哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验报告

课程名称：DSP 的原理与应用

实验名称：实验 2.1 DSP 数据存取实验

专业-班级：电气 1 班 学号：220330124 姓名：舒晟超

实验日期：2024 年 5 月 7 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.了解 TMS320F28335 的内部存储器空间的分配及指令寻址方式。

2.了解 ICETEK-F28335-AF 评估板扩展存储器空间寻址方法，及其应用。

3.了解 ICETEK-F28335-AF 实验箱扩展存储器空间寻址方法，及其应用。

4.学习用 Code Composer Studio 修改、填充 DSP 内存单元的方法。

5.学习操作 TMS32028xx 内存空间的指令。

二、实验过程

1. 导入工程，进行编译；

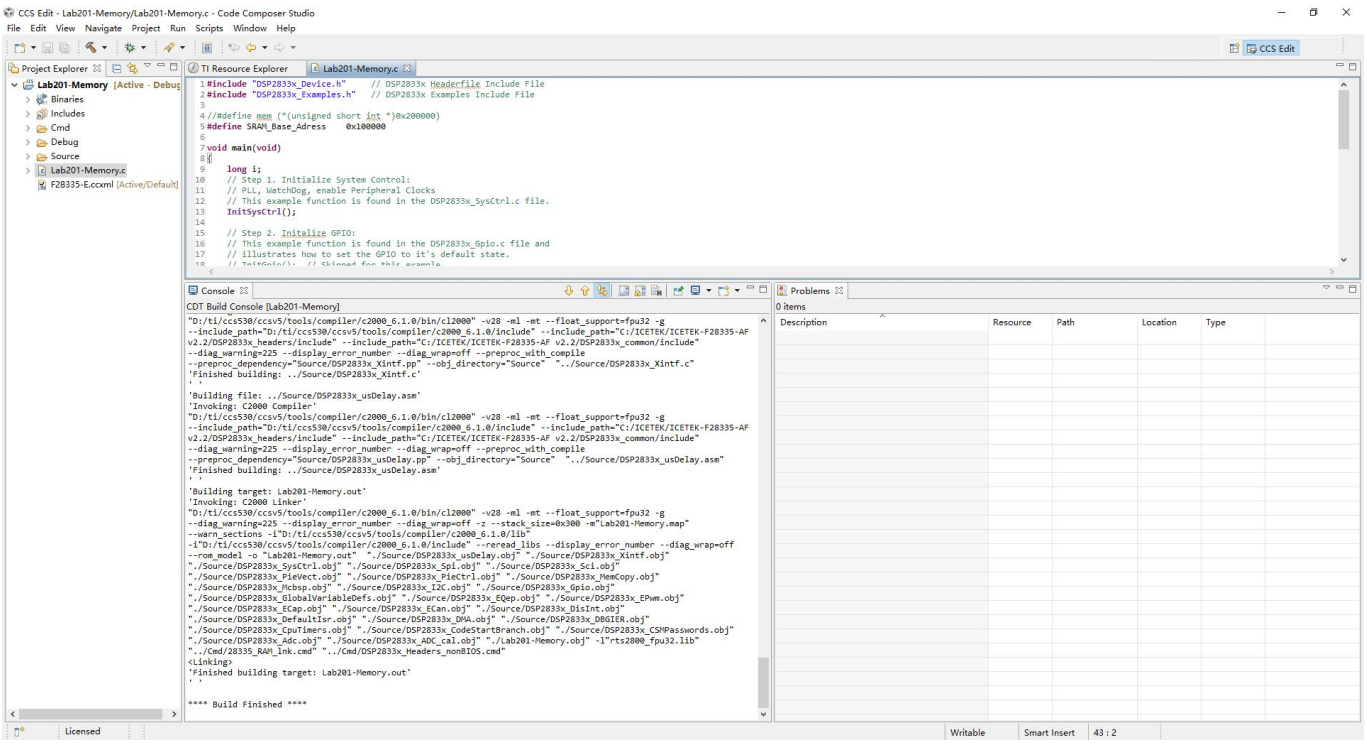
2. 打开反汇编窗口，查看函数入口程序；

3. 打开内存观察窗口，观察 SRAM 处的存储数据；

4. 添加断点，观察 SRAM 内的数据随程序运行的变化。

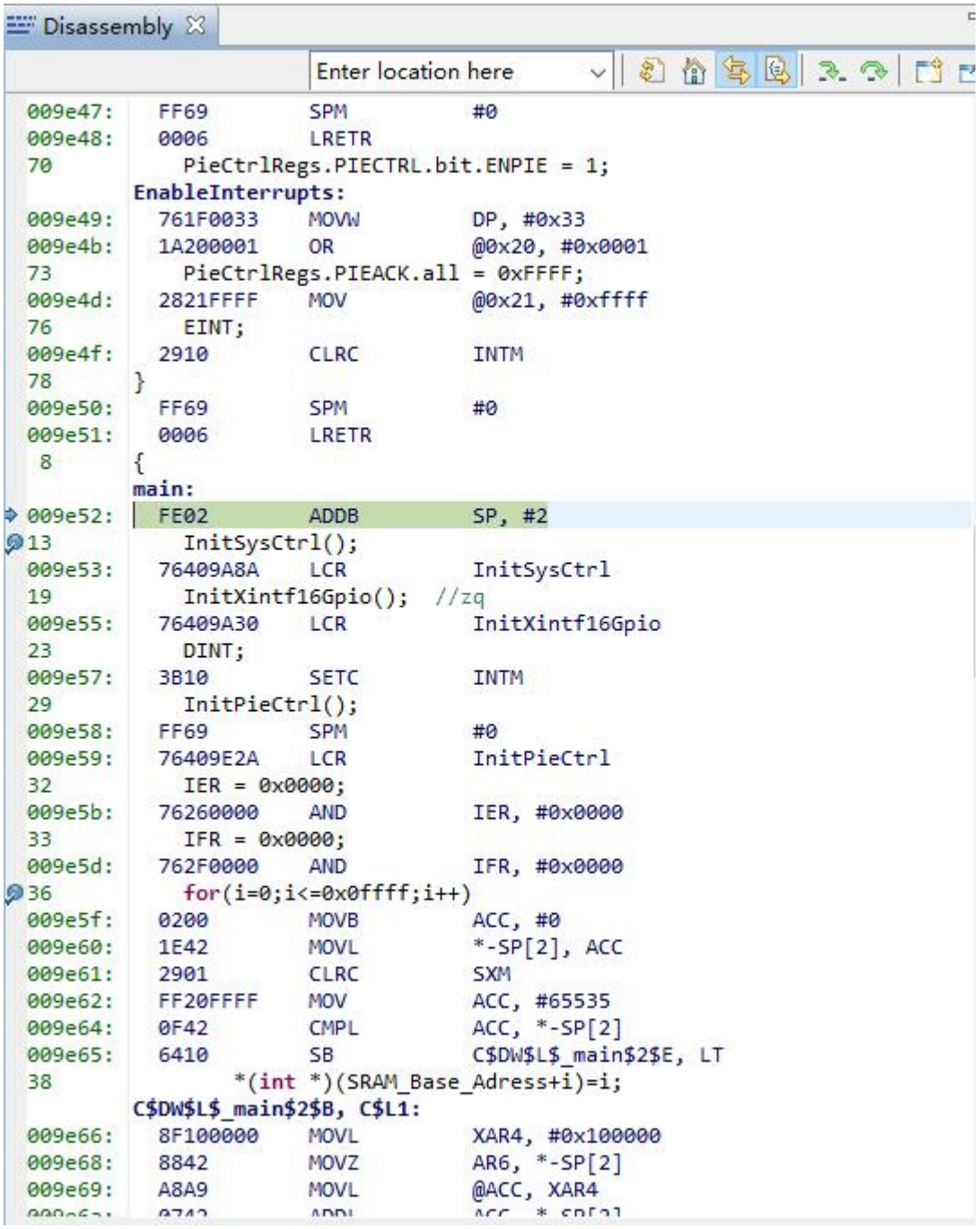
三、实验结果

1. 导入工程，进行编译；

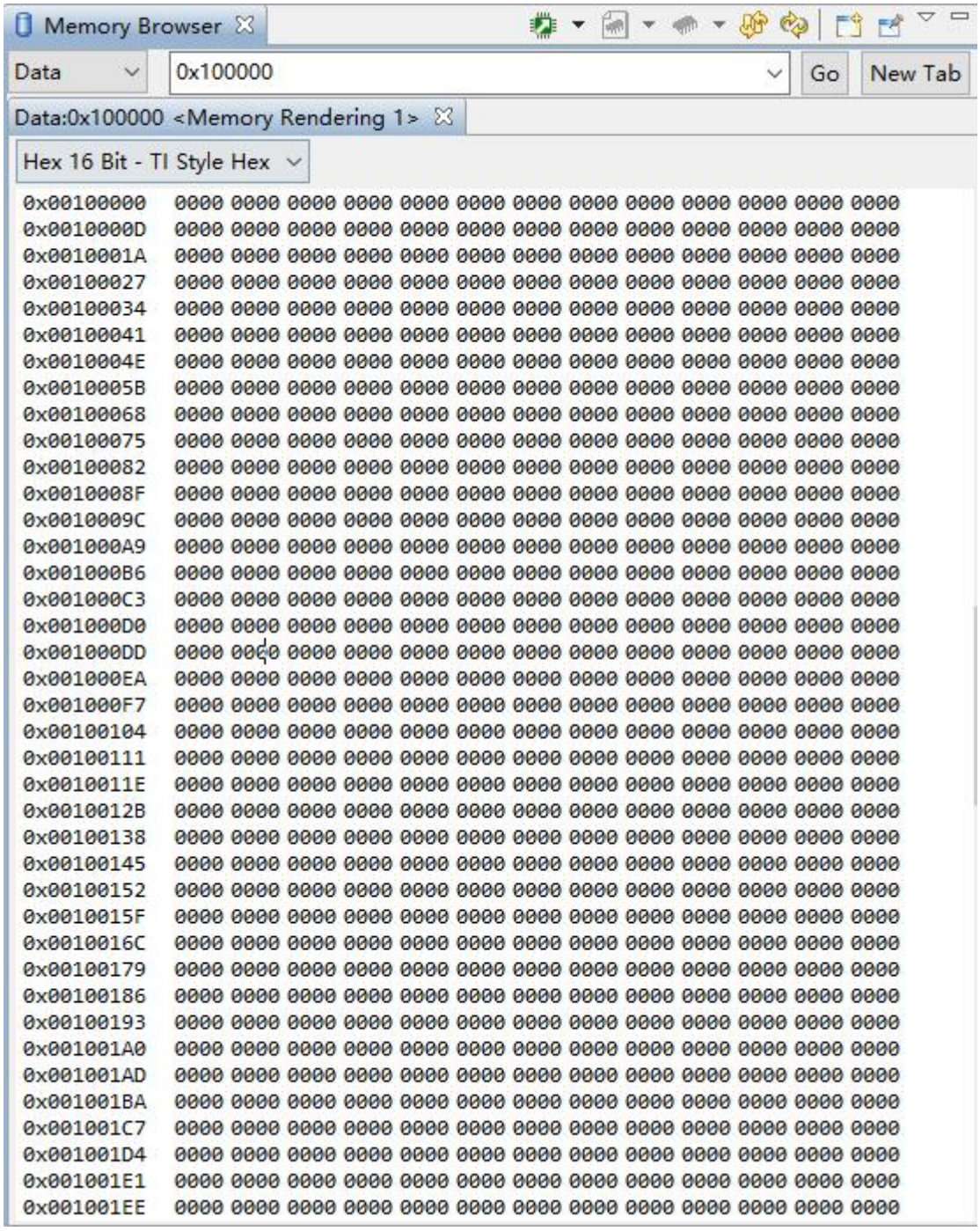


2. 添加断点，调试程序：

（1） main 函数入口地址为 FE02：

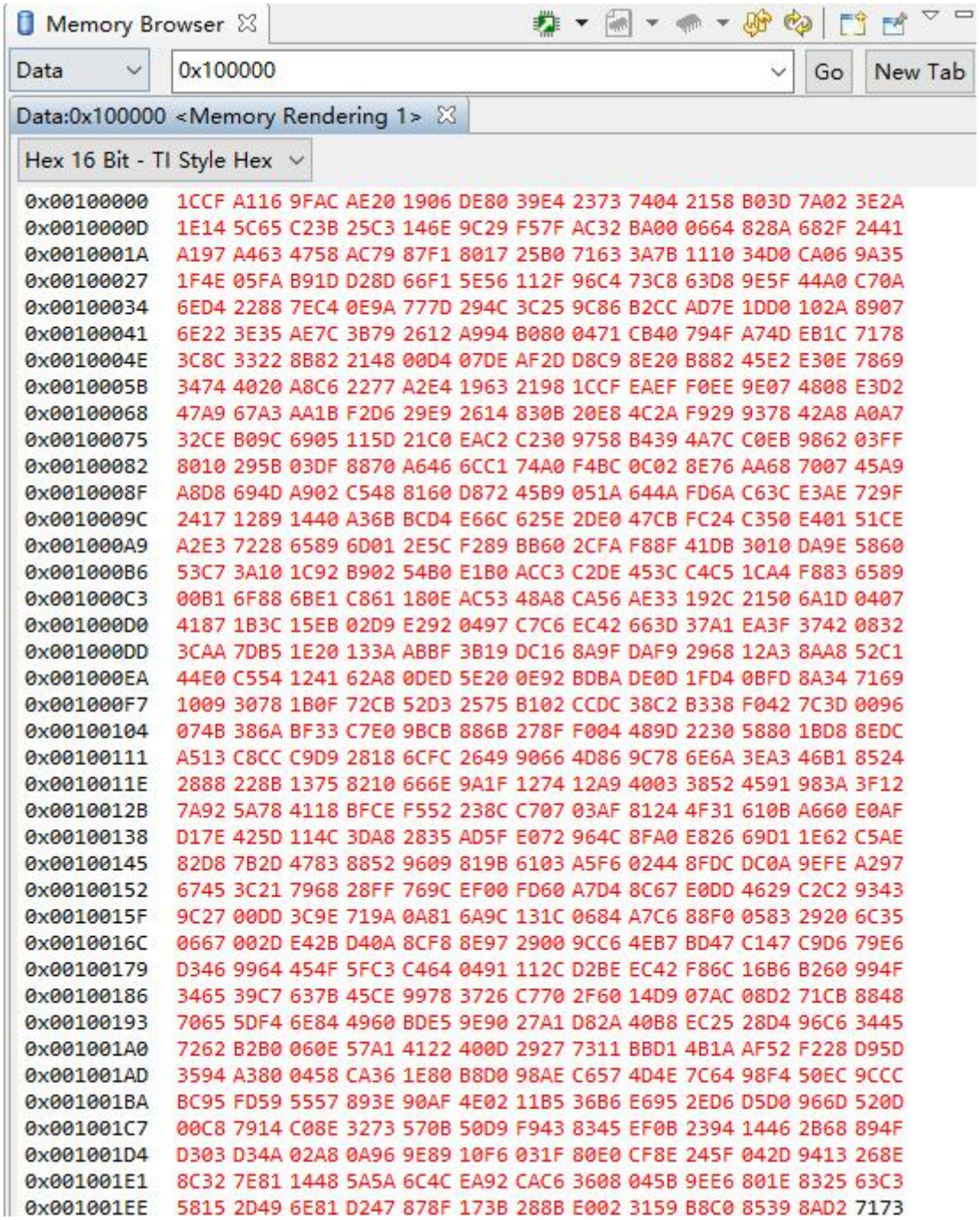


（2） XINTF 总线未使能前，SRAM 内数据如下图：



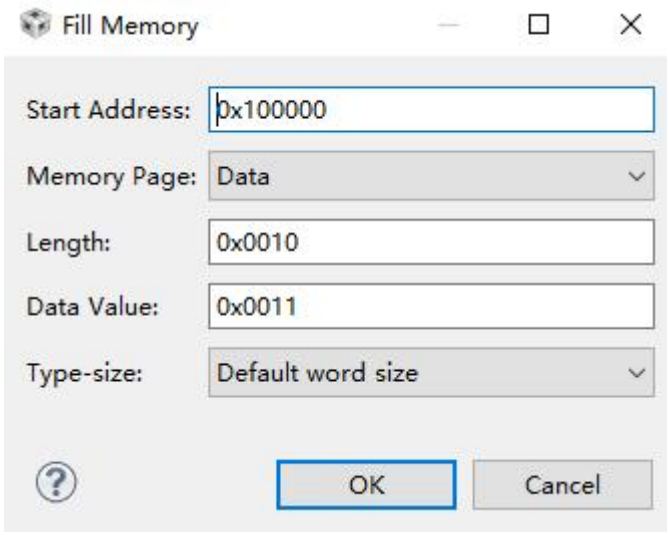
可见未初始化前，由于 SRAM 的掉电易失性，数据全为 0；

（3） InitXintf16Gpio() 函数进行 XINTF 总线初始化后，SRAM 内数据如下：

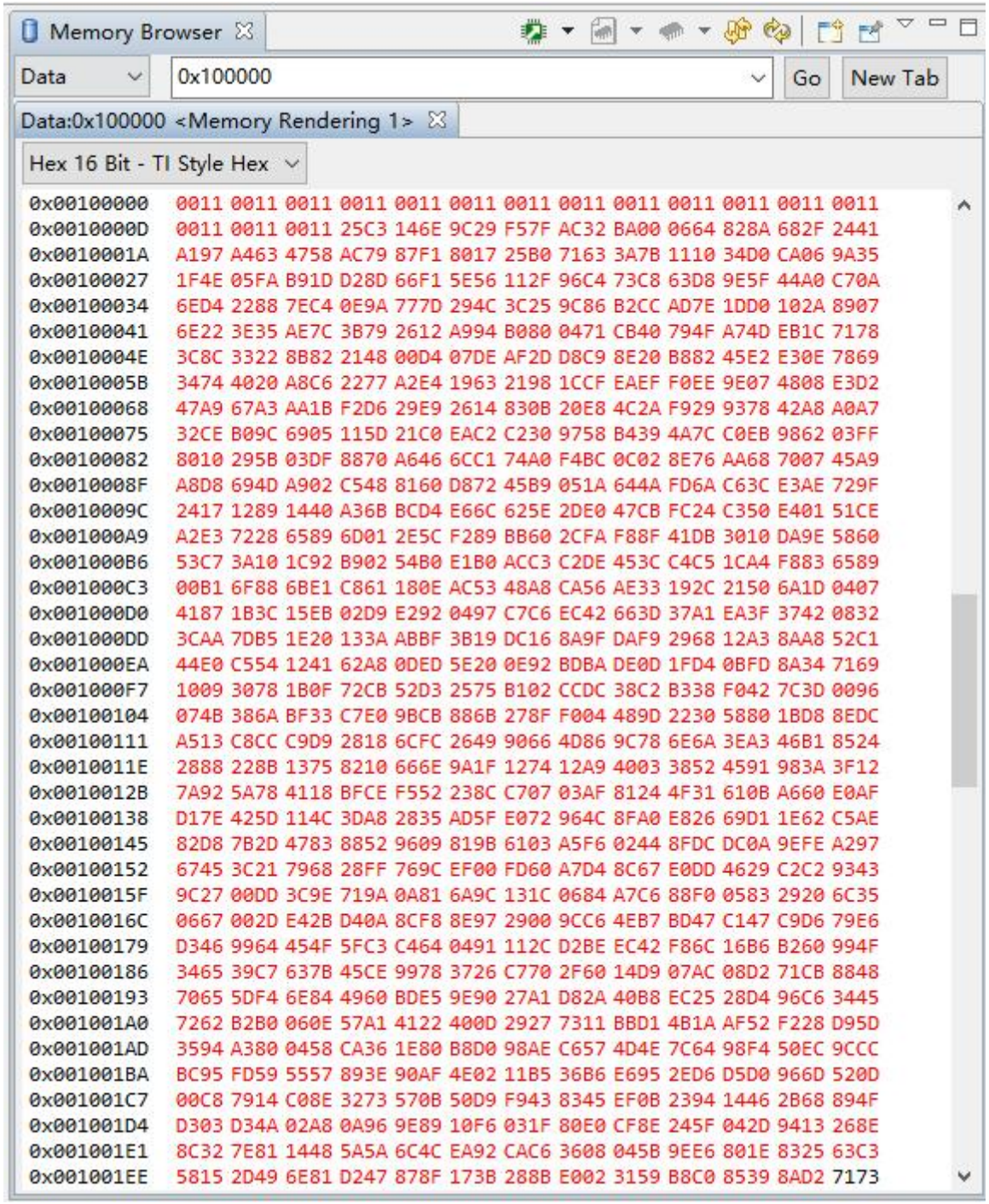


SRAM 内被填充随机数据；

（4） 使用 Fill Memory 功能向 SRAM 内写入数据：



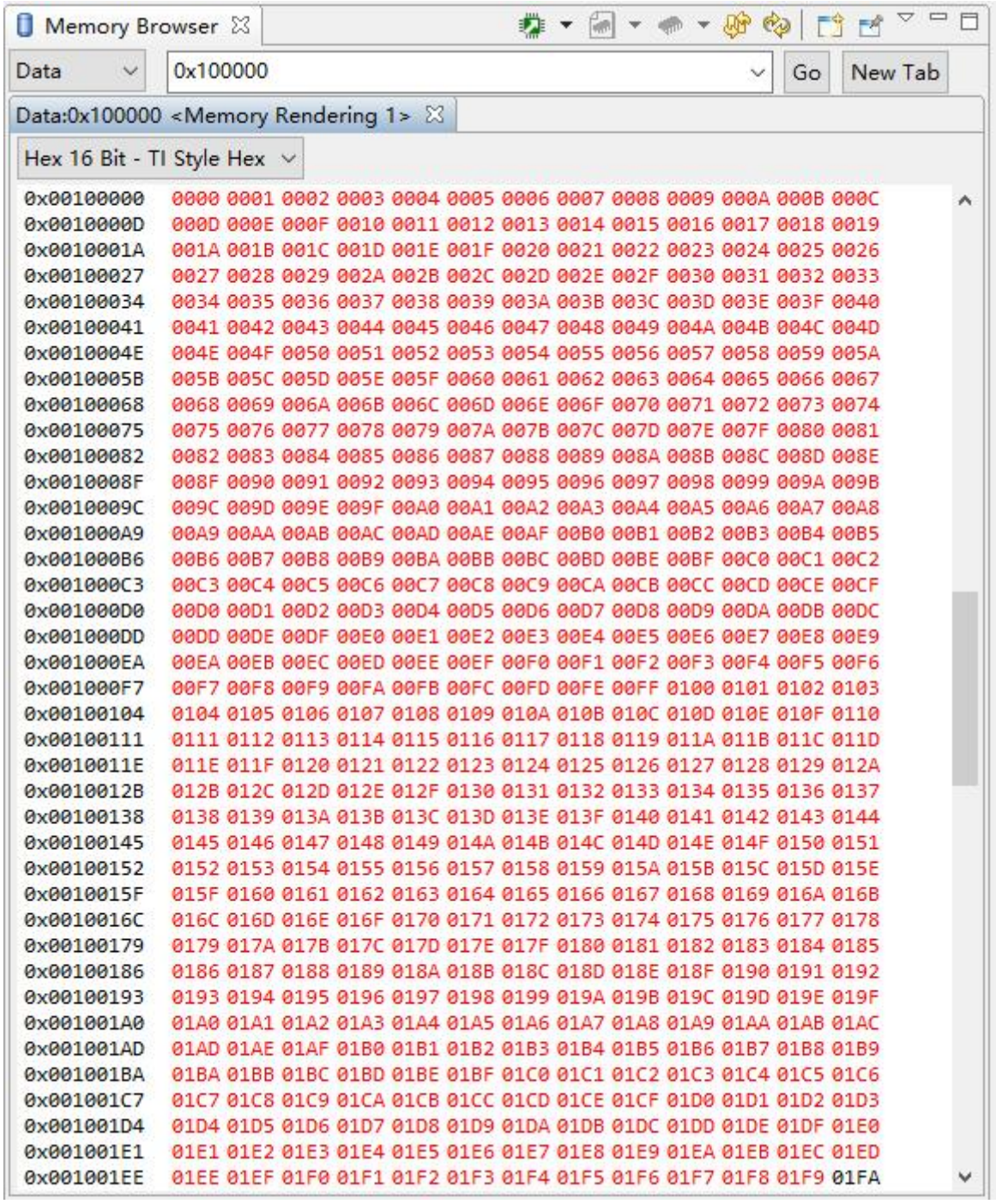
从 0x100000 开始写入，写入 16 个字节，写入值为 0x0011，结果如下：



（5） 使用程序向 SRAM 内写入数据：

```
// Step 4. Write Data to SRAM:
for(i=0;i<=0x0ffff;i++)
{
    *(int *)(SRAM_Base_Adress+i)=i;
};
```

上述代码的含义为：向 SRAM 内写入数据，从 SRAM 的基地址 SRAM_Base_Adress 开始，SRAM 内的每个值被赋值为当前地址的偏移量，结果如下：



四、问题与思考

1、实验程序运行之后，为什么是 65536 个单元的数值被赋值了？（0.5 分）

```
// Step 4. Write Data to SRAM:
for(i=0;i<=0x0ffff;i++)
{
    *(int *)(SRAM_Base_Adress+i)=i;
};
```

此段代码中，for 循环执行了 0xffff 次（即 65535 次），每次执行 for 循环，SRAM 内的对应地址的值（基地址 SRAM_Base_Adress 加上由 i 确定的偏移量）被赋值为 i，执行 65535 次后，65536 个单元的数值被赋值了。

哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验报告

课程名称：DSP 的原理与应用

实验名称：实验 2.2 浮点运算

专业-班级：电气 1 班 学号：220330124 姓名：舒晟超

实验日期：2024 年 5 月 7 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

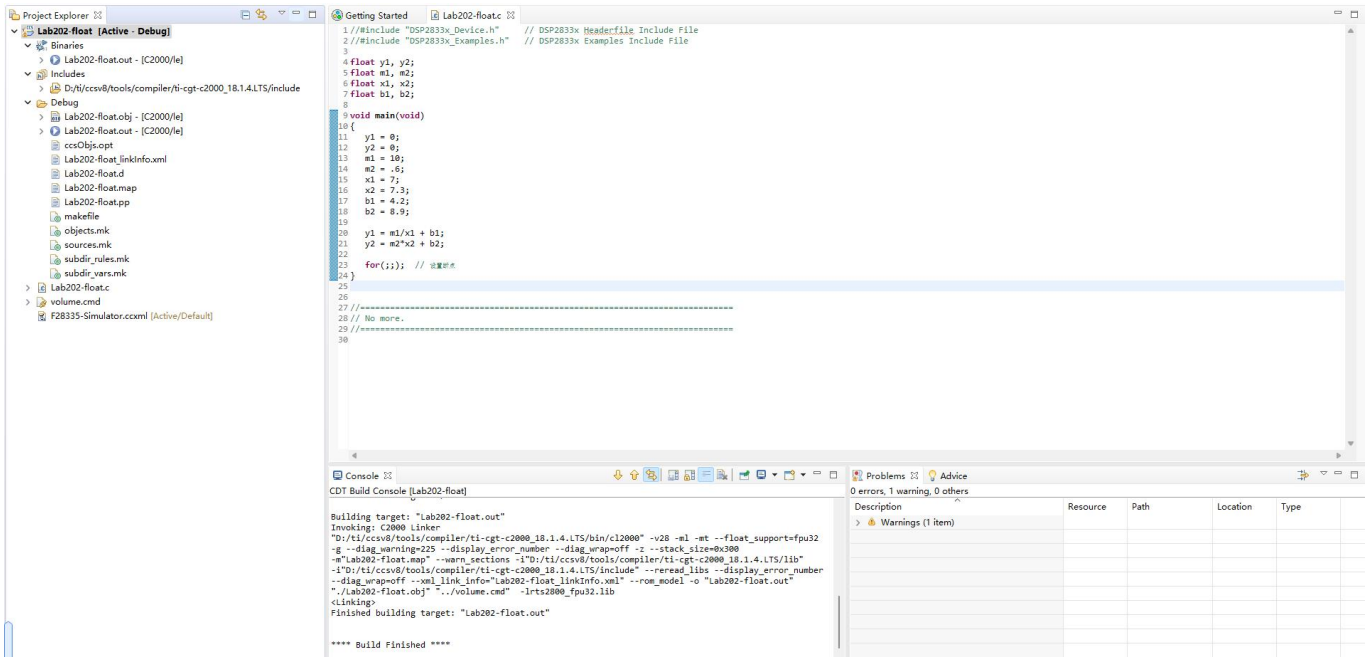
1．了解 F28335 简单的浮点运算。
2．熟悉浮点运算的编程。

二、实验过程

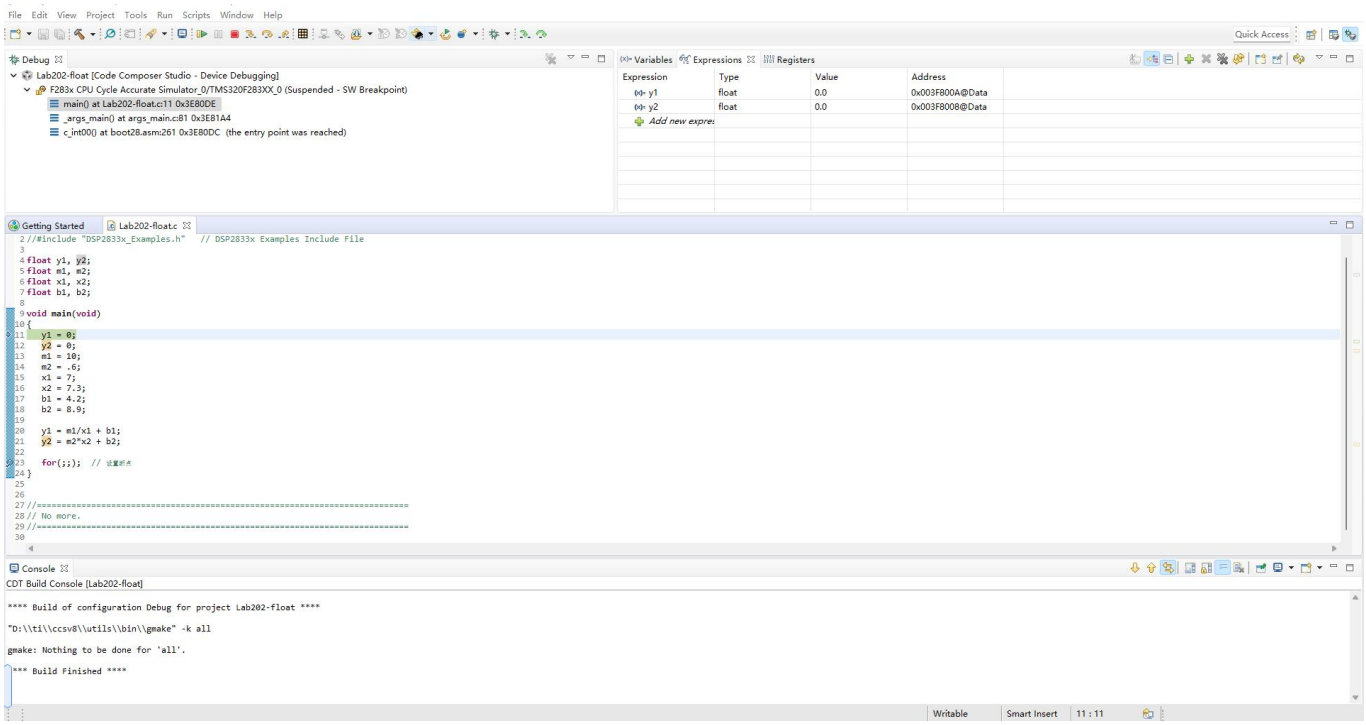
1．配置 CCS 工程并加入软件仿真文件。
2．阅读实验程序，编译并软件下载程序。
3．设置断点，将 y1，y2 添加进入观察窗中。
4．运行程序，观察 y1，y2 的变化。

三、实验结果

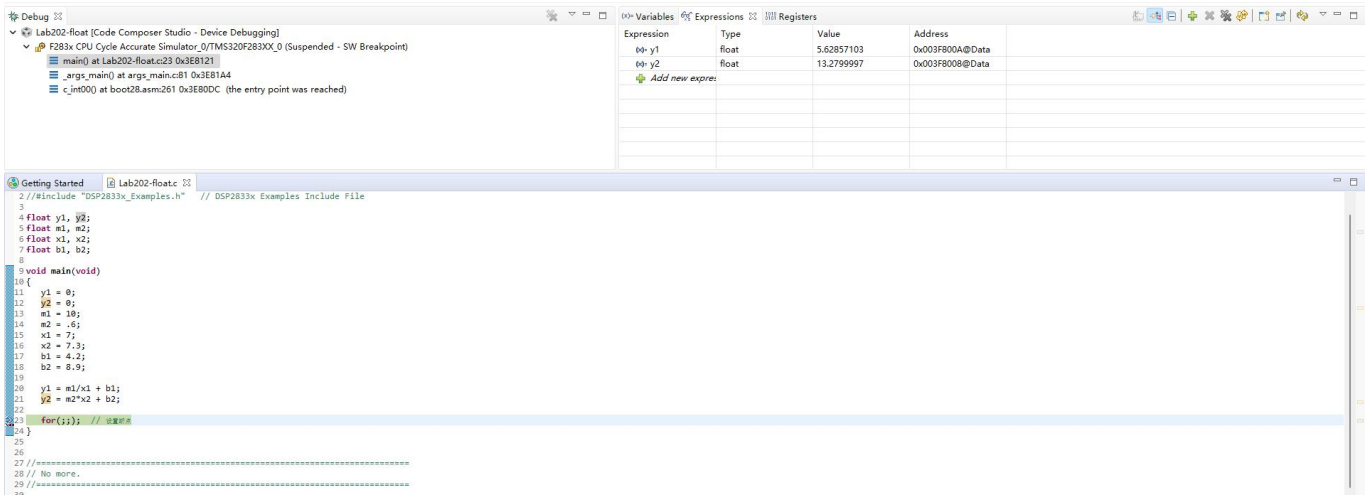
1. 编译程序，无报错。



2. 进行调试，添加断点和观察变量如图所示。



3. 进行浮点运算，结果如下图所示：



计算得到 y1 = 5.62857103；y2 = 13.279997.

四、问题与思考

1、做复杂的浮点运算。y=σ（wx+b），函数σ 满足 $\sigma(x)=\begin{cases} x, x \geq 0 \\ 0.01 * x, x < 0 \end{cases}$，

已知 w= [0.084674723,0.81282741,0.10292697,-0.047821075,-0.10049961,1.1399955,0.032412339;
-0.039466377,0.63574737,-0.099579401,0.045932423,0.056737024,-0.22110499,0.61971980;
-0.092473000,-0.32669711,0.42802703,-0.21985172,0.064885728,1.2945132,-0.38374531;
-0.094551362,0.66813517,-0.42289069,-0.70333308,0.10515465,0.012392324,0.17113267;
0.11922766,1.0923374,-0.52938485,-0.26163799,0.023534546,1.3951927,-0.62430763;
-0.34834740,0.93761641,0.99460375,-0.26750037,0.40269896,2.7415528,0.16500330;
0.069107793,0.37492210,-0.58213925,0.17143276,0.11286087,-0.28556311,-0.13933337;
-0.025005803,0.27282801,0.85436279,-0.079539515,0.048446644,2.9071076,-0.043221515;
-0.17358761,-0.71363419,-0.0023091321,0.35966095,0.22097406,-0.34137756,0.094514966];

b= [-0.43400329;6.0672159;-0.30059275;3.7516675;1.2958264;-2.5990894;3.3421099;
-0.83062923;0.30215138];

x= [0.1;0.2;0.3;0.4;0.5;0.6;7];求 y 的值。（1 分）

题目要求进行矩阵运算并将运算后的矩阵用 Sigma 函数进行处理。

w 为 9 行 7 列矩阵；x 为 7 行 1 列矩阵；b 为 9 行 1 列矩阵。

```
float w[9][7] =  {{0.084674723,0.81282741,0.10292697,-0.047821075,-0.10049961,1.1399955,0.032412339},
                  {-0.039466377,0.63574737,-0.099579401,0.045932423,0.056737024,-0.22110499,0.61971980},
                  {-0.092473000,-0.32669711,0.42802703,-0.21985172,0.064885728,1.2945132,-0.38374531},
                  {-0.094551362,0.66813517,-0.42289069,-0.70333308,0.10515465,0.012392324,0.17113267},
                  {0.11922766,1.0923374,-0.52938485,-0.26163799,0.023534546,1.3951927,-0.62430763},
                  {-0.34834740,0.93761641,0.99460375,-0.26750037,0.40269896,2.7415528,0.16500330},
                  {0.069107793,0.37492210,-0.58213925,0.17143276,0.11286087,-0.28556311,-0.13933337},
                  {-0.025005803,0.27282801,0.85436279,-0.079539515,0.048446644,2.9071076,-0.043221515},
                  {-0.17358761,-0.71363419,-0.0023091321,0.35966095,0.22097406,-0.34137756,0.094514966}};

float x[7][1] = {{0.1},
                 {0.2},
                 {0.3},
                 {0.4},
                 {0.5},
                 {0.6},
                 {7}};

float b[9][1] = {{-0.43400329},
                 {6.0672159},
                 {-0.30059275},
                 {3.7516675},
                 {1.2958264},
                 {-2.5990894},
                 {3.3421099},
                 {-0.83062923},
                 {0.30215138}};
```

定义 sigma 函数

```
float sigma_x(float x)
{
    if(x >= 0.0f)
    {
        return x;
    }
    else
    {
        return (0.01 * x);
    }
}
```

定义完成后在主程序中进行矩阵运算和函数处理，

```c
/**
 * main.c
 */
int main(void)
{
    float tmp[9][1] = {0};
    // Caculate bx+w
    int i,j,z;
    for(i = 0;i < 9;i++)
    {
        for(j = 0;j < 1;j++)
        {
            tmp[i][j]=0;
            for(z = 0; z < 7; z++)
            {
                tmp[i][j] += w[i][z] * x[z][j];
            }
        }
    }
    for(i = 0;i < 9;i++)
    {
        tmp[i][0] += b[i][0];
    }
    // Sigma x
    for(i = 0;i < 9;i++)
    {
        y[i][0] = sigma_x(tmp[i][0]);
    }

    for(;;)
    {
        ;
    }
}

```

主程序如下

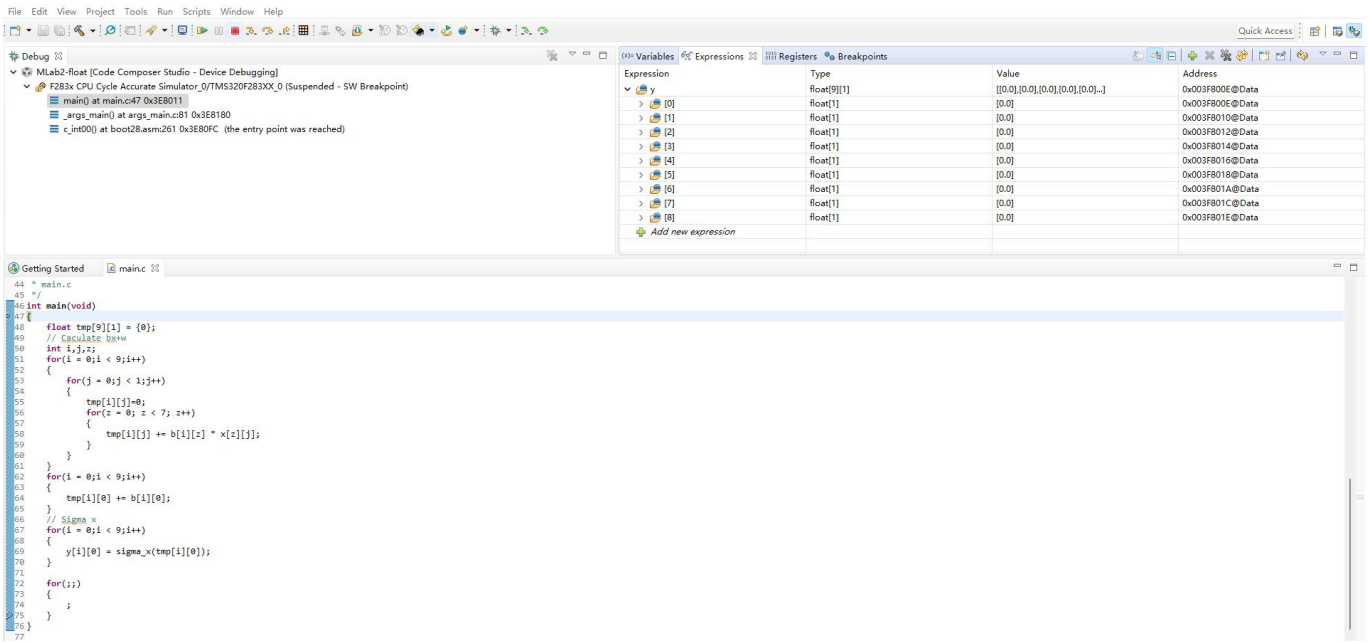
```
float w[9][7] =  {{0.084674723,0.81282741,0.10292697,-0.047821075,-0.10049961,1.1399955,0.032412339},
                  {-0.039466377,0.63574737,-0.099579401,0.045932423,0.056737024,-0.22110499,0.61971980},
                  {-0.092473000,-0.32669711,0.42802703,-0.21985172,0.064885728,1.2945132,-0.38374531},
                  {-0.094551362,0.66813517,-0.42289069,-0.70333308,0.10515465,0.012392324,0.17113267},
                  {0.11922766,1.0923374,-0.52938485,-0.26163799,0.023534546,1.3951927,-0.62430763},
                  {-0.34834740,0.93761641,0.99460375,-0.26750037,0.40269896,2.7415528,0.16500330},
                  {0.069107793,0.37492210,-0.58213925,0.17143276,0.11286087,-0.28556311,-0.13933337},
                  {-0.025005803,0.27282801,0.85436279,-0.079539515,0.048446644,2.9071076,-0.043221515},
                  {-0.17358761,-0.71363419,-0.0023091321,0.35966095,0.22097406,-0.34137756,0.094514966}};

float x[7][1] = {{0.1},
                 {0.2},
```

```
                {0.3},
                {0.4},
                {0.5},
                {0.6},
                {7}};

float b[9][1] = {{-0.43400329},
                {6.0672159},
                {-0.30059275},
                {3.7516675},
                {1.2958264},
                {-2.5990894},
                {3.3421099},
                {-0.83062923},
                {0.30215138}};

float y[9][1];

float sigma_x(float x)
{
    if(x >= 0.0f)
    {
        return x;
    }
    else
    {
        return (0.01 * x);
    }
}

/**
 * main.c
 */
int main(void)
{
    float tmp[9][1] = {0};
    // Caculate bx+w
    int i,j,z;
    for(i = 0;i < 9;i++)
    {
        for(j = 0;j < 1;j++)
        {
            tmp[i][j]=0;
            for(z = 0; z < 7; z++)
            {
                tmp[i][j] += w[i][z] * x[z][j];
            }
        }
```

```
    }
    for(i = 0;i < 9;i++)
    {
        tmp[i][0] += b[i][0];
    }
    // Sigma x
    for(i = 0;i < 9;i++)
    {
        y[i][0] = sigma_x(tmp[i][0]);
    }

    for(;;)
    {
        ;
    }
}
```

编译通过无报错。

进行调试，加入断点和监视变量如图所示：



运行代码至断点处。得到计算结果如图所示。

Expression	Type	Value	Address
y	float[9][1]	[[0.609413207],[10.4126625],[-0.02211778...	0x003F800E@Data
[0]	float[1]	[0.609413207]	0x003F800E@Data
[1]	float[1]	[10.4126625]	0x003F8010@Data
[2]	float[1]	[-0.0221177842]	0x003F8012@Data
[3]	float[1]	[4.72558022]	0x003F8014@Data
[4]	float[1]	[-0.022585243]	0x003F8016@Data
[5]	float[1]	[0.746284485]	0x003F8018@Data
[6]	float[1]	[2.22769547]	0x003F801A@Data
[7]	float[1]	[0.911866307]	0x003F801C@Data
[8]	float[1]	[0.852502644]	0x003F801E@Data
Add new expression			