## 实验1 基于C的MDK工程建立及跟踪和调试过程

### 1.1 实验目的

（1） 了解 MDK开发环境建立过程

（2） 掌握 µVision IDE下创建C语言工程（Project）的基本步骤

（3） 了解 Project的组成和Project管理的作用

（4） 了解基于软件仿真的系统开发流程

（5） 养成查阅联机帮助的习惯，并掌握联机帮助查询技巧

（6） 养成调试习惯，并掌握基础代码调试技巧

### 1.2 实验内容

#### 1.2.1 MDK基于C程序Project的建立、编译、链接

[1-1] 建立一个用于存放工程的目录，该目录将用于保存和工程相关的所有文件（建议初学者每个工程都建立一个单独的目录）。

[1-2] 双击桌面 Keil µVision5图标，通过 IDE 环境的主菜单“Project”→“New µVision Project”创建Project，选择保存工程的目录和工程名。 选择Device为“STMicroelectronics

->STM32F4 Series->…->STM32F407ZG”如下图 1-1 所示（如果选择了其他Device，后续界面显示会略有差异）。

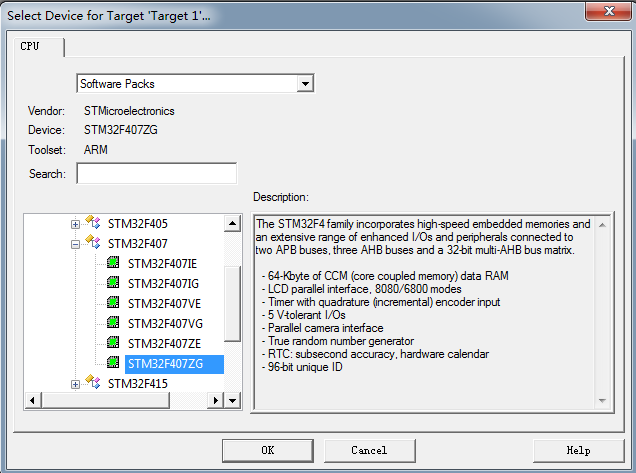


图 1-1 创建 Project 过程的 Device 选择

[1-3] 配置 Manage Run-Time Environment，选择 CMSIS 库模块和对应 Device 的启动文件 （Startup）。

* 勾选 CMSIS :: CORE Keil µVision 自带的 CMSIS 库的核心模块
* 勾选 Device :: Startup 对应 Device的启动文件（Startup）

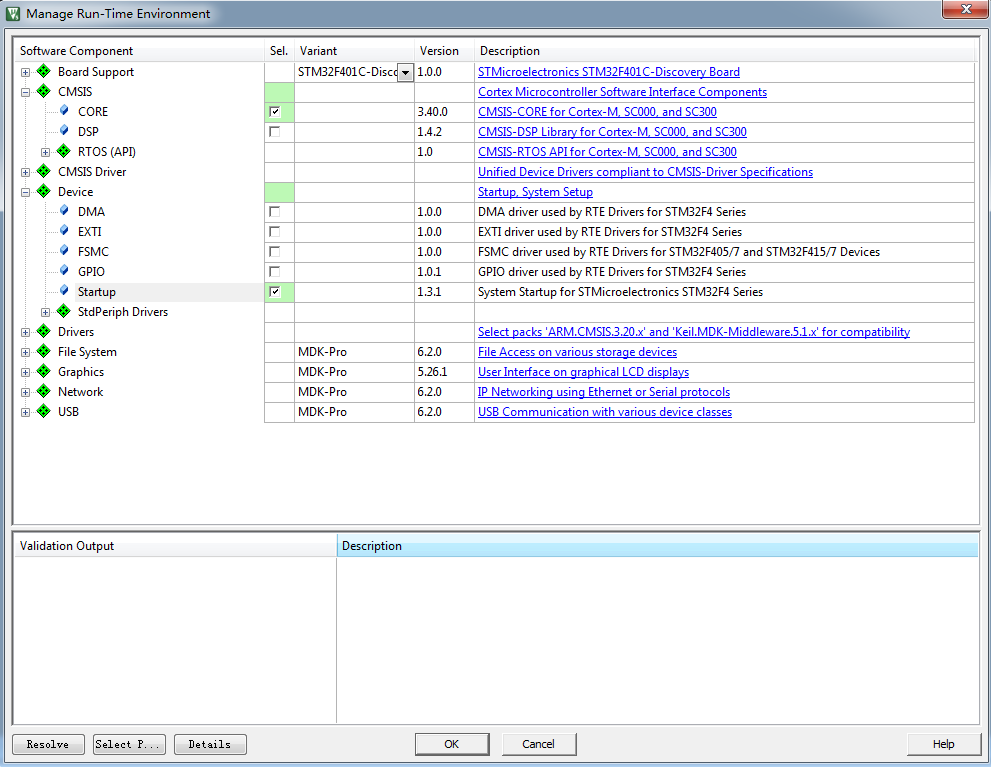


图 1-2 Manage Run-Time Environment 配置界面示意

[1-4] 工程建立完成后需要对工程进行必要配置，点击工具栏魔术棒图标“Options for Target”，如图1-3所示IDE已根据所选Device完成ROM和RAM地址空间配置。

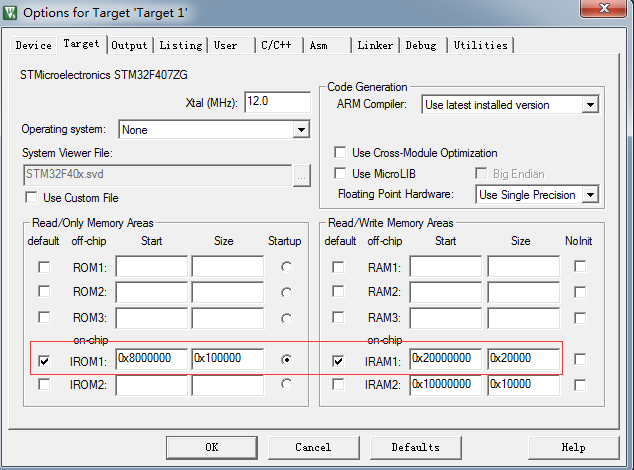


图 1-3 工程配置Target界面

切换至Debug页，选择仿真器类型，如图1-4所示从默认ULINK改为J-LINK。

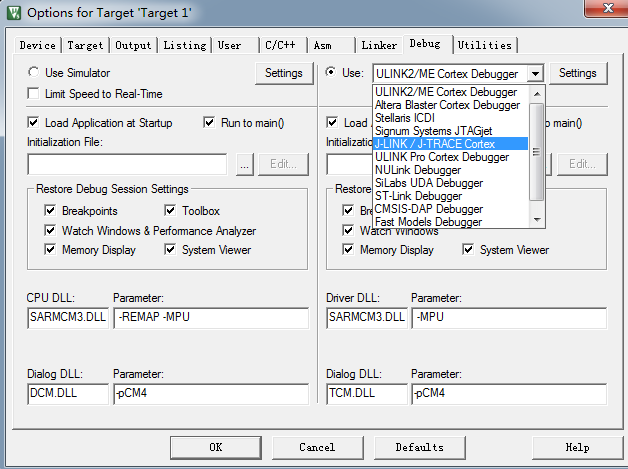


图 1-4 选择实验平台的仿真器类型

点击“Settings”按钮，显示实验平台使用的仿真器资源属性。

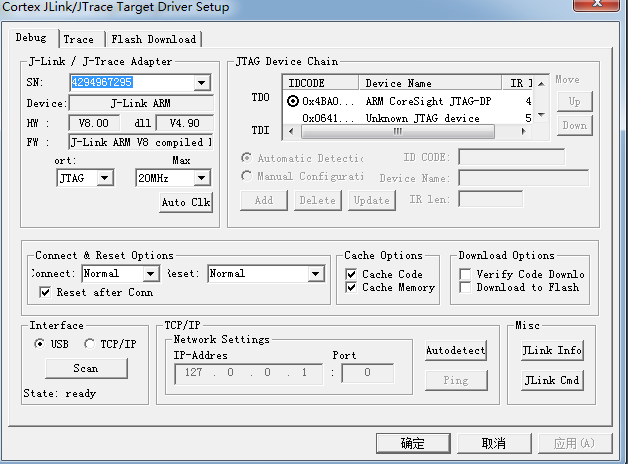


图 1-5 仿真器资源属性

至此工程建立完成，工程建立后必须添加相关源文件和包含头文件等。

[1-5]建立C源文件。File->New建立文件，并添加代码如下：

int main (void)

{

unsigned int ui\_tmp;

unsigned int ui\_a, ui\_b, ui\_c;

ui\_tmp = 255;

ui\_a = 1;

ui\_b = 2;

ui\_c = 0xFF;

}

添加代码后，保存文件为\*\*\*.C 表示保存为文件名不限的C源文件。

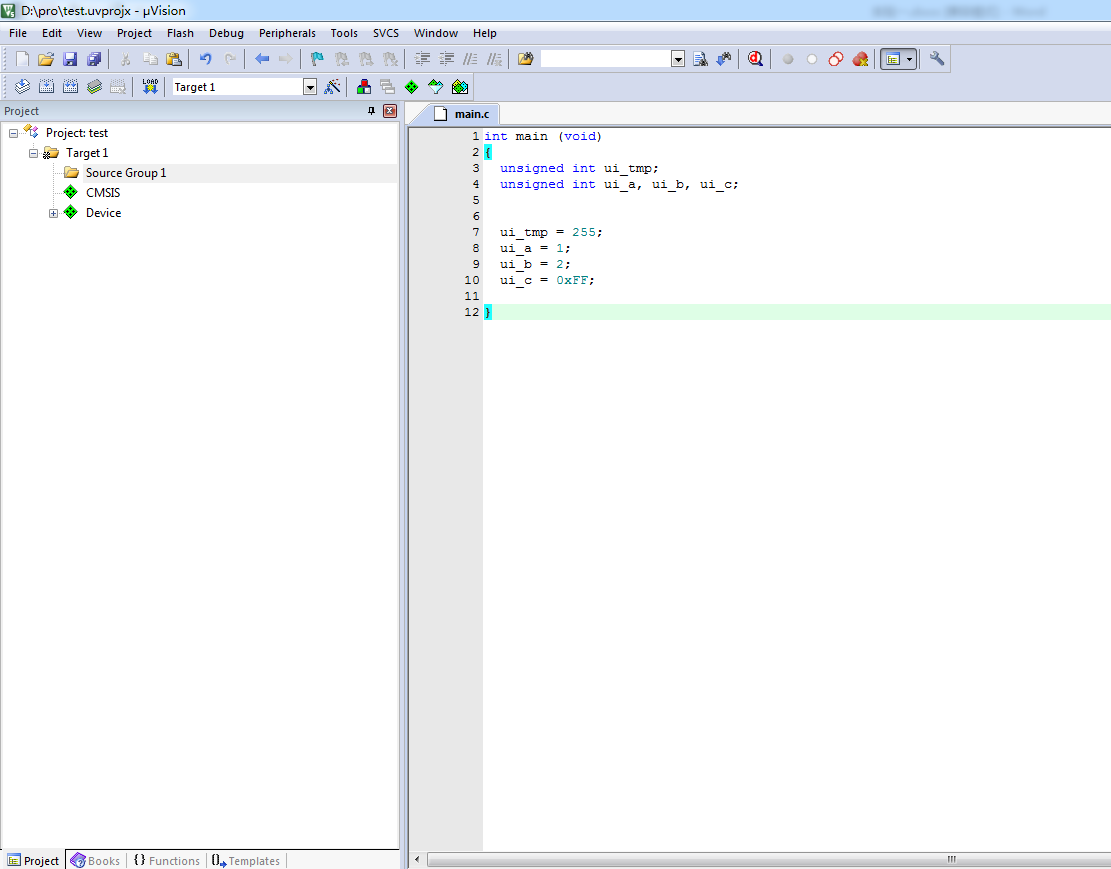
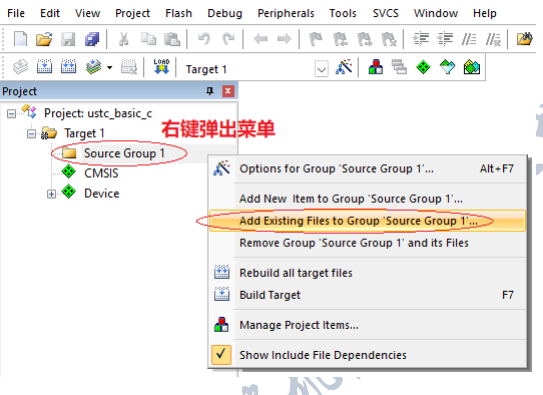


图 1-6 新建C源文件

[1-6] 添加源文件至 Project。



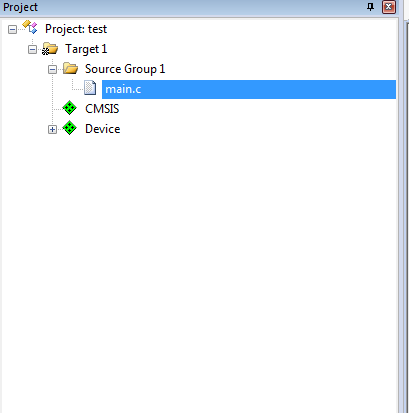
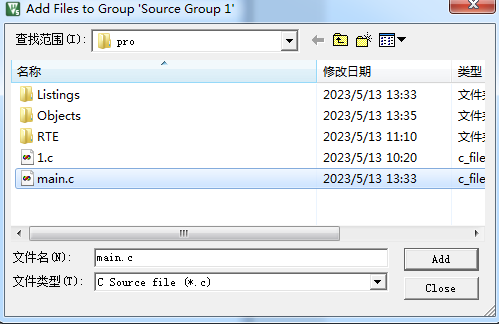


图 1-7 在 Project 中添加已有的 C 文件

[1-7] 编译、链接 Project。 可从主菜单“Project”选择“Build Target”，或采用热键“F7”，或者点击工具栏按钮，如图 1-8 所示。

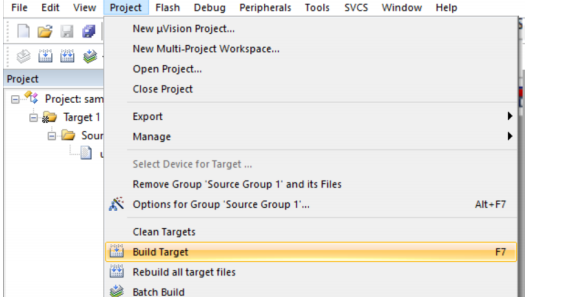


图 1-8 Build Target 操作示意

编译过程请注意观察 Build Output 窗口的输出，如图 1-9 所示。有错误，都会在 Build Output 窗口有输出提示。

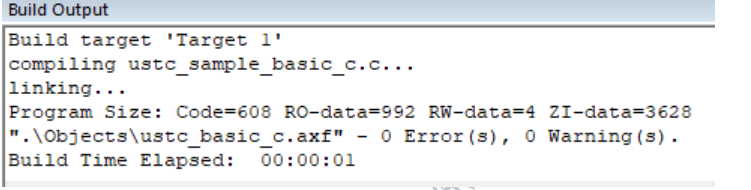


图 1-9 Build Output 窗口示意

注意：示例代码如有警告信息，可以在变量声明前加“volatile”关键字修饰。

volatile unsigned int ui\_tmp;

#### 1.2.2 Project 的调试（Debug）

[1-8] 程序编译链接无误后，点击工具栏，Download下载执行。在 main 函数第一行中设置断点（Breakpoint），以便后续采用单步跟踪方式执行程序。 在程序中设置断点有 3 种方式，如图 1-10 所示。

❑ 方法 1：点击菜单 Debug → Insert/Remove Breakpoint

❑ 方法 2：点击工具栏上设置断点图标

❑ 方法 3：用热键 F9

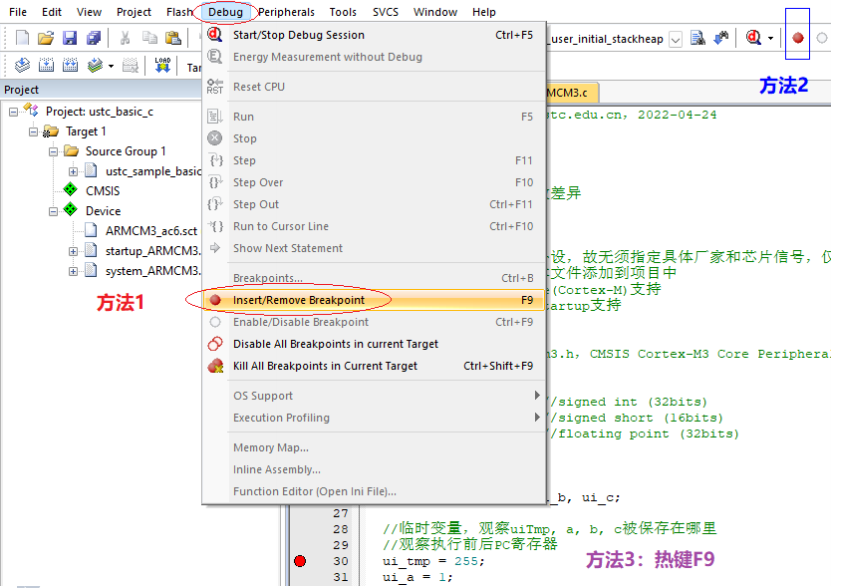


图 1-10 设置断点的 3 种途径（菜单、工具栏图标、热键 F9）

[1-9] 以调试方式执行程序（菜单 Debug → Start/Stop Debug Sesssion，或热键 CTRL+F5）。

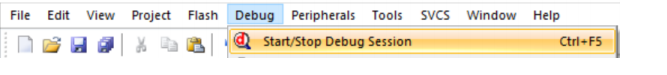


图 1-11 进入调试状态的 2 种方法（菜单、热键 CTRL+F5）

常用的程序调试操作包括：Run（运行到下一个断点），Step（单步运行、会进入子函数内部）， Step Over（单步运行、不会进入子函数内部），Run to cursor line（运行到光标位置）、设置/取消断点等功能，各项功能均可以通过点击菜单项或使用热键触发，如图 1-12 所示。请逐一测试功能，如果程序完毕或调试进入了不可理解的状态，可随时通过 CTRL+F5 结束本次调试，再重头开始调试。

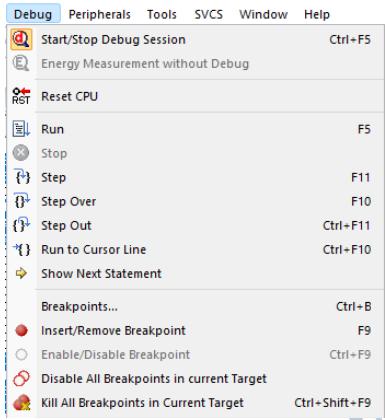


图 1-12 调试状态下可以进行的操作示意

注意：由于在执行 main()函数前需要先执行启动文件中代码，故而以调试方式运行程序 可能需要多次 Run（F5）才能到达 main()函数中的断点。

[1-10] 通过试验分析图 1-13 所示 Debug 工具栏（红色框内）各个图标对应功能的区别。

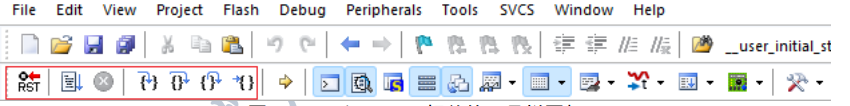


图 1-13 和 Debug 相关的工具栏图标

[1-11] 单步调试过程中观察通用寄存器的变化，即图 1-14 所示左边的子窗口内各个寄存器。

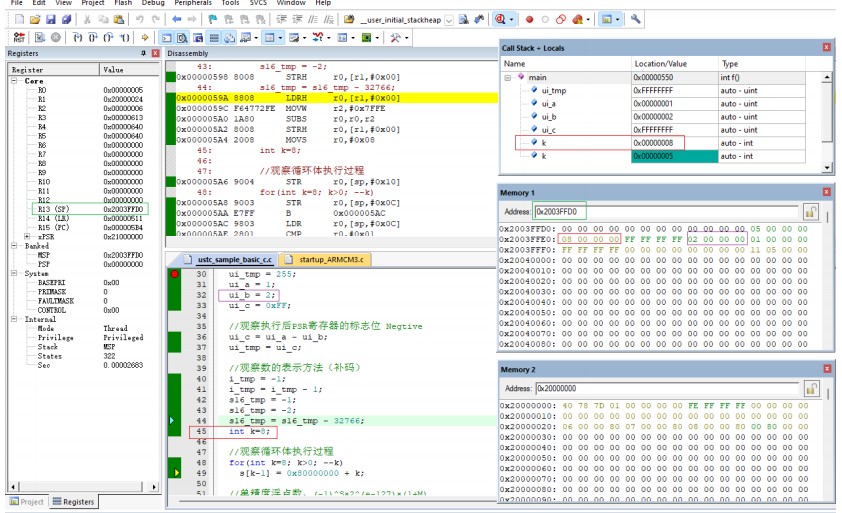


图 1-14 调试过程各个监测窗口示意图

[1-12] 单步调试过程中观察反汇编（Diassembly）窗口中汇编代码与 C 程序行的关系，如图 1-14 中上子窗口所示。课程不要求记住汇编指令，但是在 C 程序行和汇编指令映射关系确 定的情况下，应具备读懂汇编指令的能力。如果遇到反汇编（Diassembly）窗口未显示情况， 可以点击菜单 View → Diassembly Window。

[1-13] 单步调试过程中观察 Call Stack + Locals 子窗口的变量值变化情况，即图 1-14 所示右 上子窗口。如果遇到 Call Stack + Locals 子窗口未显示情况，可以点击菜单 View → Call Stack Window。

[1-14] 单步调试过程中观察 Memory 1 子窗口显示的存储器内容变化情况，即图 1-14 所示 右中子窗口，图 1-16 中该子窗口显示的起始地址是 SP 寄存器中保存的值。如果遇到 Memory1 子窗口未显示情况，可以点击菜单 View → Memory Window → Memory1。

[1-15] 单步调试过程中观察 Memory 2 子窗口显示的存储器内容变化情况，即图 1-14 所示 右下子窗口，图 1-14 中该子窗口显示的起始地址是 Cortex-M4 的 SRAM 区起始地址。如果 遇到 Memory2 子窗口未显示情况，可以点击菜单 View → Memory Window → Memory2。 MDK 支持看是 4 个 Memory Window。

[1-16] 通过 Watch 窗口（主菜单 View → Watch Window）观察一下代码执行前后变量值。 Watch Window 中可输入拟监控的变量，MDK 支持 2 个 Watch window。

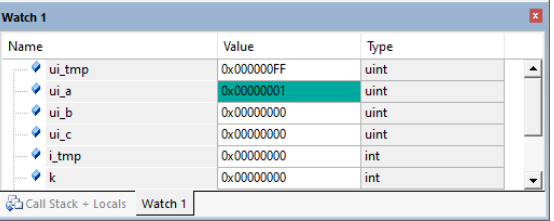


图 1-15 Watch Window 示意图

#### 1.2.3 µVision联机资源使用

[1-17] 打开联机帮助文档（菜单 Help → µVision Help），搜索“Disassembly Window”的作用。阅读返回结果中关于“Instruction Trace Window”的信息，如图 1-16 所示。

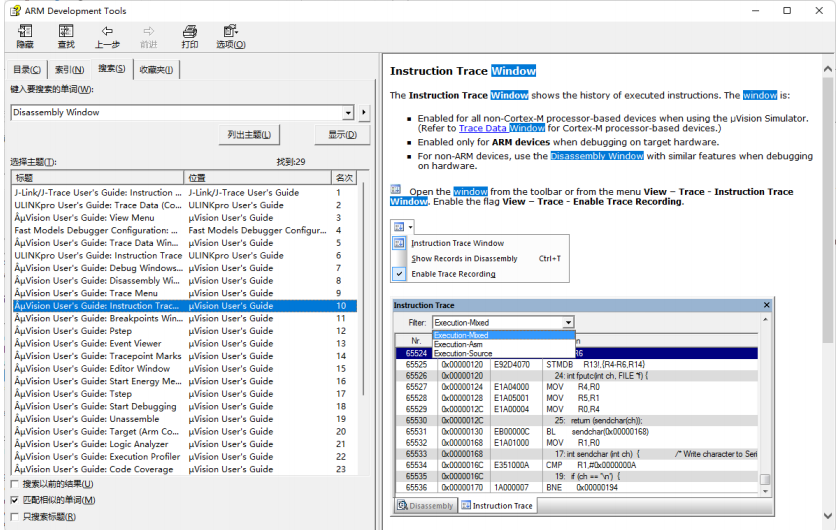


图 1-16 联机帮助搜索“Disassembly Window”的返回结果

### 1.3 实验练习及思考题

1. 观察以下变量存放格式并记录。

int main (void)

{

unsigned int ui\_tmp;

unsigned int ui\_a, ui\_b, ui\_c;

static int i\_tmp; //signed int (32bits)

static short s16\_tmp; //signed short (16bits)

static float f\_tmp; //floating point (32bits)

static int s[8];

int k;

//临时变量，观察uiTmp, a, b, c被保存在哪里

//观察执行前后PC寄存器

ui\_a = 1;

ui\_b = 2;

ui\_c = 0xFF;

//观察执行后PSR寄存器的标志位 Negtive

ui\_c = ui\_a - ui\_b;

ui\_tmp = ui\_c;

//观察数的表示方法（补码）

i\_tmp = -1;

i\_tmp = i\_tmp - 1;

s16\_tmp = -1;

s16\_tmp = -2;

s16\_tmp = s16\_tmp - 32766;

//观察循环体执行过程

for(k=8; k>0; --k)

s[k-1] = 0x80000000 + k;

//单精度浮点数，(-1)^S×2^(e-127)×(1+M)

f\_tmp = -0.5;

f\_tmp = f\_tmp + 1;

}

2. 编写子函数实现统计unsigned char型数据中二进制“1”功能。

例：unsigned char uc\_c=0x78;统计结果为4，提示：可使用C语言中“位与”及“移位”功能实现。

3. 使用 CMSIS-CORE 函数实现底层操作并记录读取内容。

阅读教材 P278~P282，特殊寄存器。单步执行程序中如下代码并记录，对比 ui\_tmp 的值和 Register 窗口观察到的数值是否一致，如图 1-17 所示。

注意：相关CMSIS-CORE函数调用需包含头文件#include "stm32f4xx.h"

ui\_tmp = \_\_get\_FAULTMASK(); //Get Fault Mask register

ui\_tmp = \_\_get\_BASEPRI(); //Get Base Priority register

ui\_tmp = \_\_get\_PRIMASK(); //Get Priority Mask Register

ui\_tmp = \_\_get\_CONTROL(); //Get CONTROL Register

ui\_tmp = \_\_get\_xPSR(); //Get xPSR Register

ui\_tmp = \_\_get\_MSP(); //Get Main Stack Pointer

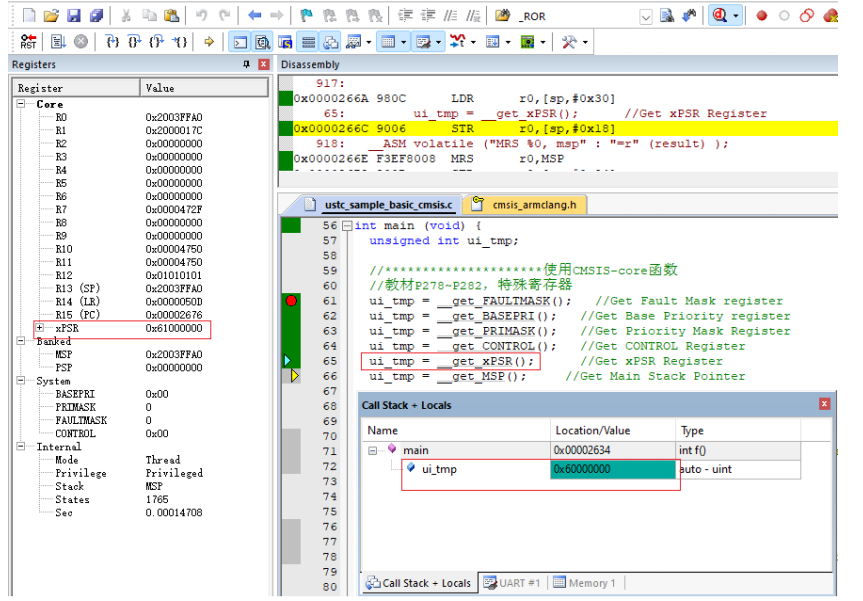


图 1-17 使用 CMSIS-core 函数访问寄存器

### 1.4 附录

## 附录STM32F407实验箱管脚约束

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 信号名 | 管脚 | 名称 | 信号名 | 管脚 |
| LED 灯 | D0 | PG11 | SEG 数码管 | SEG\_A | PG7 |
| D1 | PG10 | SEG\_B | PG6 |
| D2 | PG9 | SEG\_C | PG5 |
| D3 | PD7 | SEG\_D | PG4 |
| D4 | PG3 | SEG\_E | PA8 |
| D5 | PG2 | SEG\_F | PC7 |
| D6 | PD13 | SEG\_G | PC6 |
| D7 | PD12 | SEG\_DP | PG8 |
| DIP | DIP0 | PE4 | SEG\_S0 | PF13 |
| DIP1 | PE5 | SEG\_S1 | PG0 |
| DIP2 | PC14 | SEG\_S2 | PE9 |
| DIP3 | PC15 | SEG\_S3 | PF12 |
| DIP4 | PF0 | SEG\_S4 | PE8 |
| DIP5 | PF1 | SEG\_S5 | PE7 |
| DIP6 | PF2 | SEG\_S6 | PE10 |
| DIP7 | PF3 | SEG\_S7 | PF11 |
| 轻触开关 | SW\_RST | NRST | LCD12864 | LCD\_CS | PG1 |
| SW0 | PE0 | LCD\_SID | PF15 |
| SW1 | PE1 | LCD\_CLK | PF14 |
| SW2 | PE2 | RS232\_TOP | GPIO\_UART\_TXD | PA9 |
| SW3 | PE3 | GPIO\_UART\_RXD | PA10 |
| RS422 | RS422\_TXD | PA2 | DS1302 | RTC\_RST | PC5 |
| RS422\_RXD | PA3 | RTC\_IO | PB5 |
| DS18B20 | Tem\_IN | PG14 | RTC\_SCLK | PG15 |

第 7 章 附录 STM32F407 管脚约束

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 信号名 | 管脚 | 名称 | 信号名 | 管脚 |
| 矩阵键盘 | SW\_C0 | PF10 | 24LC02 | SCL\_I2C | PB8 |
| SW\_C1 | PC4 | SDA\_I2C | PB9 |
| SW\_C2 | PA0 | 93LC46 | SPI2\_CS | PB12 |
| SW\_C3 | PA1 | SPI2\_MISO | PB14 |
| SW\_R0 | PF4 | SPI2\_MOSI | PB15 |
| SW\_R1 | PF5 | SPI2\_SCK | PB13 |
| SW\_R2 | PF8 | PWM | PWM\_OUT1 | PB10 |
| SW\_R3 | PF9 | PWM\_OUT2 | PB11 |
| IR | INF\_IN | PB0 | SD | SDIO\_D0 | PC8 |
| ADC | ADC\_IN1 | PC3 | SDIO\_D1 | PC9 |
| ADC\_IN2 | PC2 | SDIO\_D2 | PC10 |
| ADC\_VOL | PC1 | SDIO\_D3 | PC11 |
| ADC\_VOL\_G | PC0 | SDIO\_CMD | PD2 |
| DAC | DAC\_OUT1 | PA4 | SDIO\_CLK | PC12 |
| DAC\_OUT2 | PA5 | VS1003 | VS\_MISO | PA6 |
| CAN/USB | USB\_D+ | PA12 | VS\_MOSI | PA7 |
| USB\_D- | PA11 | VS\_SCLK | PA5 |
|  |  |  | XCS | PF7 |
|  |  |  | XDCS | PF6 |
|  |  |  | DREQ | PC13 |
|  |  |  | XRESET | PE6 |