SigmaDSP 启动与控制

Content

- 搭建 SigmaDSP 开发环境 Sigmastudio
- 用 MCU 启动 SigmaDSP
- 用 MCU 控制 SigmaDSP

一． SigmaDSP 开发环境 SigmaStudio 搭建及简单音频链路设计

1、PC 端安装 SigmaStudio

从官网下载最新版的 SigmaStudio：

https://form.analog.com/Form_Pages/sigmastudio/SSDownLoad.aspx

找到合适的 PC 版本，如 32 位 PC 系统下载 X86 版的 SigmaStudio，64 位的 PC 系统下载 X64 版的 SigmaStudio。或向代理商寻要最新版本的 SigmaStudio。

安装 SigmaStudio 时，有可能需要你安装 Microsoft.NET Framework Ver3.5，如果你的机器之前没有安装过此类软件包，请到官网下载安装。若是 WIN7 或以上 PC 系统，请确保“我的文档”在 C 盘目录中，否则在安装过程中会提示出错。

2、连接 USBi 仿真下载器

将 USBi 仿真器通过 MiniUSB 连接至 PC，SigmaStudio 已集成 USBi 的驱动程序，插上 PC 后在设备管理器中就能看到 Analog Devices USBi 设备，如图 9 所示：

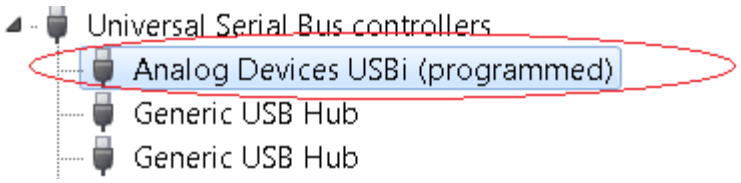


图 9 Analog DevicesUSBi 设备

3、将 USBi 仿真器按如图 3 所示连接到开发板

4、建立 SigmaStudio 工程文件

打开 SigmaStudio，选择菜单“File——New Project”新建一个工程，在左侧的“Tree Tool Box”工具栏中将 USBi 及 ADAU176x 拖做中间空白处，如图 10 所示：

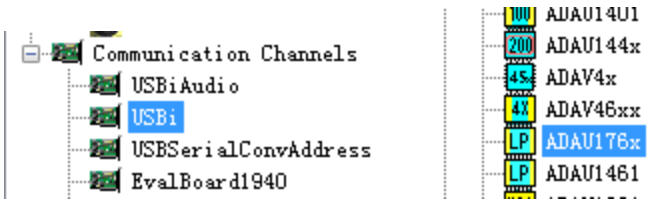


图 10 USBi 及 ADAU1761 插件

在空白处用鼠标按图 11 所示将线连接起来

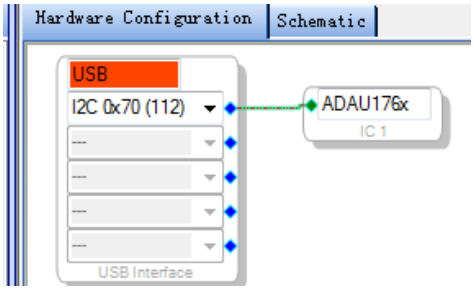


图 11 USBi 连接 DSP

点击“Schematic”选项卡，进入 ADAU1761 音频链路设计。同样在左侧的“Tree Tool Box”工具栏分别找到 Input、Output、Mute、Single Volume、Medium Size EQ 拖至中间空白处，选中 Single Volume 右键 Add Algorithm--IC1—GAIN(NO SLEW)，选中 Medium Size EQ 右键 Add Algorithm—1.2 channel double precision—9，然后按照图 12 将链路连接起来，这样一个简单的 DSP 音频链路就设计好了。详细的 SigmaStudio 工程设计资料请参考“SigmaStudioHelp_3.0(中文).pdf”或按 F1 获取帮助，也可参考“SigmaStudio Basic uC Integration Tutorial.pdf”、“SigmaStudio 实例.pdf”，或联系代理商获取帮助。

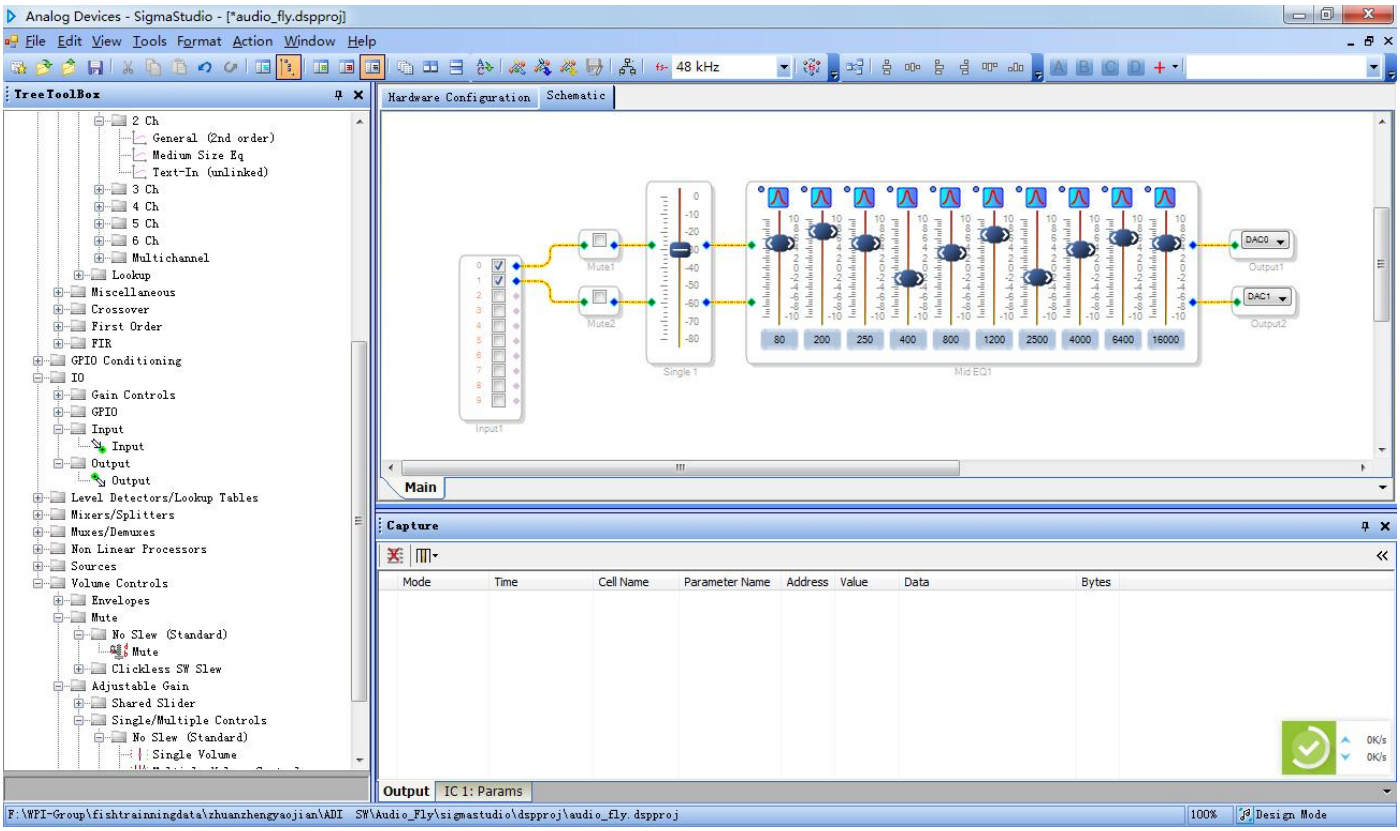


图 12 DSP 音频链路

5、编译下载工程

点击工具栏中 Link Compile Download 按钮（如图 13 所示），可将设计好的工程文件编译，并通过 USBi 将编译过的配置参数载入 DSP，DSP 的参数及时生效。

图 13 工程文件编译和下载

6、导出配置文件

点击工具栏中的 Export System Files 按钮（如图 14 所示），可将设计好的工程文件导出为 8 个 MCU 配置文件。

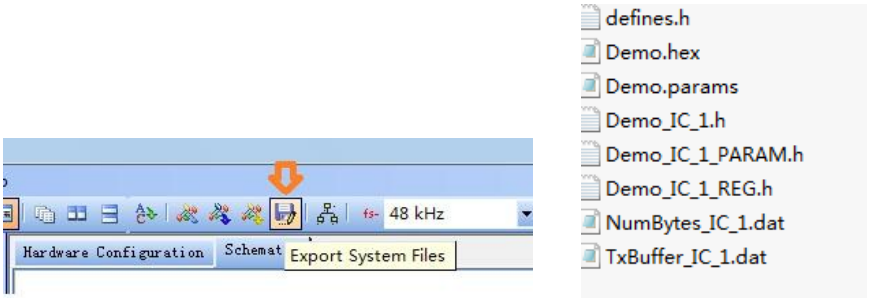


图 14 工程配置文件导出

二. MCU 启动 SigmaDSP

1、在设计好 SigmaStudio 的工程后，编译后点击 export System files 按钮，就可以生成导出工程文件，导出的配置文件有 8 个。如下图所示

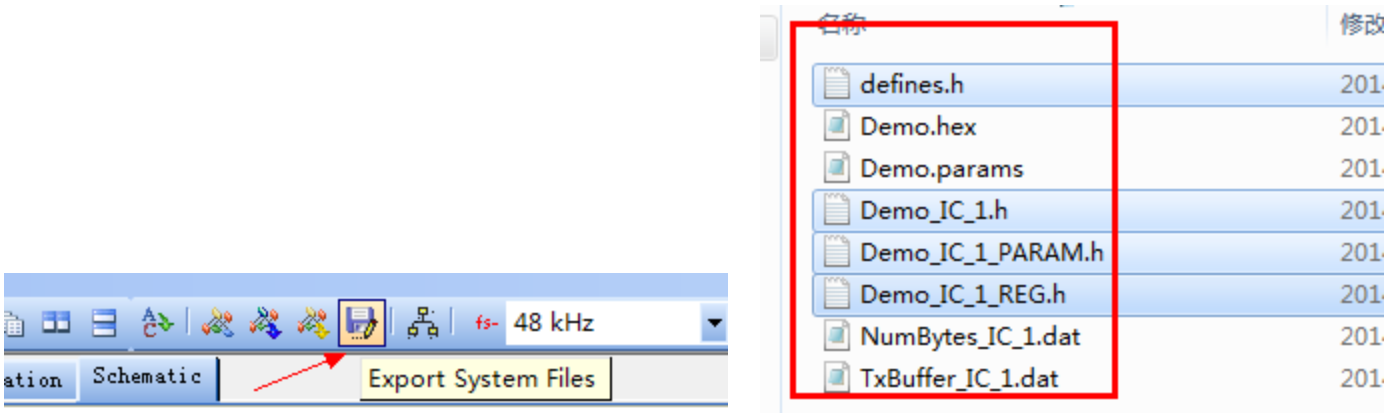


图 18 SigmaStudio 导出配置文件

2、在图 18 所生成的配置文件中，我们一般的 MCU 工程用到的有四个（SigmaStudioFW.h 是用户的 I2C 接口文件，需要用户根据 MCU 平台自己配置 I2C）文件：“define.h”，“Demo_IC_1.h”，“Demo_IC_1_PARAM.h”，“Demo_IC_1_REG.h”。在 MCU 工程下建好 DSP 的子工程如图 3 所示：

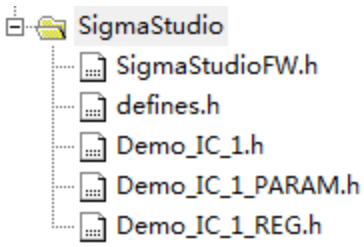


图 19 MCU 工程中的 DSP 子工程

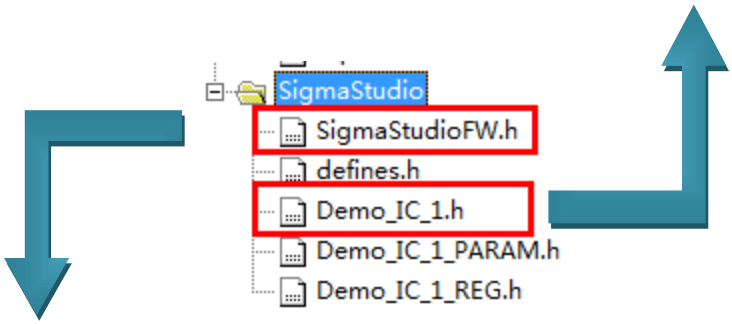
3、DSP 需要配置相关的参数才能启动，而这些所有与 DSP 启动相关的参数由“Demo_IC_1.h”中的 default_download_IC_1() 函数完成，只要在 MCU 主程序中执行一次 default_download_IC_1() 函数即可启动

DSP。如图 20 所示

```
 * Default Download
 */
#define DEFAULT_DOWNLOAD_SIZE_IC_1 84

void default_download_IC_1() {
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SAMPLE_RATE_SETTING_IC_1_ADDR, REG_SAMPLE_RATE_SETTING_IC_1_BY
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_DSP_RUN_REGISTER_IC_1_ADDR, REG_DSP_RUN_REGISTER_IC_1_BYTE, R1
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_CLKCTRLREGISTER_IC_1_ADDR, REG_CLKCTRLREGISTER_IC_1_BYTE, R2_C
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_PLLCRLREGISTER_IC_1_ADDR, REG_PLLCRLREGISTER_IC_1_BYTE, R3_PLL
  SIGMA_WRITE_DELAY( DEVICE_ADDR_IC_1, R4_DELAY_IC_1_SIZE, R4_DELAY_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SERIAL_PORT_CONTROL_0_IC_1_ADDR , R5_SERIAL_PORT_CONTROL_REGIS
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_ALC_CONTROL_0_IC_1_ADDR , R6_ALC_CONTROL_REGISTERS_IC_1_SIZE,
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_MICCTRLREGISTER_IC_1_ADDR, REG_MICCTRLREGISTER_IC_1_BYTE, R7_M
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_RECORD_PWR_MANAGEMENT_IC_1_ADDR , R8_RECORD_INPUT_SIGNAL_PATH_
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_ADC_CONTROL_0_IC_1_ADDR , R9_ADC_CONTROL_REGISTERS_IC_1_SIZE,
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_PLAYBACK_MIXER_LEFT_CONTROL_0_IC_1_ADDR , R10_PLAYBACK_OUTPUT_
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_CONVERTER_CTRL_0_IC_1_ADDR , R11_CONVERTER_CONTROL_REGISTERS_I
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_DAC_CONTROL_0_IC_1_ADDR , R12_DAC_CONTROL_REGISTERS_IC_1_SIZE,
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SERIAL_PORT_PAD_CONTROL_0_IC_1_ADDR , R13_SERIAL_PORT_PAD_CONT
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_COMM_PORT_PAD_CTRL_0_IC_1_ADDR , R14_COMMUNICATION_PORT_PAD_CO
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_JACKREGISTER_IC_1_ADDR, REG_JACKREGISTER_IC_1_BYTE, R15_JACKRE
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_DSP_ENABLE_REGISTER_IC_1_ADDR, REG_DSP_ENABLE_REGISTER_IC_1_BY
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_CRC_IDEAL_1_IC_1_ADDR , R17_CRC_REGISTERS_IC_1_SIZE, R17_CRC_R
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_GPIO_0_CONTROL_IC_1_ADDR , R18_GPIO_REGISTERS_IC_1_SIZE, R18_G
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_NON_MODULO_RAM_1_IC_1_ADDR , R19_NON_MODULO_REGISTERS_IC_1_SIZ
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_WATCHDOG_ENABLE_IC_1_ADDR , R20_WATCHDOG_REGISTERS_IC_1_SIZE,
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SAMPLE_RATE_SETTING_IC_1_ADDR, REG_SAMPLE_RATE_SETTING_IC_1_BY
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_ROUTING_MATRIX_INPUTS_IC_1_ADDR, REG_ROUTING_MATRIX_INPUTS_IC_
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_ROUTING_MATRIX_OUTPUTS_IC_1_ADDR, REG_ROUTING_MATRIX_OUTPUTS_I
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SERIAL_DATAGPIO_PIN_CONFIG_IC_1_ADDR, REG_SERIAL_DATAGPIO_PIN_
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_DSP_SLEW_MODES_IC_1_ADDR, REG_DSP_SLEW_MODES_IC_1_BYTE, R25_DS
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SERIAL_PORT_SAMPLE_RATE_SETTING_IC_1_ADDR, REG_SERIAL_PORT_SAM
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_CLOCK_ENABLE_REG_0_IC_1_ADDR , R27_CLOCK_ENABLE_REGISTERS_IC_1
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, PROGRAM_ADDR_IC_1, PROGRAM_SIZE_IC_1, Program_Data_IC_1 );
```

DSP 启动需要通过此函数载入参数




```
void SIGMA_WRITE_REGISTER_BLOCK(int devAddress, int address, int length, ADI_REG_TYPE *pData )
{
  int zz = 0;
  uint32_t i;
  for ( i = 0; i < length+3; i++ )  /* clear buffer */
  {
  I2CMasterBuffer[i] = 0;
  I2CSlaveBuffer[i] = 0;
  }
  I2CWriteLength = length+3;                  //length = byte of slave device register need to
  I2CReadLength = 0;                          //no need to reda, readlength=0;
  I2CMasterBuffer[0] = devAddress;
  I2CMasterBuffer[1] = (address & 0xFF00)>>8;
  I2CMasterBuffer[2] = address & 0x00FF;
  for(zz=0;zz<length;zz++)
  {
  I2CMasterBuffer[zz + Address_Length + 1] =   pData[zz];
  }
  I2CEngine();
}
```

通过 MCU 的 I2C 发送数据

图 20 SigmaStudio 子工程文件解析

4、default_download_IC_1( ) 中的子函数 SIGMA_WRITE_REGISTER_BLOCK( ) 由 I2C/SPI 接口完成数据传送，需要传送的参数有：int devAddress，int address，int length，ADI_REG_TYPE *pData，四个，分别代表的是：I2C/SPI 设备地址，寄存器地址，数据长度，数据。这四个数据在 Sigmastudio 导出配置的时候已经定义

在 “Demo_IC_1.h"，"Demo_IC_1_REG.h" 中。

5、所以 MCU 启动 DSP 时序是：①上电，②延时等待约 100ms，③执行一次 default_download_IC_1( ) 函数。

三. MCU 控制 SigmaDSP

1、控制 SigmaDSP 需要用到 SIGMA_WRITE_REGISTER_BLOCK( ) 和 SIGMA_SAFELOAD_WRITE_REGISTER( ) 两个接口函数。SIGMA_WRITE_REGISTER_BLOCK( ) 函数与启动 DSP 的接口函数是一样的，SIGMA_SAFELOAD_WRITE_REGISTER( ) 接口需要先写到 DSP 里面的 Buffer，然后再触发，参考示意图 21：

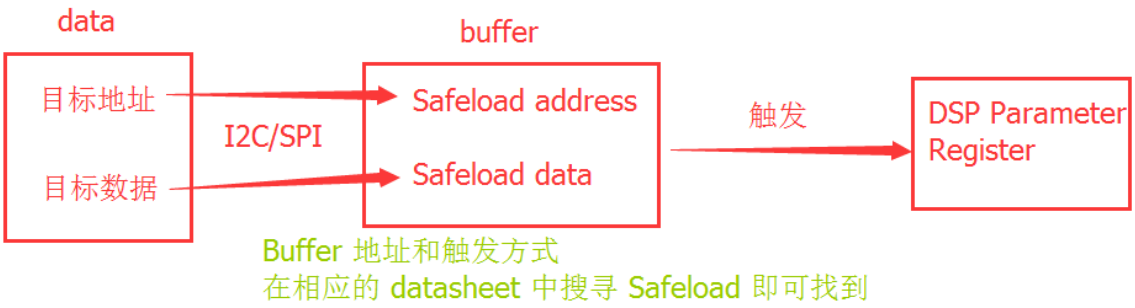
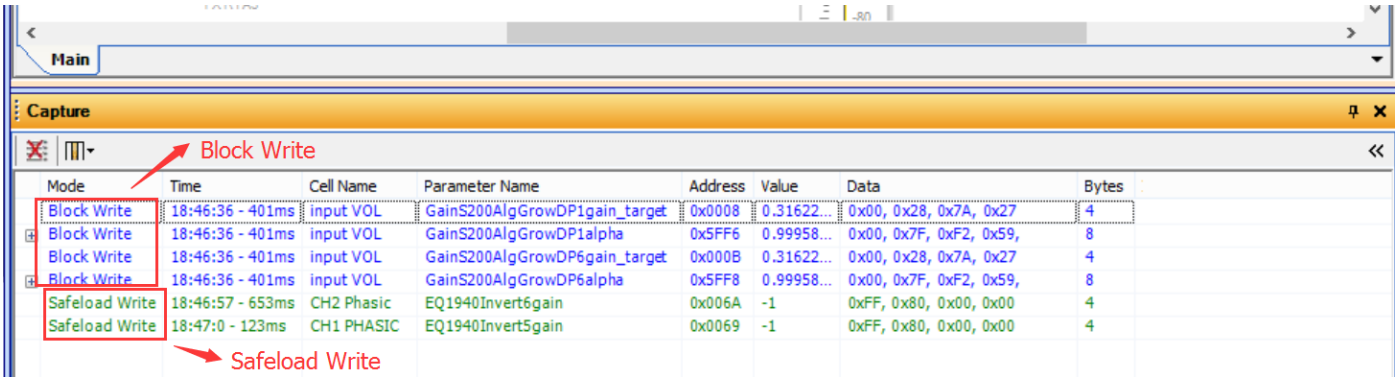


图 21 Safeload Write 示意图

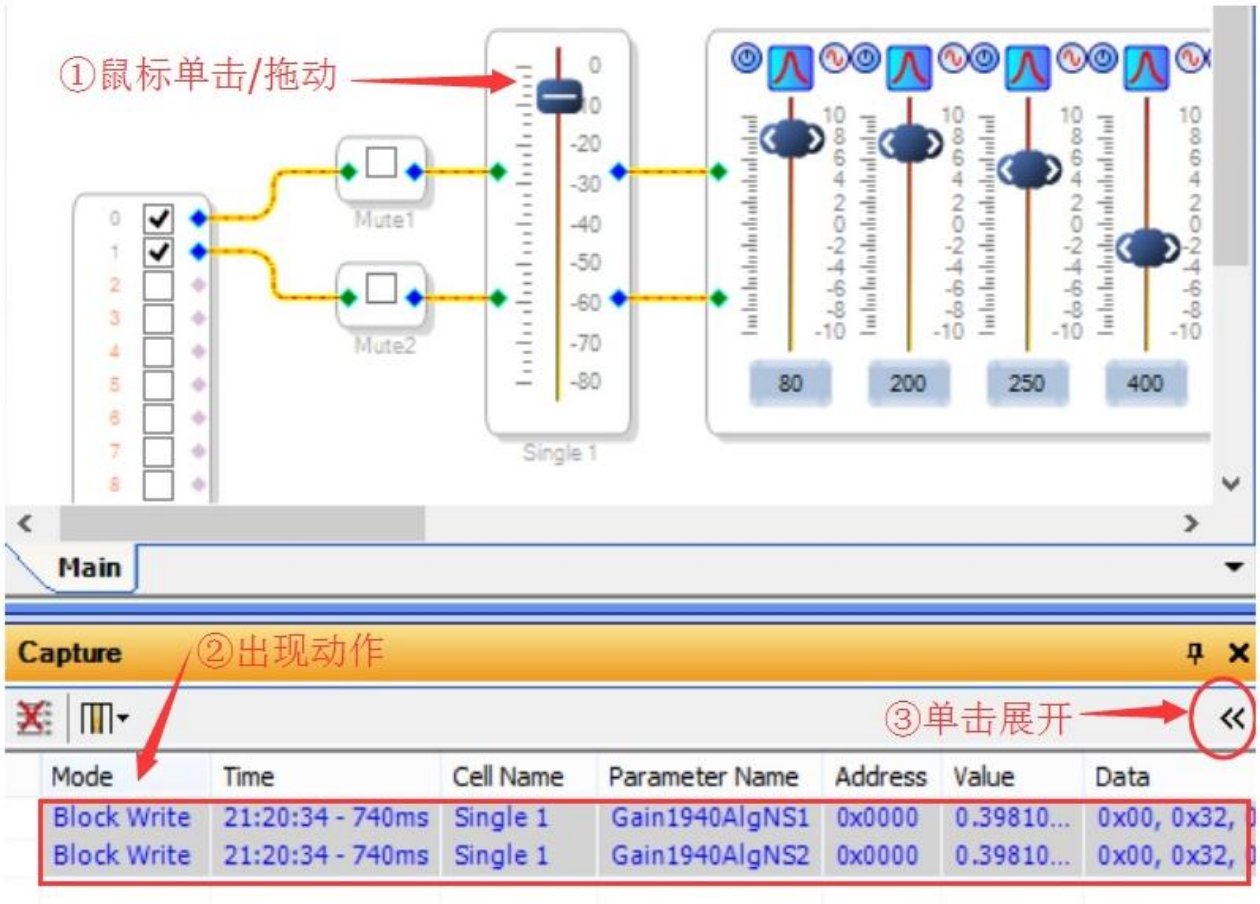
2、编译好 SigmaStudio 工程文件后，用鼠标在 SigmaStudio 界面调节一些参数，然后在下面的 Capture window 中可以看到会有如下图 22 的信息，每个鼠标动作都会在 Capture window 出现相应的一行/几行动作。Mode 列中显示的即是控制方式，有 Block Write 和 Safeload Write。

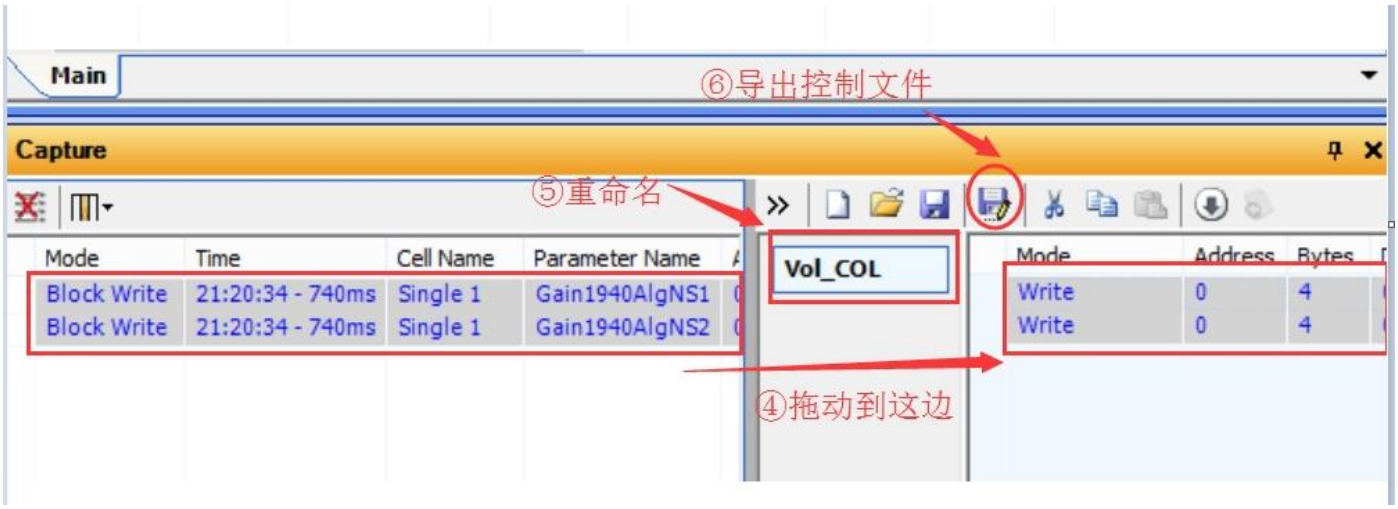


Mode	Time	Cell Name	Parameter Name	Address	Value	Data	Bytes
Block Write	18:46:36 - 401ms	input VOL	GainS200AlgGrowDP1gain_target	0x0008	0.31622...	0x00, 0x28, 0x7A, 0x27	4
Block Write	18:46:36 - 401ms	input VOL	GainS200AlgGrowDP1alpha	0x5FF6	0.99958...	0x00, 0x7F, 0xF2, 0x59,	8
Block Write	18:46:36 - 401ms	input VOL	GainS200AlgGrowDP6gain_target	0x000B	0.31622...	0x00, 0x28, 0x7A, 0x27	4
Block Write	18:46:36 - 401ms	input VOL	GainS200AlgGrowDP6alpha	0x5FF8	0.99958...	0x00, 0x7F, 0xF2, 0x59,	8
Safeload Write	18:46:57 - 653ms	CH2 Phasic	EQ1940Invert6gain	0x006A	-1	0xFF, 0x80, 0x00, 0x00	4
Safeload Write	18:47:0 - 123ms	CH1 PHASIC	EQ1940Invert5gain	0x0069	-1	0xFF, 0x80, 0x00, 0x00	4

图 22 Capture window 动作信息

3、上图中蓝色的控制方式 Block Write 使用 SIGMA_WRITE_REGISTER_BLOCK( ) 函数，具体导出控制文件的流程如下所示：





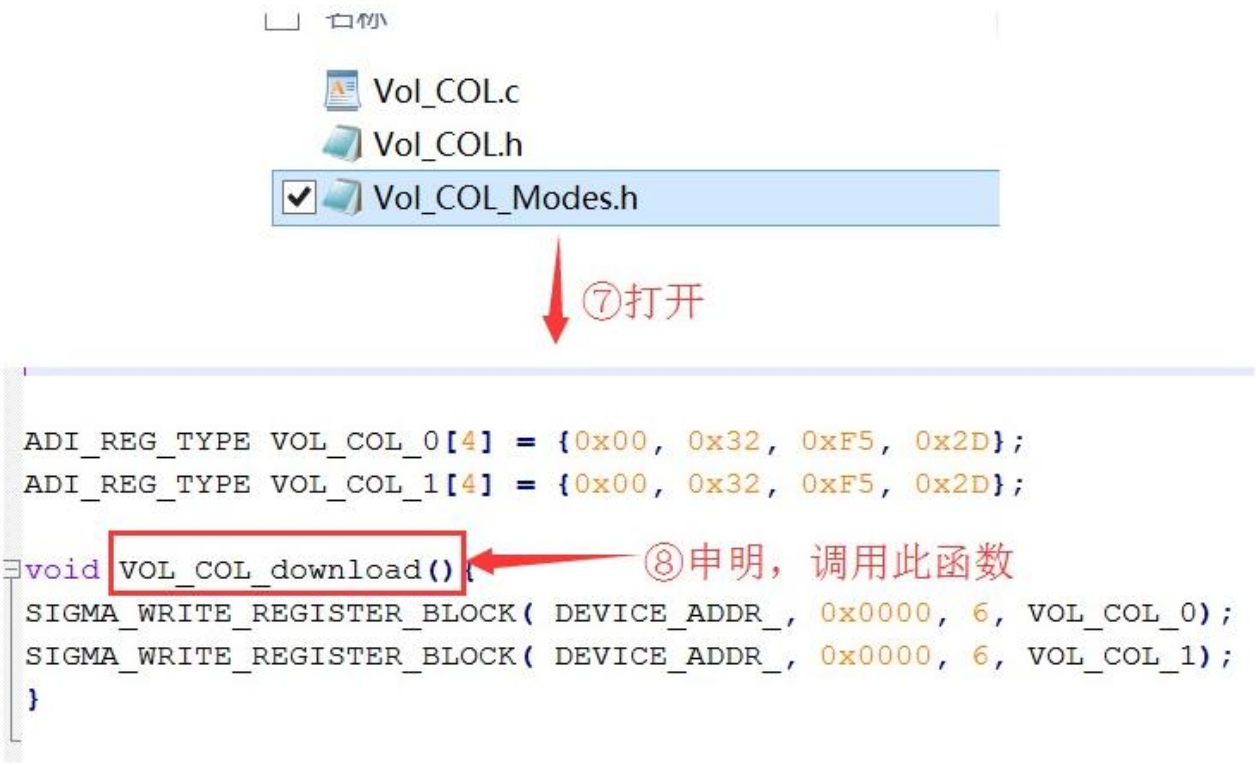
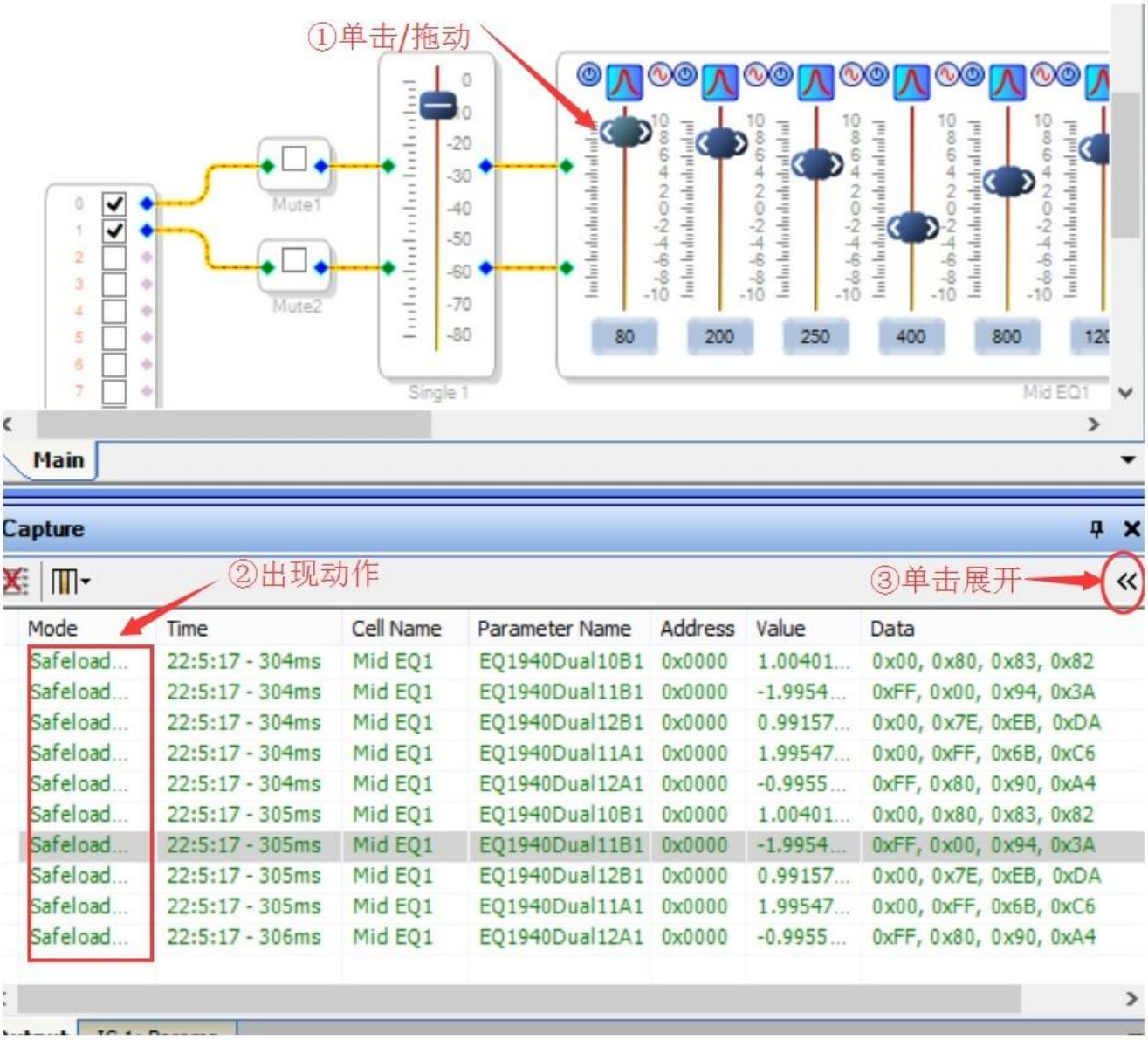
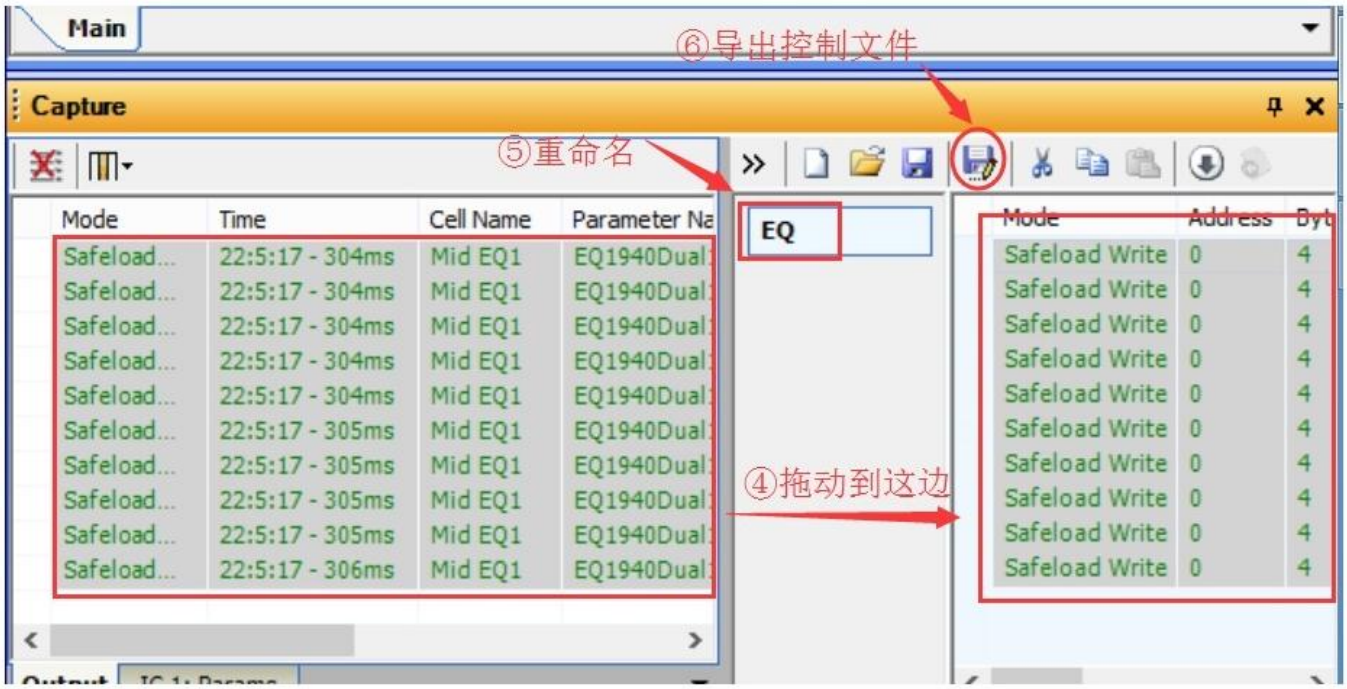


图 23 Block Write 导出动作

4、Safeload Write 控制方式用 SIGMA_SAFELOAD_WRITE_REGISTER( )函数实现，其导出动作方式与 Block Write 方式类似，如图 24 所示：


①单击/拖动
Mute1
Mute2
Single 1
Mid EQ1
80
200
250
400
800
Main
Capture
②出现动作
③单击展开
Mode
Time
Cell Name
Parameter Name
Address
Value
Data
Safeload... 22:5:17 - 304ms Mid EQ1 EQ1940Dual10B1 0x0000 1.00401... 0x00, 0x80, 0x83, 0x82
Safeload... 22:5:17 - 304ms Mid EQ1 EQ1940Dual11B1 0x0000 -1.9954... 0xFF, 0x00, 0x94, 0x3A
Safeload... 22:5:17 - 304ms Mid EQ1 EQ1940Dual12B1 0x0000 0.99157... 0x00, 0x7E, 0xEB, 0xDA
Safeload... 22:5:17 - 304ms Mid EQ1 EQ1940Dual11A1 0x0000 1.99547... 0x00, 0xFF, 0x6B, 0xC6
Safeload... 22:5:17 - 304ms Mid EQ1 EQ1940Dual12A1 0x0000 -0.9955... 0xFF, 0x80, 0x90, 0xA4
Safeload... 22:5:17 - 305ms Mid EQ1 EQ1940Dual10B1 0x0000 1.00401... 0x00, 0x80, 0x83, 0x82
Safeload... 22:5:17 - 305ms Mid EQ1 EQ1940Dual11B1 0x0000 -1.9954... 0xFF, 0x00, 0x94, 0x3A
Safeload... 22:5:17 - 305ms Mid EQ1 EQ1940Dual12B1 0x0000 0.99157... 0x00, 0x7E, 0xEB, 0xDA
Safeload... 22:5:17 - 305ms Mid EQ1 EQ1940Dual11A1 0x0000 1.99547... 0x00, 0xFF, 0x6B, 0xC6
Safeload... 22:5:17 - 306ms Mid EQ1 EQ1940Dual12A1 0x0000 -0.9955... 0xFF, 0x80, 0x90, 0xA4


Main
⑥导出控制文件
Capture
⑤重命名
EQ
④拖动到这边
Mode
Time
Cell Name
Parameter Na
Safeload...
22:5:17 - 304ms
22:5:17 - 305ms
22:5:17 - 306ms
Mid EQ1
EQ1940Dual
Mode
Address
Byt
Safeload Write
0
4


EQ_Modes.h
EQ.h
EQ.c
⑦打开

```
ADI_REG_TYPE EQ_0[4] = {0x00, 0x80, 0x83, 0x82};
ADI_REG_TYPE EQ_1[4] = {0xFF, 0x00, 0x94, 0x3A};
ADI_REG_TYPE EQ_2[4] = {0x00, 0x7E, 0xEB, 0xDA};
ADI_REG_TYPE EQ_3[4] = {0x00, 0xFF, 0x6B, 0xC6};
ADI_REG_TYPE EQ_4[4] = {0xFF, 0x80, 0x90, 0xA4};
ADI_REG_TYPE EQ_5[4] = {0x00, 0x80, 0x83, 0x82};
ADI_REG_TYPE EQ_6[4] = {0xFF, 0x00, 0x94, 0x3A};
ADI_REG_TYPE EQ_7[4] = {0x00, 0x7E, 0xEB, 0xDA};
ADI_REG_TYPE EQ_8[4] = {0x00, 0xFF, 0x6B, 0xC6};
ADI_REG_TYPE EQ_9[4] = {0xFF, 0x80, 0x90, 0xA4};

void EQ_download(){
SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_, 0x0000, 6, EQ_0);
SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_, 0x0000, 6, EQ_1);
SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_, 0x0000, 6, EQ_2);
SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_, 0x0000, 6, EQ_3);
SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_, 0x0000, 6, EQ_4);
SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_, 0x0000, 6, EQ_5);
SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_, 0x0000, 6, EQ_6);
SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_, 0x0000, 6, EQ_7);
SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_, 0x0000, 6, EQ_8);
SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_, 0x0000, 6, EQ_9);
}
```

⑧申明，调用此函数

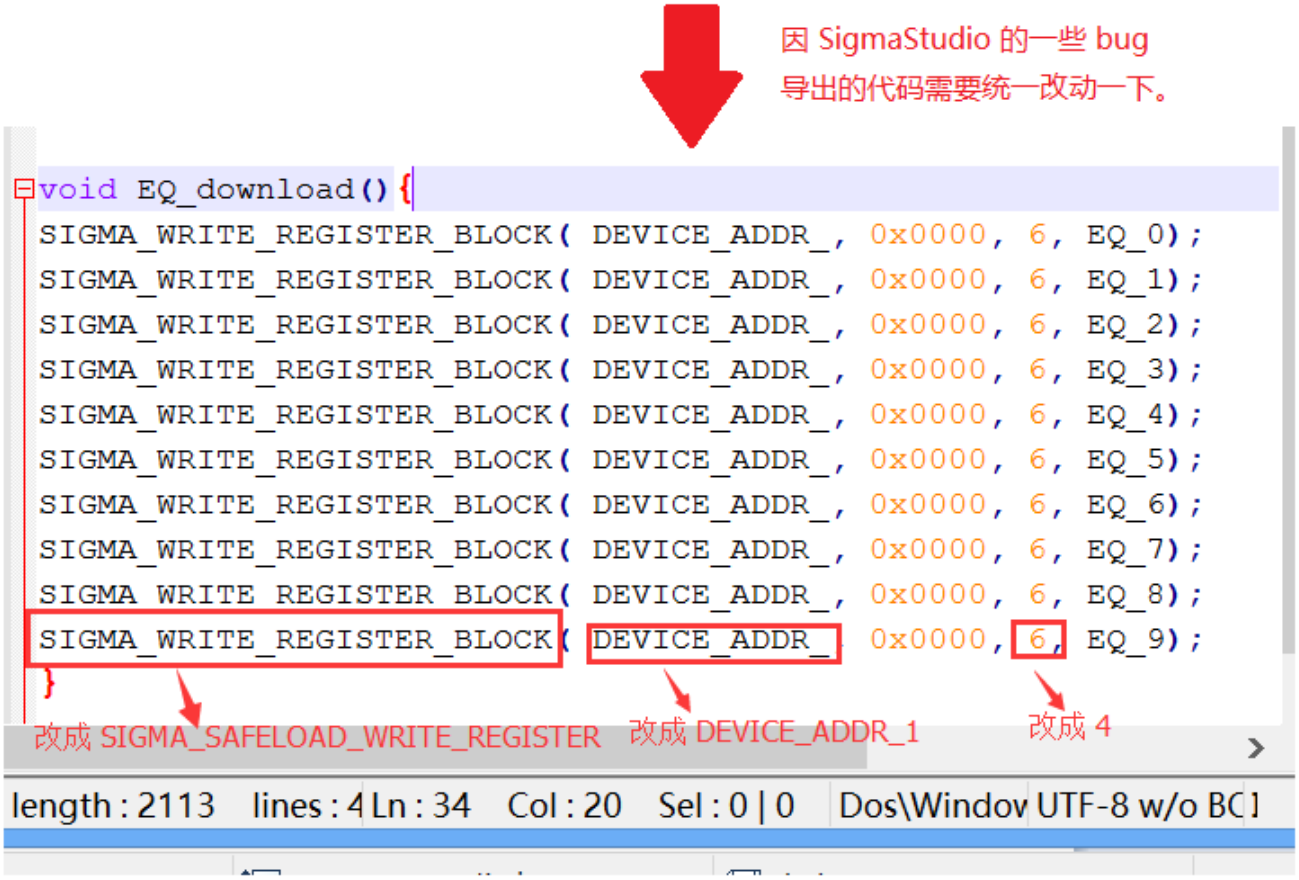


图 24 Safeload write 导出动作

5、Block Write 和 Safeload Write 动作可以混在一起导出，并且顺序可以任意组合，方法与前面一样。在 SigmaStudio 上面用鼠标按顺序控制，在 Capture window 会按顺序出现动作信息，把这些动作导出来，用 MCU 调用一遍导出来的函数就可以实现鼠标操作的效果。

完毕

6、为什么一定要用 Safeload？因为 Safeload 机制可以避免更新 DSP 内核参数时产生噪声。

Good Luck！