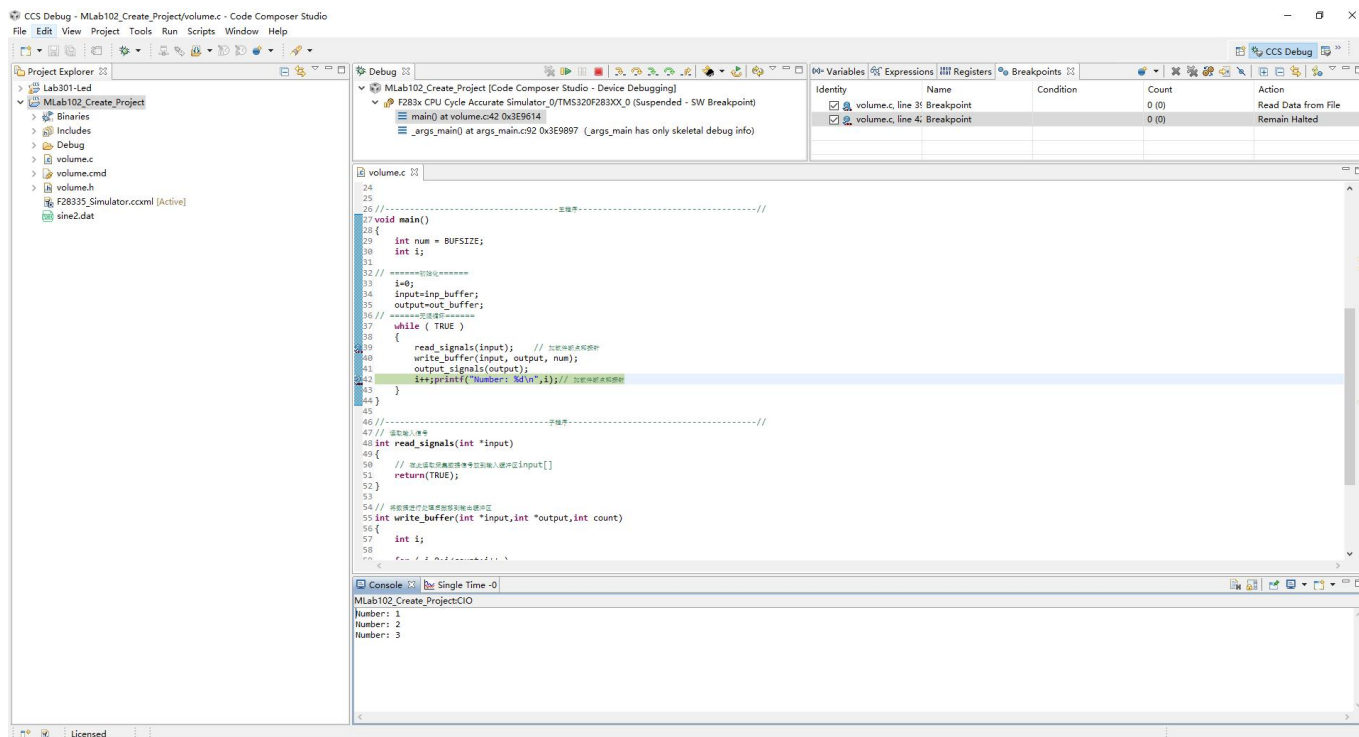
加入断点：



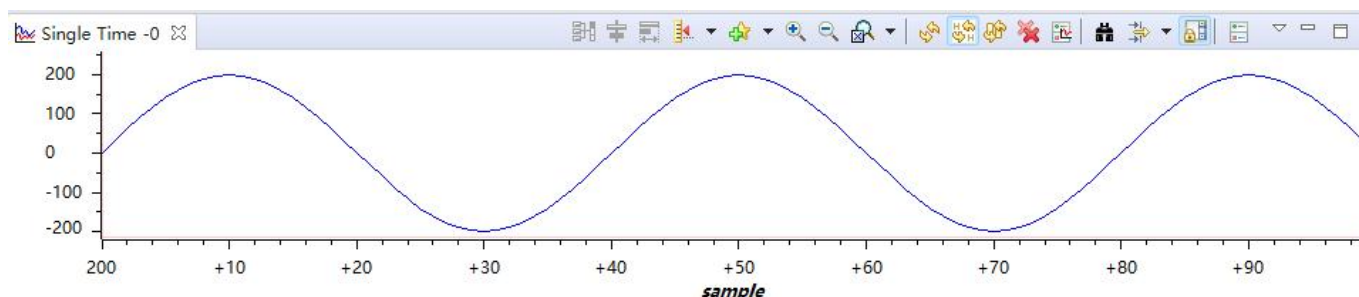
添加观察变量 num, str, 使用单步调试到 write_buffer 函数内得到以下结果:



使用断点读取 sine.dat 文件并进行调试得到以下结果:



得到波形如下：



四、问题与思考

1、Code Composer Studio 软件调试断点的作用是什么？（0.5 分）

1. 实现基本调试功能，比如运行到程序的特定位置；
2. 可以设置断点关联事件，从而实现数据流的传输以及其他 DSP 与 PC 之间的操作（读取数据流，写入数据流等）。

实验报告

课程名称: DSP 的原理与应用

实验名称: 实验 1.3 编写一个以 C 语言为基础的 DSP 程序

专业-班级: 电气 1 班 学号: 220330124 姓名: 舒晟超

实验日期: 2024 年 4 月 1 日

试验台号: _____

报告总分数: _____

教师评语:

助教签字: _____

教师签字: _____

日 期: _____

一、实验目的

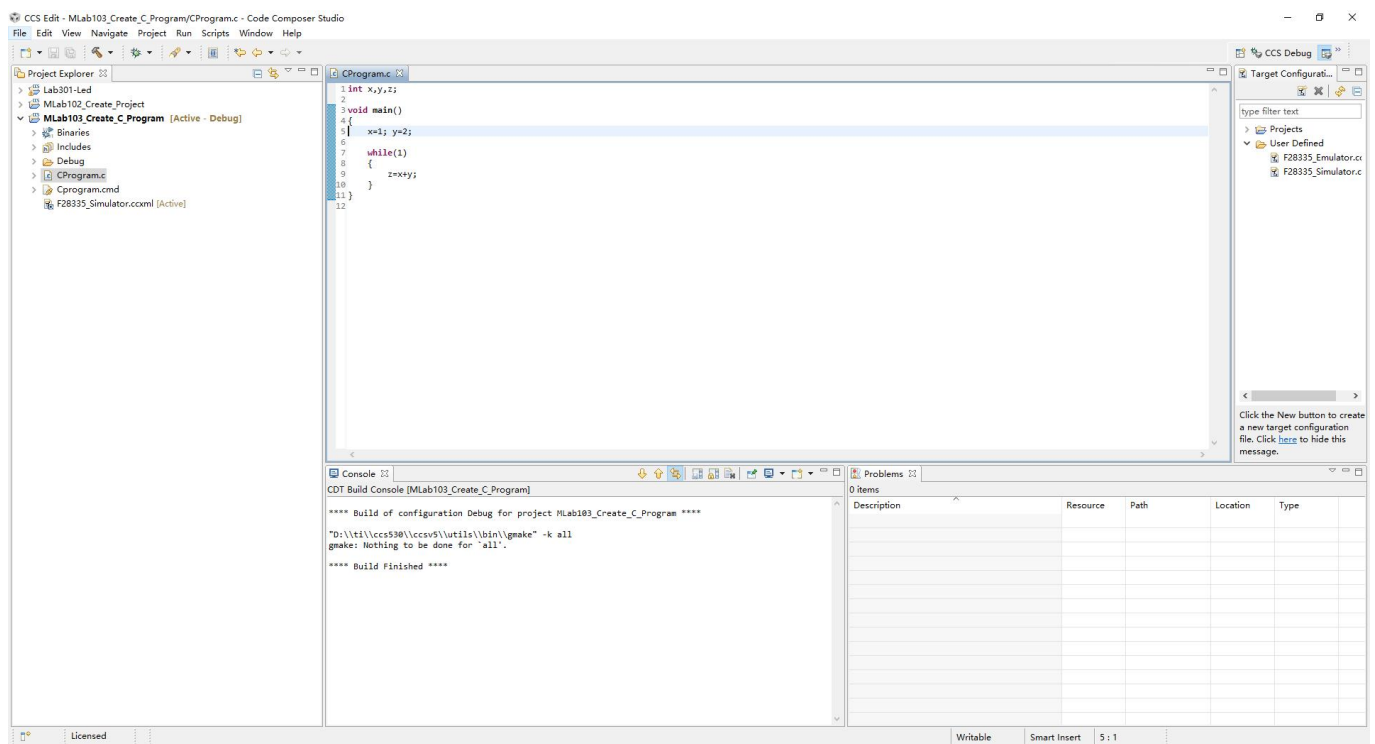
1. 学习用标准 C 语言编制程序；了解常用的 C 语言程序设计方法和组成部分。
2. 学习编制连接命令文件，并用来控制代码的连接。
3. 学会建立和改变 map 文件，以及利用它观察 DSP 内存使用情况的方法。
4. 熟悉使用软件仿真方式调试程序。

二、实验过程

1. 创建 CCS 工程并导入文件，进行编译；
2. 观察 CPU 寄存器和反汇编窗口，观察变量值和寄存器使用情况；
3. 观察.map 文件和.cmd 文件的内容。

三、实验结果

1. 创建工程并导入文件，进行编译得到以下结果



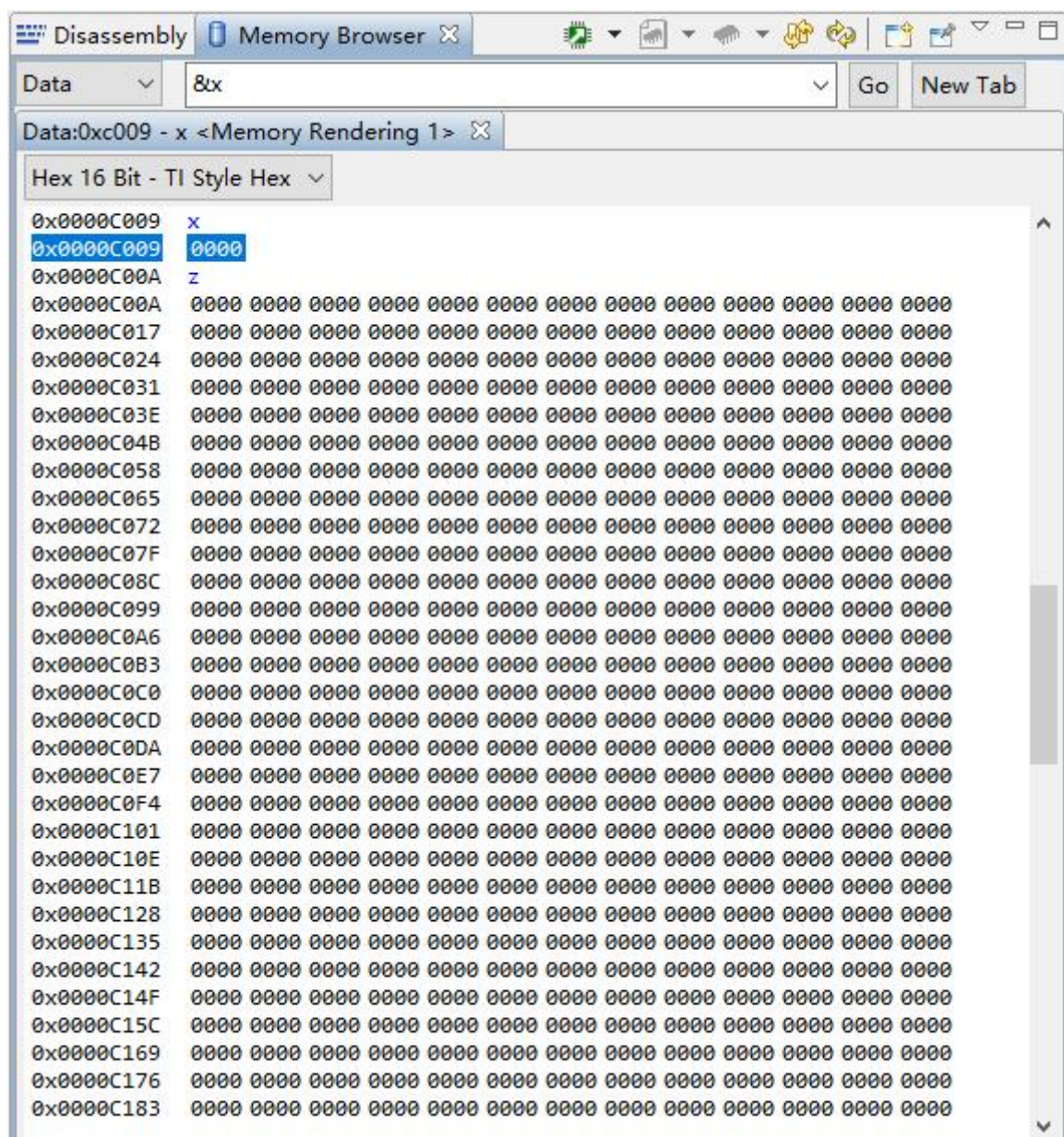
2. 进行调试，观察 CPU 寄存器得到以下结果：

Name	Value	Description
▼ CPU Registers		
> ACC	0xFFFF0000	Core
> P	0xFFFFFFFF	Core
> XT	0x00000000	Core
> XAR0	0x00000000	Core
> XAR1	0x0000FFFF	Core
> XAR2	0x00000000	Core
> XAR3	0x00000000	Core
> XAR4	0x00000000	Core
> XAR5	0x00000000	Core
> XAR6	0x00000000	Core
> XAR7	0x00008015	Core
PC	0x009078	Core
IC	0x009078	Core
RPC	0x00905F	Core

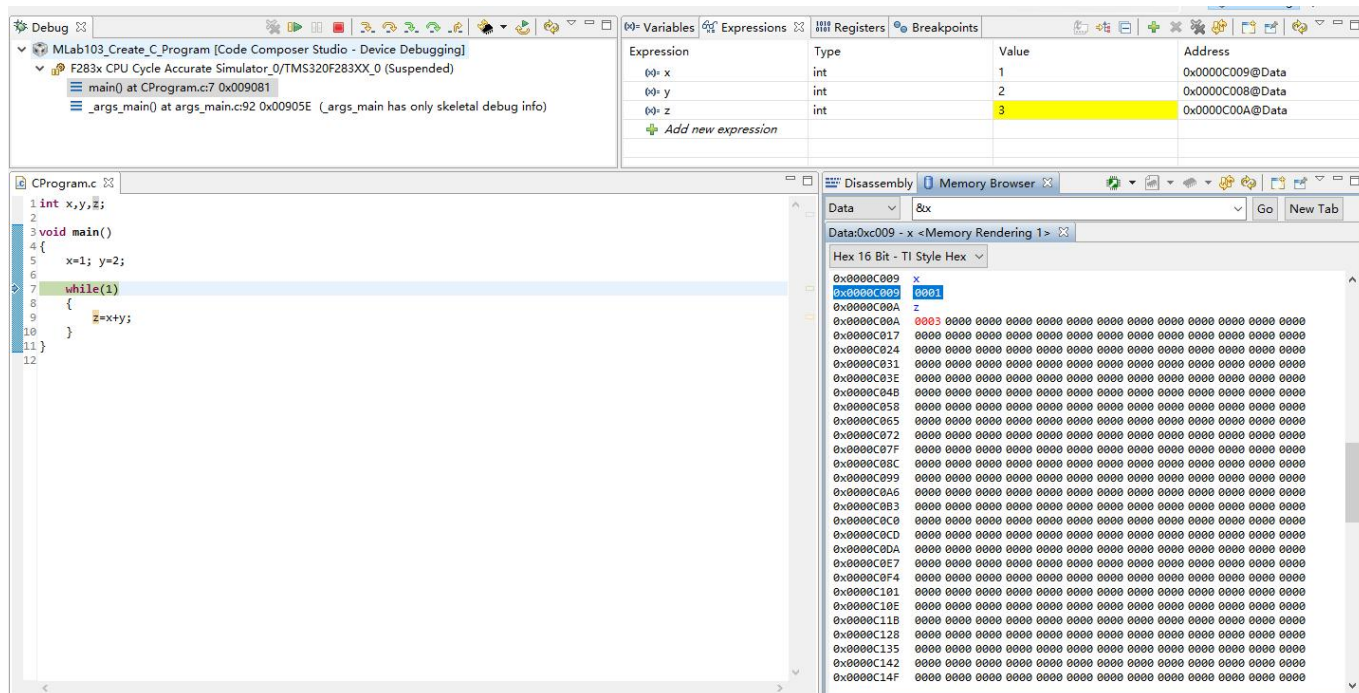
3. 使用反汇编观察窗口观察到以下结果

Disassembly		Memory Browser	
		Enter location here	
main:			
009078:	761F0300	MOVW	DP, #0x300
00907a:	56BF0109	MOVB	@0x9, #0x01, UNC
00907c:	56BF0208	MOVB	@0x8, #0x02, UNC
9			z=x+y;
C\$DW\$L\$_main\$2\$B, C\$L1:			
00907e:	9208	MOV	AL, @0x8
00907f:	9409	ADD	AL, @0x9
009080:	960A	MOV	@0xa, AL
7			while(1)
009081:	6FFD	SB	C\$L1, UNC
C\$DW\$L\$_main\$2\$E, _register_unlock:			
009082:	761F0300	MOVW	DP, #0x300
009084:	A800	MOVL	@0x0, XAR4
009085:	0006	LRETR	
_register_lock:			
009086:	761F0300	MOVW	DP, #0x300
009088:	A802	MOVL	@0x2, XAR4
009089:	0006	LRETR	
_nop:			
00908a:	0006	LRETR	
_etext, etext:			
00908b:	0000	ITRAP0	
00908c:	0000	ITRAP0	
00908d:	0000	ITRAP0	
00908e:	0000	ITRAP0	
00908f:	0000	ITRAP0	
009090:	0000	ITRAP0	
009091:	0000	ITRAP0	
009092:	0000	ITRAP0	
009093:	0000	ITRAP0	
009094:	0000	ITRAP0	
009095:	0000	ITRAP0	
009096:	0000	ITRAP0	
009097:	0000	ITRAP0	
009098:	0000	ITRAP0	

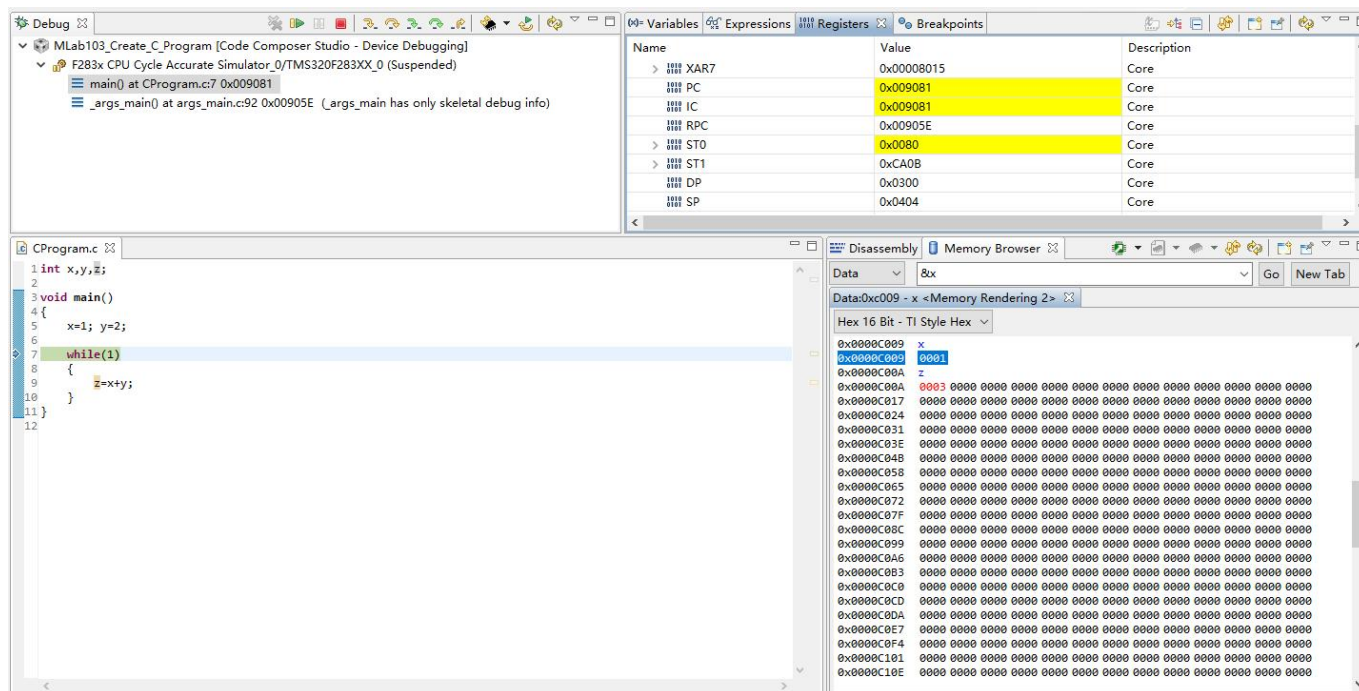
4. 内存观察窗口观察到以下结果:



5. 运行程序, 观察到 x,y,z 的变化:



6. 重新运行程序， PC, IC, ST0 寄存器参加运算，进入循环后只有 PC, IC 寄存器参加运算



7. 观察.map 文件，看到程序的入口地址和内存使用情况

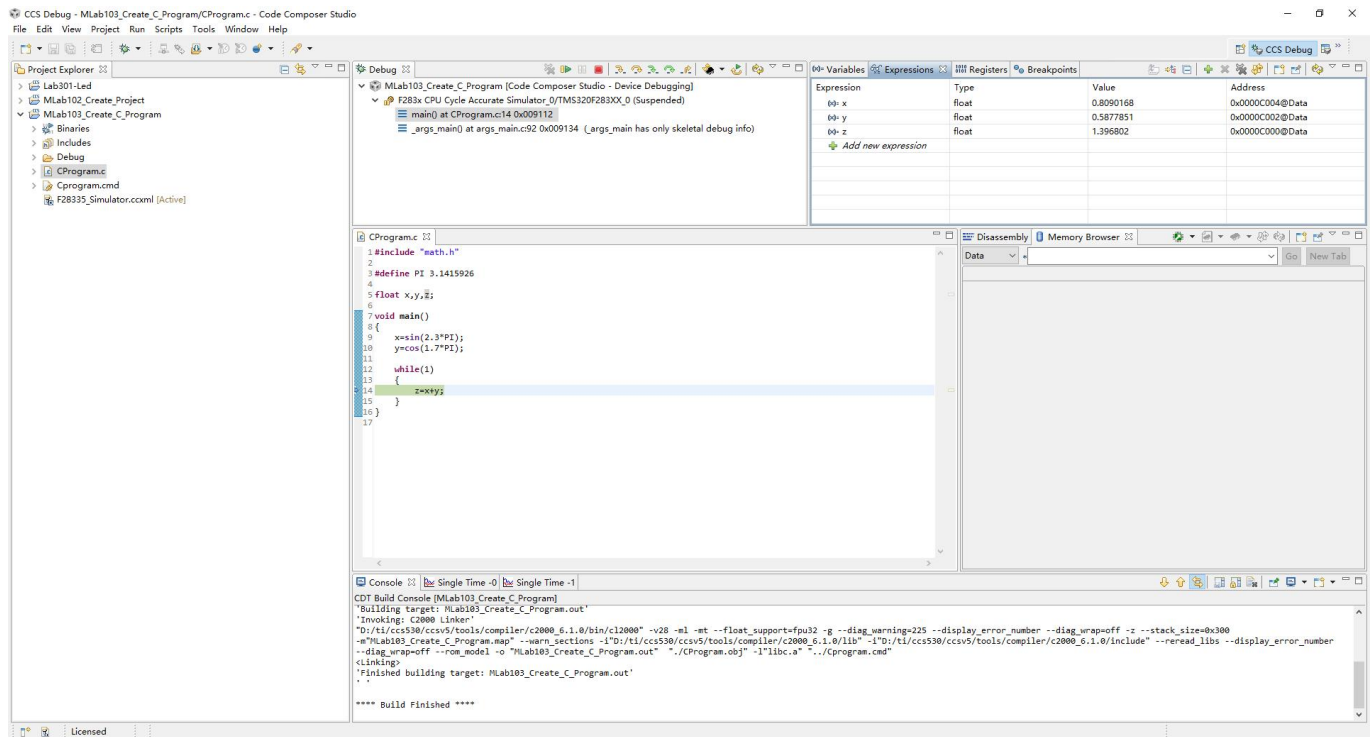
output section	page	origin	length	attributes/ input sections
.pinit	0	00008000	00000000	UNINITIALIZED
.cinit	0	00008000 00008000 0000800a 00008014	00000016 0000000a 0000000a 00000002	rts2800_fpu32.lib : _lock.obj (.cinit) : exit.obj (.cinit) --HOLE-- [fill = 0]
.text	0	00009000 00009000 00009046 0000905f 00009078 00009082	0000008b 00000046 00000019 00000019 0000000a 00000009	rts2800_fpu32.lib : boot.obj (.text) : args_main.obj (.text) : exit.obj (.text) CProgram.obj (.text) rts2800_fpu32.lib : _lock.obj (.text)
csm_rsvd	0	0033ff80	00000000	DSECT
csmpasswd*	0	0033ffff	00000000	DSECT
.reset	0	003fffc0 003fffc0	00000002 00000002	DSECT rts2800_fpu32.lib : boot.obj (.reset)
.stack	1	00000400 00000400	00000400 00000400	UNINITIALIZED --HOLE--
.ebss	1	0000c000 0000c000 0000c004 0000c008	0000000b 00000004 00000004 00000003	UNINITIALIZED rts2800_fpu32.lib : _lock.obj (.ebss) : exit.obj (.ebss) CProgram.obj (.ebss)

.cmd 文件的空间分配决定了程序和数据在 DSP 内存资源中的分配和位置；

.map 文件中描述了程序和数据所占用的实际尺寸和地址。

四、问题与思考

- 1、请修改程序完成计算 $\sin(2.3\pi) + \cos(1.7\pi)$ 的值。（0.5 分）



程序如上图所示，计算结果为 **1.396802**
实际上，可以使用 **IQmath** 库从而提高运算效率。