Application Note

SPC11X8/SPD11X8 J-LINK 使用指南

2021 年 1 月 – 版本 1

目录

表格列表

图片列表

1 J-LINK 与 SPC11X8/SPD11X8 连接

J-LINK 适配器支持 2 种接口，如图 1-1 所示。推荐使用 SWD 接口，因为更省引脚而且调试功能不受影响。该接口如表 1-1 所示。

图 1-1. J-LINK 接口

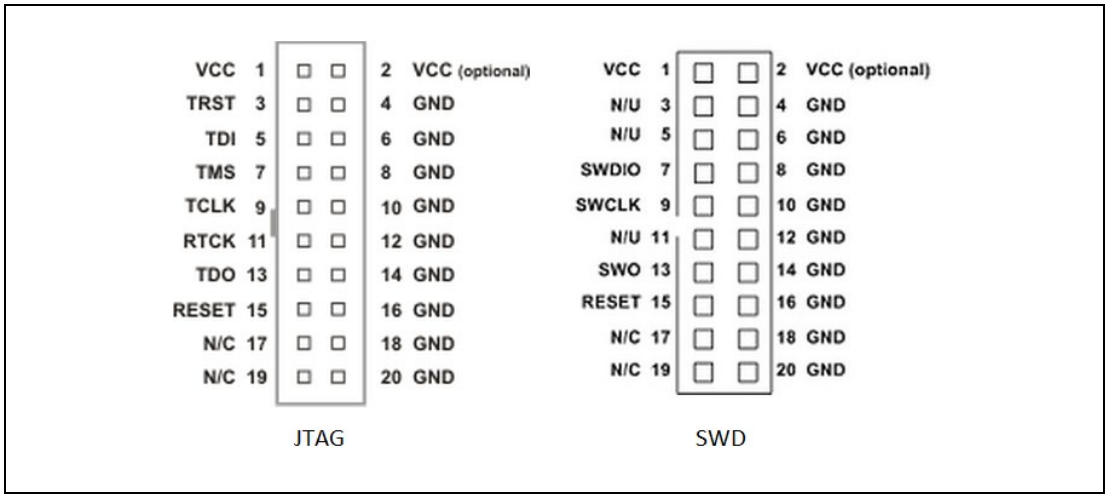


表 1-1. SW 接口信号定义

Signal	Connects to...
SWDIO	Data I/O pin
SWCLK	Clock pin
VCC	Positive Supply Voltage, the pin is optional.
GND	Digital ground
RESET	RSTIN pin, the pin is optional.
SWO	Serial data output, the pin is optional.

在采用 SPC11X8/SPD11X8 芯片进行应用开发的过程的中，需要经常使用 J-LINK 进行程序的调试。以 SPC1168 开发板为例，J-LINK 与 SPC11X8/SPD11X8 的硬件连接如图 1-2 所示。表 1-2 中为具体的 PIN 脚连接关系。

图 1-2. J-LINK 与 SPC11X8/SPD11X8 实物连接（以 SPC1168 开发板为例）

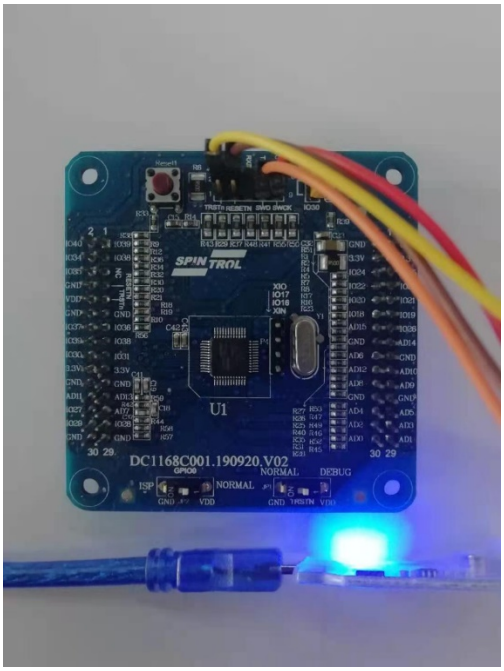


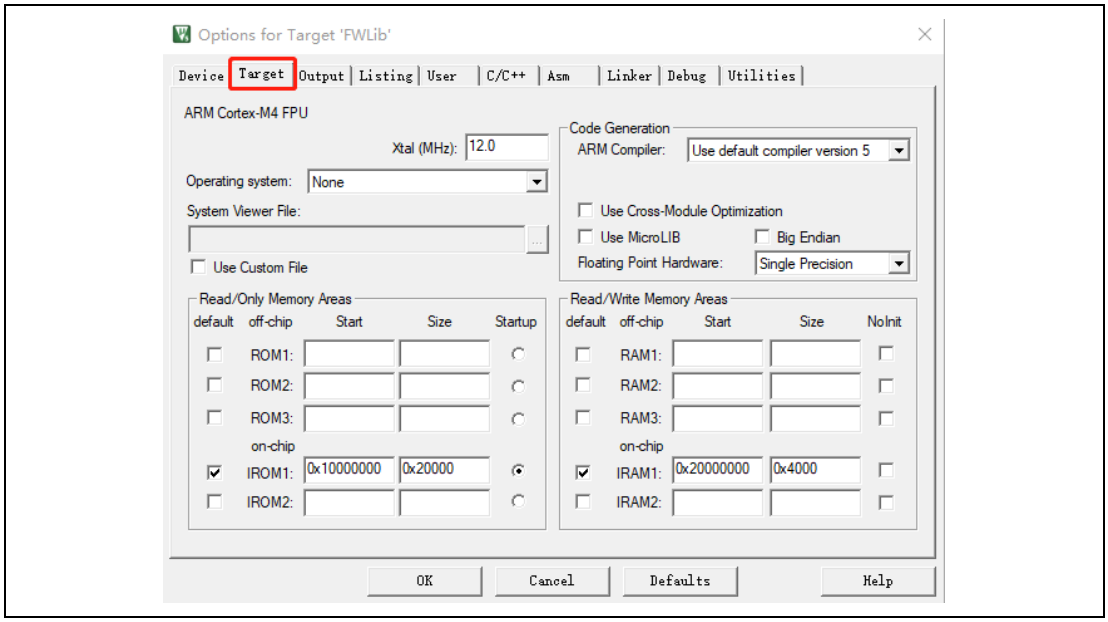
表 1-2. J-LINK 与 SPC11X8/SPD11X8 管脚连接

J-LINK	SPC11X8/SPD11X8
SWDIO	SWDIO(GPIO38)
SWCLK	SWCLK(GPIO39)
VCC	VDD
GND	GND

2 KEIL 环境下 J-LINK 配置

在安装 KEIL MDK 时，软件会默认安装 J-LINK 设备的驱动。按照表 1-1 将 J-LINK 与 SPC11X8/SPD11X8 连接，然后给芯片上电。这时打开 KEIL 软件，鼠标左键单击图标，弹出界面如下：

图 2-1. Options for Target 对话框



选择 Debug 选项卡，会看到如图 2-2 所示的界面。红色矩形框标记的内容是 Debug 时需要设置的选项。

图 2-2. Debug 配置界面

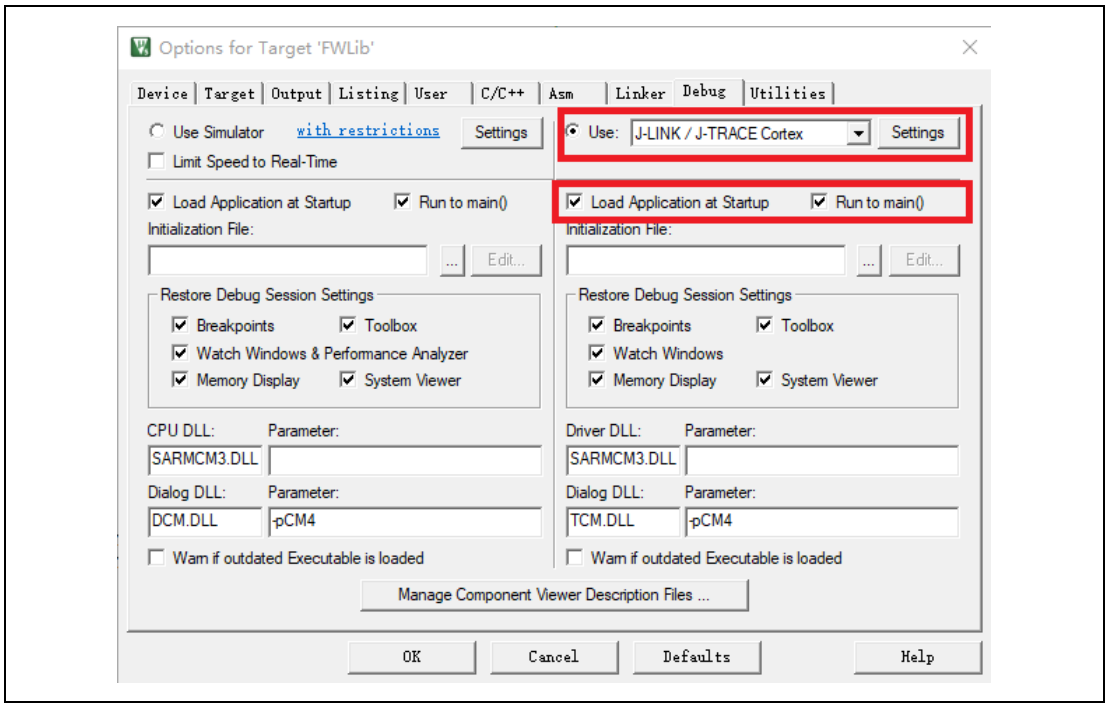
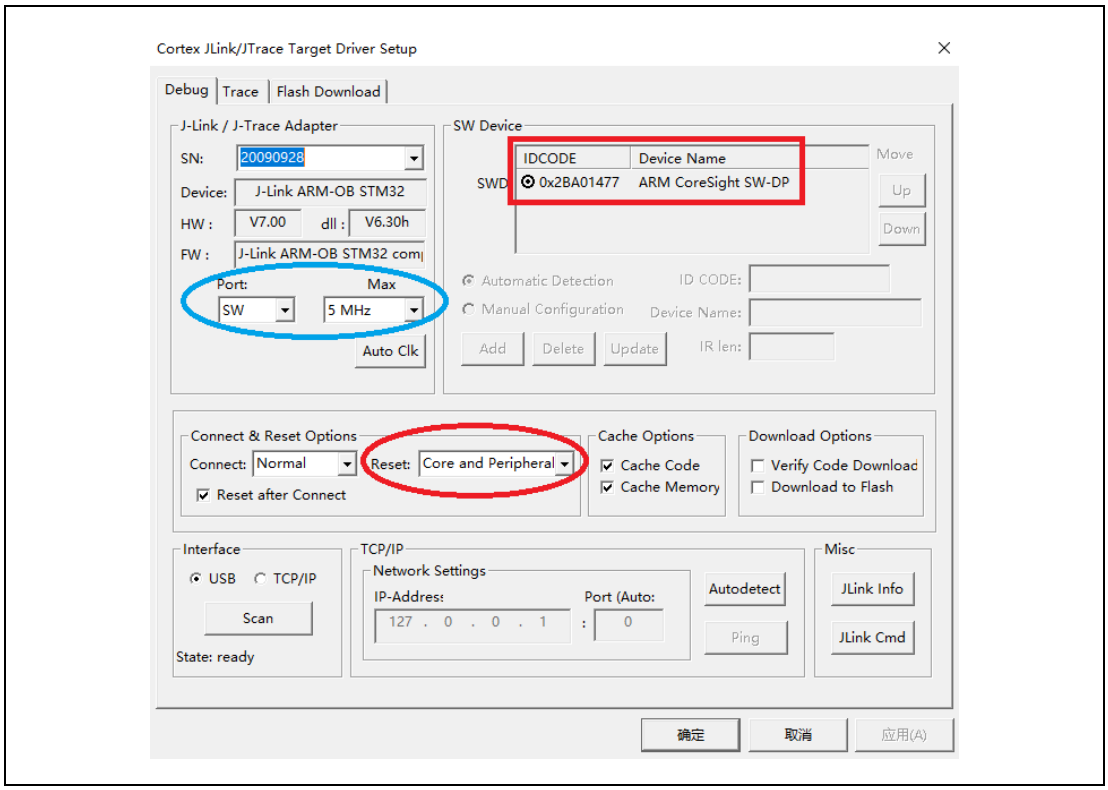


图 2-2 所示界面中，左侧是仿真调试相关的配置选项，右侧则是与硬件调试相关的选项。根据实际情形，选择使用 J-LINK/J TRACE Cortex 选项。单击 Settings 按钮，会弹出与 J-LINK 相关的设置，如图 2-3 所示。可以看到，红色矩形框中出现 Debug targets 的信息，表明 J-LINK 设备此时是正常工作的；否则，则表明 J-LINK 设备不可用。因此，在用 J-LINK 调试程序时，常常用此方法检查 J-LINK 设备是否正常。此外，建议用户按照图 2-3 配置 Connect & Reset Options，Reset 方式选择 Core and Peripheral。此外，SPC11X8/SPD11X8 芯片支持 JTAG 和 SWD 两种 Debug 协议，用户可以根据需要进行配置。

图 2-3. J-LINK 设置对话框



在使用 J-LINK 调试程序之前，还需要设置 Flash Download 选项，如图 2-4 所示。其中，SPC11X8/SPD11X8 Programming Algorithm 可以通过点击 Add 按钮来添加，如图 2-5 所示。（**注意：需要将 V1_x\IDE_Support\MDK-ARM 目录下的 SPC1168.FLM 文件复制到 KEIL 软件安装路径下的目录 Keil_v5\ARM\Flash**）

图 2-4. Flash Download 设置

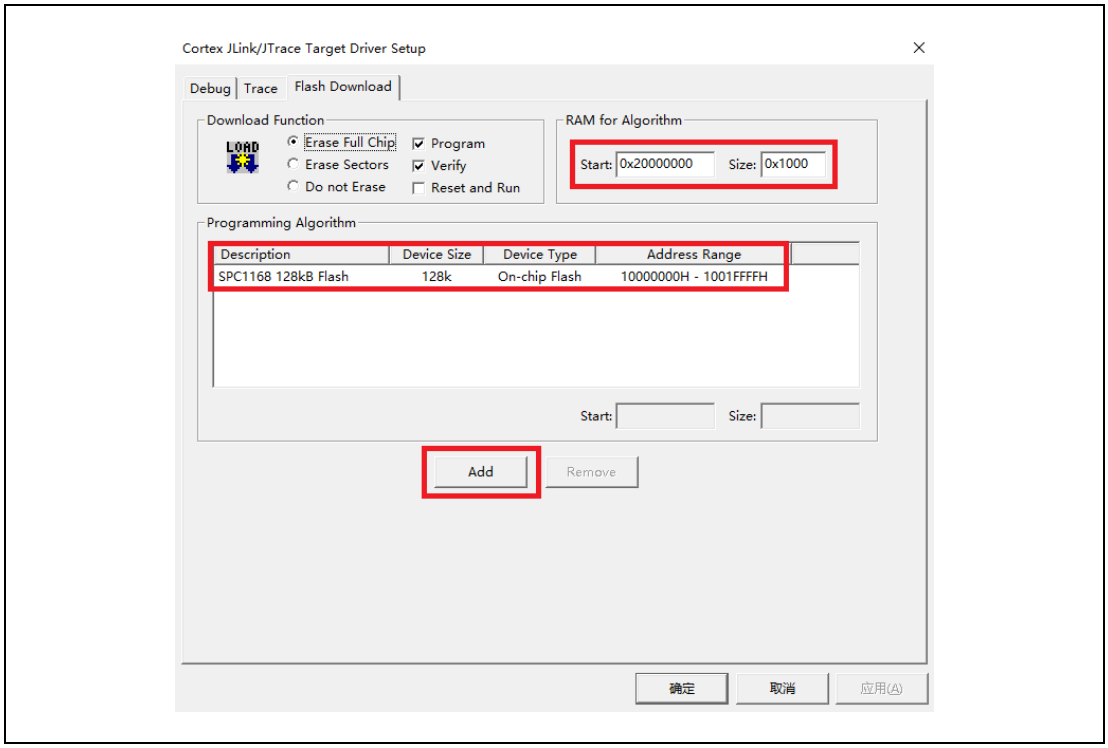
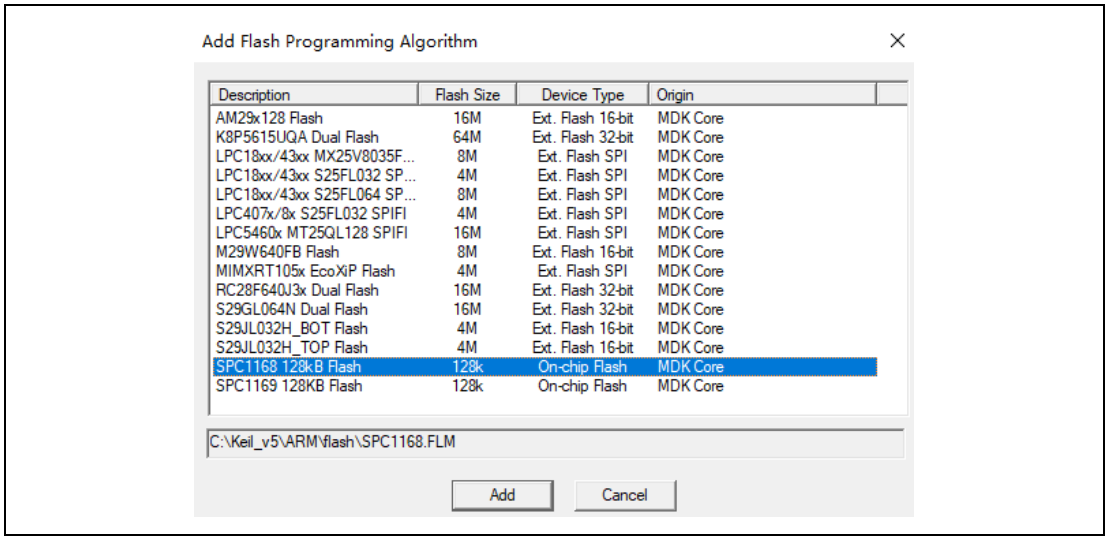
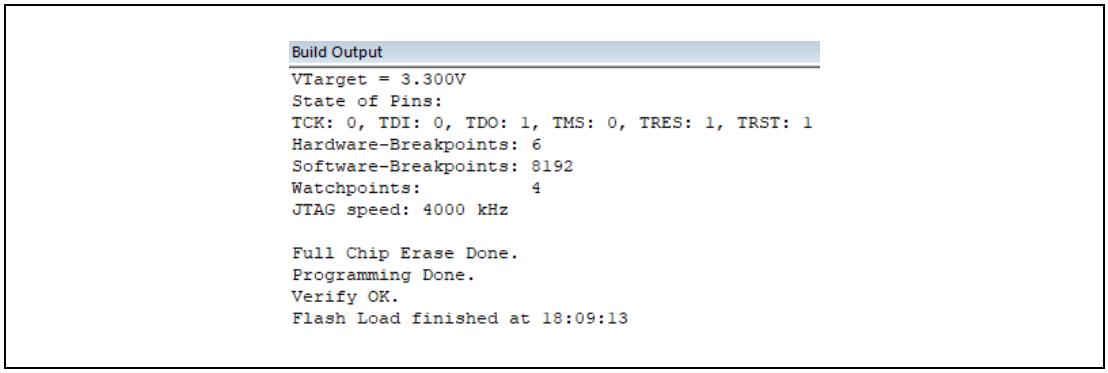


图 2-5. Add Flash Programming Algorithm



Flash Download 设置完成之后，将应用程序编译，然后点击 KEIL 软件工具栏上的按钮，就可以将应用程序下载到芯片中。用户可以在 Build Output 窗口中查看具体的 Download 过程信息，如图 2-6 所示。

图 2-6. Build Output 窗口信息

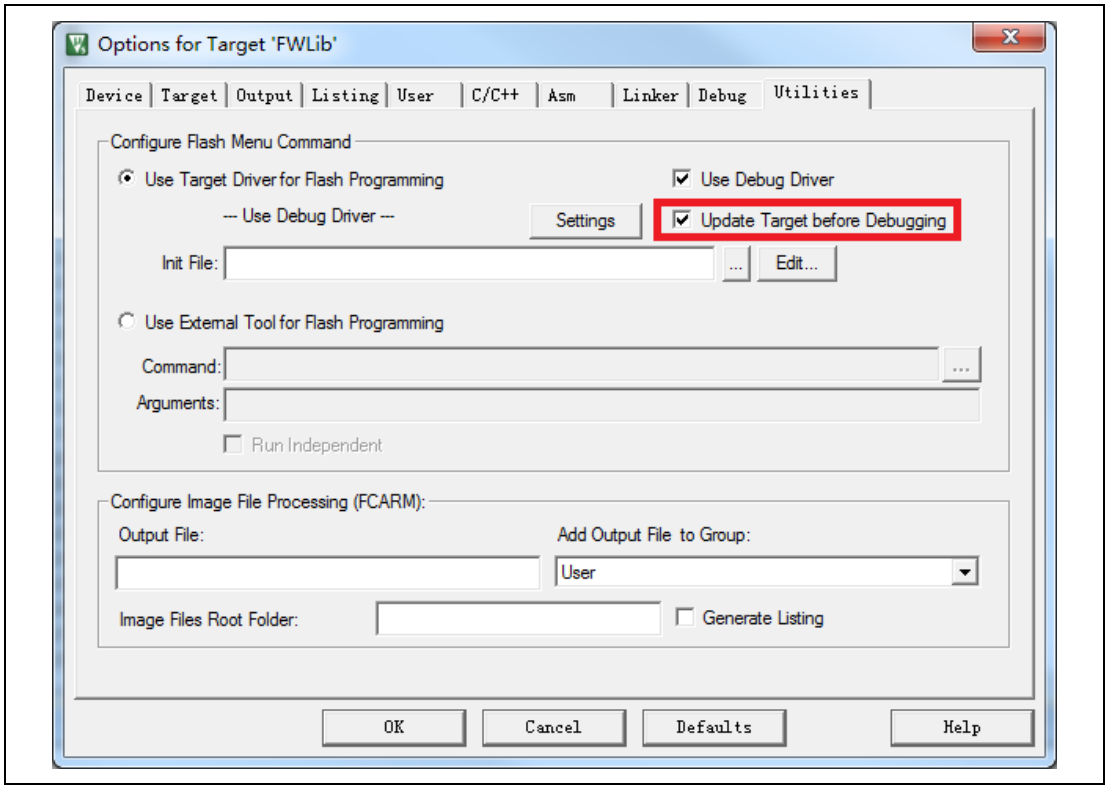


接下来，在图 2-2 中，我们看到有两个选项：Load Application at Startup 和 Run to main()。其中 Load Application at Startup 选项是必须要勾选的，Run to main()选项根据需要决定要不要勾选。

3 KEIL 环境下使用 J-LINK 调试

根据前面的介绍，将 J-LINK 设备与 SPC11X8/SPD11X8 正确连接后，按照图 2-2、图 2-3 以及图 2-4 设置 Debug 的相关选项，就可以使用 J-LINK 设备调试程序了。使用 J-LINK 调试程序时，必须保证 Flash 存储器中的程序与当前程序一致。这就需要用户每次修改代码后，都要点击按钮将程序下载到 Flash 存储器中。值得一提的是，KEIL 软件提供了一个功能，可以自动上述动作，如图 3-1 所示。用户只需勾选 Update Target before Debugging 选项，那么在每次启动 Debug 会话时，KEIL 软件会自动通过 J-LINK 设备将程序下载到 Flash 中，从而保证了 Flash 中的程序与当前调试的程序一致。

图 3-1. Update Target before Debugging 设置



单击工具栏上的按钮进入 Debug 状态，程序界面如图 3-2 所示。程序执行到 main 函数入口处后停止，等待用户的进一步操作。此时，KEIL 软件的界面也发生了变化：除了用户源代码窗口，还出现了汇编代码窗口和 CPU 寄存器窗口。在汇编代码窗口中，黄色底纹的汇编代码对应于用户代码窗口中光标所在位置的 C 代码；此外，菜单栏上也出现了一些与 Debug 相关的菜单选项，如表 3-1 所示。

注意：在程序进入 Debug 状态后，代码是不可以修改的。如果想修改代码，需要单击按钮退出 Debug 模式，然后才能修改代码。修改后的代码编译通过后，将代码重新下载到 Flash 中，用户可以继续单击按钮进行 Debug。

图 3-2. 启动 Debug 后的界面

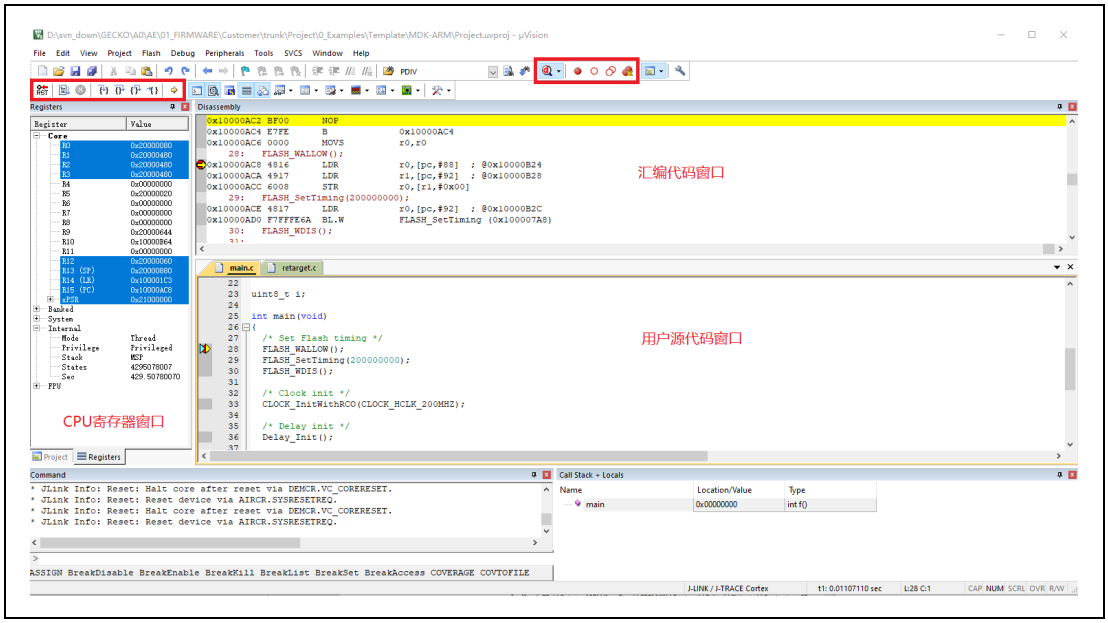


表 3-1. Debug Menu and Commands

Debug Menu	Toolbar	Shortcut	Description
Start/Stop Debug Session		**Ctrl+F5**	Starts or stops a debugging session.
Reset CPU			Sets the CPU to RESET state.
Run		**F5**	Continues executing the program until the next active breakpoint is reached.
Stop			Stops the program execution immediately.
Step		**F11**	Executes a single-step into a function; Executes the current instruction line.
Step Over		**F10**	Executes a single-step over a function.
Step Out		**Ctrl+F11**	Finishes executing the current function and stops afterwards.
Run to Cursor Line		**Ctrl+F10**	Executes the program until the current cursor line is reached.
Show Next Statement			Shows the next executable statement/instruction.
Breakpoints		**Ctrl+B**	Opens the dialog Breakpoints.
Insert/Remove Breakpoint		**F9**	Toggles the breakpoint on the current line.
Enable/Disable Breakpoint		**Ctrl+F9**	Enables/disables the breakpoint on the current line.
Disable All Breakpoints			Disables all breakpoints in the program.
Kill All Breakpoints		**Ctrl+Shift+F9**	Removes all breakpoints in the program.

3.1 单步调试

单击工具栏上的按钮后，程序进入 Debug 会话状态。此时单击工具栏上的按钮或者按下快捷键 F10 就可以单步执行程序。在单步调试的时候，用户代码窗口左侧边框处有两个三角箭头：表示当前光标所在的位置；表示当前位置的代码为下一次要执行的语句。因此，可以通过这两个三角箭头快速判断程序执行到哪条语句以及光标的位置。

有时候，我们希望程序能够快速地执行到某个位置，再进行单步调试。这时我们可以将光标定位到该位置，然后单击工具栏上的按钮，程序就会立即执行到当前光标处。此外，我们也可以通过设置断点的方式来实现上述功能。

3.2 断点设置

当启动 Debug 会话后，在用户代码所在行左侧边框处单击鼠标左键，即可快速地设置断点，此时左侧边框会出现一个红色的圆形标记，如图 3-3 所示。当然，也可以通过工具栏上的按钮设置断点，具体做法是：将光标定位到欲设置断点的代码行，然后单击按钮，即可设置断点。此时，单击工具栏上的按钮或者按下快捷键 F5，程序就会执行起来，一直执行到断点处停下来，如图 3-4 所示。

图 3-3. 设置断点

```
22
23  uint8_t i;
24
25  int main(void)
26  {
27    /* Set Flash timing */
28    FLASH_WALLOW();
29    FLASH_SetTiming(200000000);
30    FLASH_WDIS();
31
32    /* Clock init */
33    CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);
34
35    /* Delay init */
36    Delay_Init();
37
38    /* UART init */
39    GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);
40    GPIO_SetPinChannel(GPIO_35, GPIO35_UART_RXD);
41
42    UART_Init(UART, 38400);
43
44    PWM0->TBCTL.all = 0;
45    PWM0->TBCTL.all = 0x1234;
46
47    while(1)
48    {
49
50    }
```

图 3-4. 程序执行到断点

设置断点除了可以让程序快速的执行到某个位置外，在 Debug 程序时也非常有用。例如，可以在中断服务函数中设置断点，用来判断相应的中断是否发生。

3.3 观察变量值

在调试程序的时候，常常需要观察变量的值。这个可以通过 KEIL 软件提供的 Watch Window 来实现。具体实现过程如下：将光标定位到要观察的变量（光标移到变量名左侧），单击鼠标右键，在弹出的快捷菜单中即可将变量添加到观察窗口中，如图 3-5 所示。

从图 3-5 可以看出，我们将变量 i 添加到观察窗口 Watch1 里，结果如图 3-6 所示。从图中可以看出在代码区下方出现了 Watch1 窗口，在 Watch1 中可以看到刚刚添加的变量 i。

图 3-5. 添加变量到观察窗口

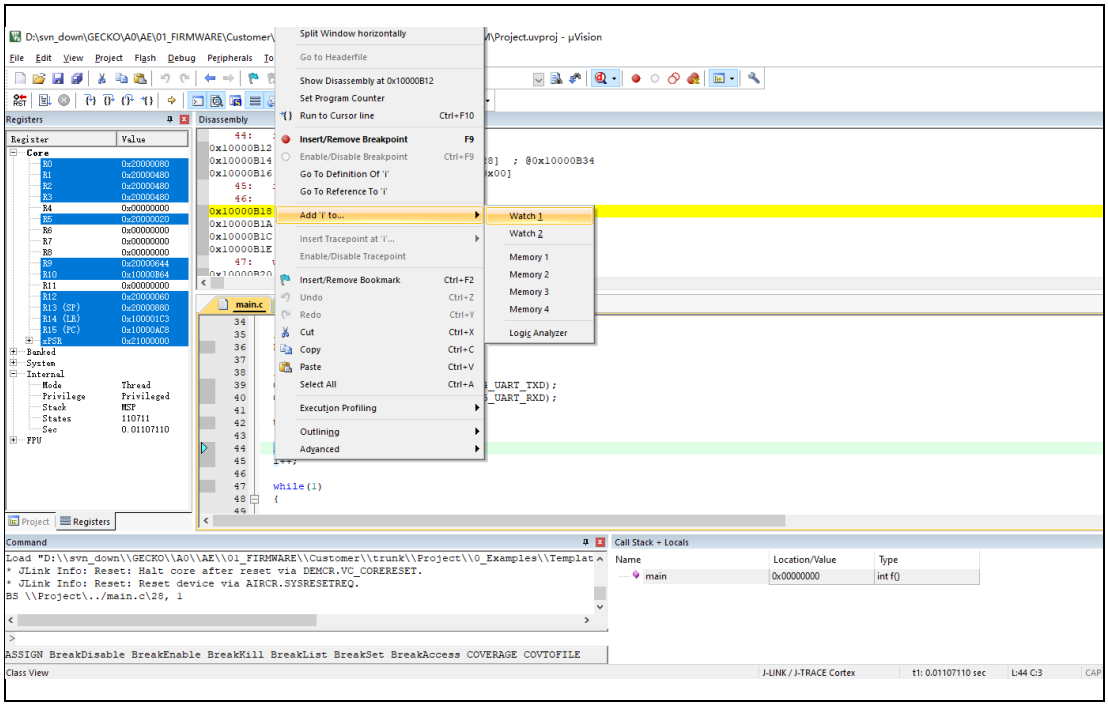
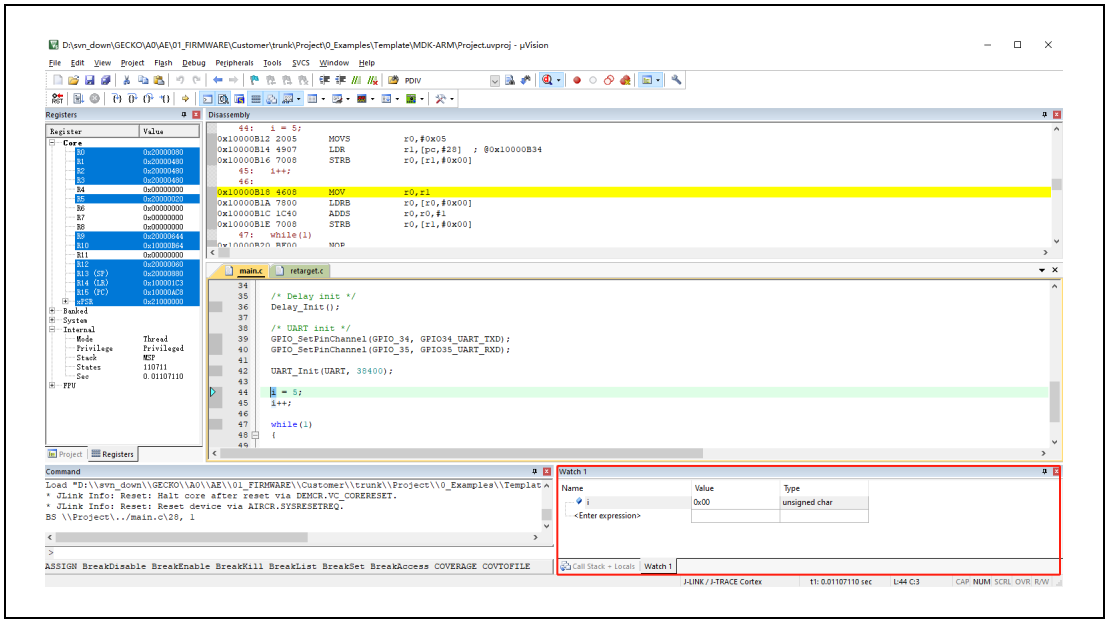


图 3-6. 添加变量到观察窗口的结果



接下来，我们就通过单步执行观察变量 i 的值。

第一步：单步执行语句 i = 5，执行结果如图 3-7 所示，可以看出变量 i 的值变为 5；

第二步：单步执行语句 i++，执行结果如图 3-8 所示，可以看出变量 i 的值变为 6。

图 3-7. i=5 执行结果

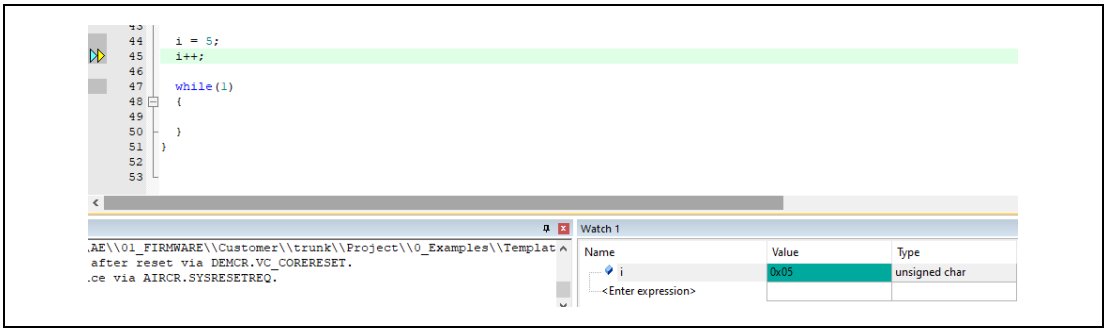
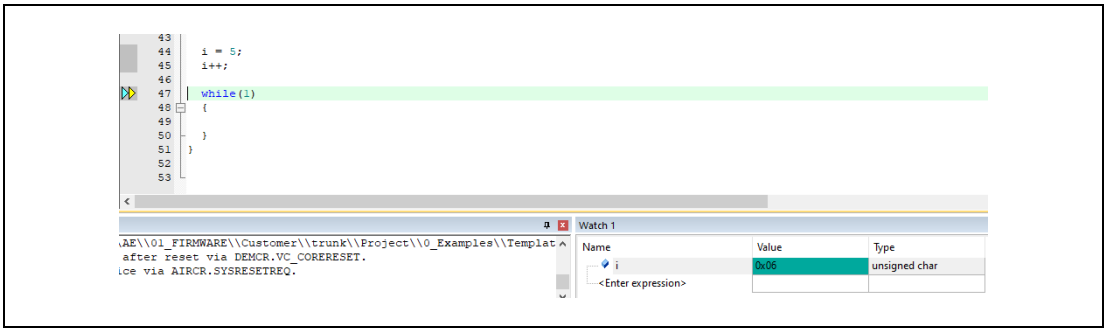


图 3-8. i++执行结果



3.4 观察外设寄存器

在调试程序的时候，我们不仅需要观察变量的值，也需要查看芯片外设 Register 的值。本节以芯片 SPC11X8/SPD11X8 外设模块 PWM0 为例介绍实现过程。

（1）通过 KEIL 软件添加芯片 System Viewer File。单击图标，在弹出的界面中勾选 Use Custom File 选项，然后单击图标，在弹出的对话框中选中 SPC11X8/SPD11X8.SFR 文件，位于目录 V1_x\IDE_Support\MDK-ARM 中。设置结果如图 3-9 所示。

（2）单击按钮，进入 Debug 模式，将芯片外设 PWM0 添加到 System Viewer 窗口，如图 3-10 所示。添加后的结果如图 3-11 所示。从图 3-11 可以看出，在 System Viewer 窗口中，不仅可以看到 PWM0 模块各个 Register 的值，而且还可以看到 Register 各个位段的值。

按照上面的步骤将 PWM0 添加到 System Viewer 窗口后，就可以在 Debug 程序的过程中观察到 PWM0 各个寄存器的值。我们单步执行图 3-11 中 main 函数的程序，分别将 TBCTL 寄存器赋值为 0 和 0x1234，结果分别如图 3-12 和图 3-13 所示。

图 3-9. System Viewer File 设置界面

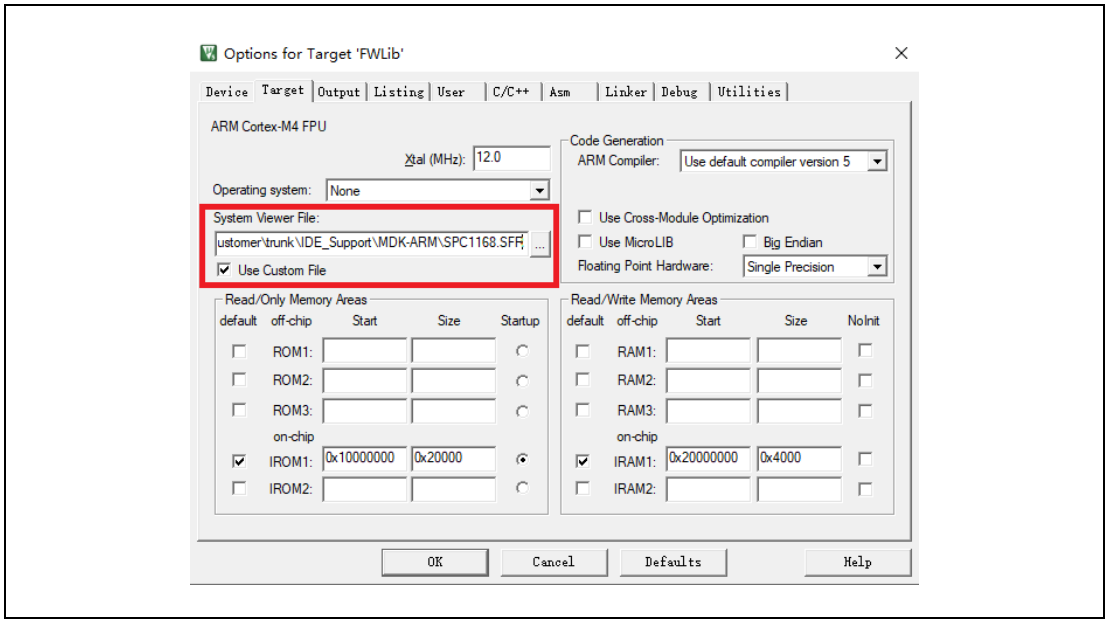


图 3-10. 添加 PWM0 到 System Viewer 窗口

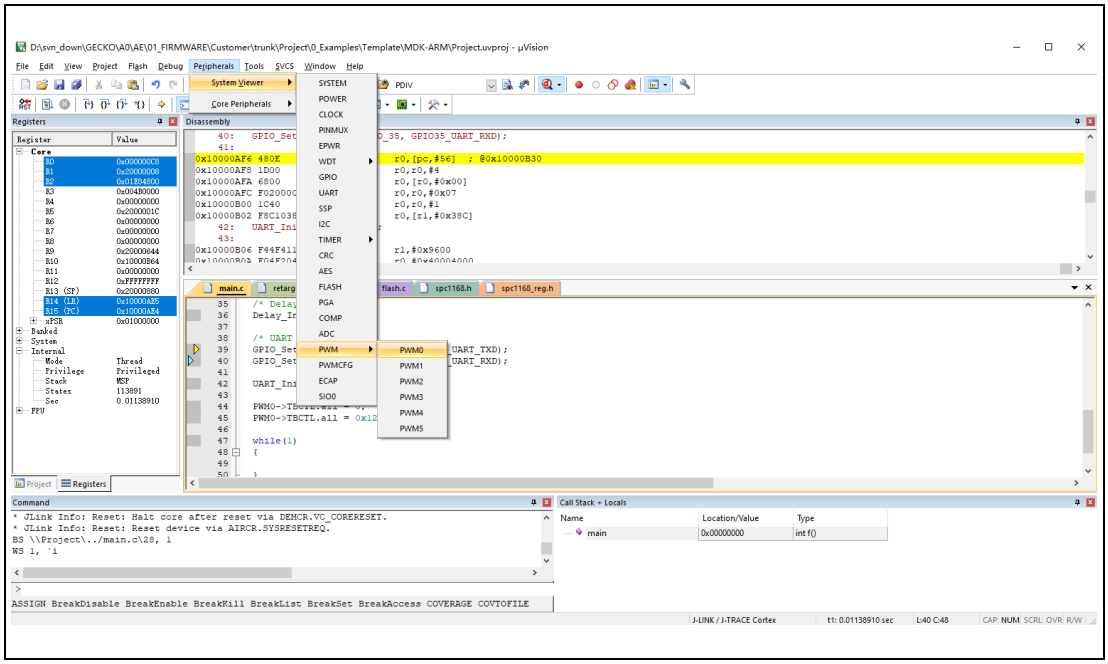


图 3-11. 添加 PWM0 到 System Viewer 窗口的结果

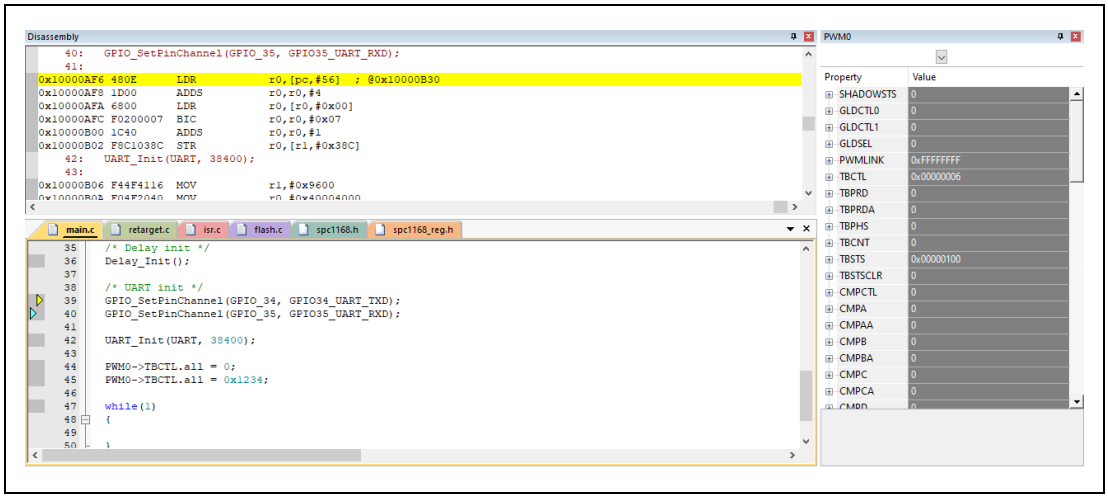


图 3-12. TBCTL=0 执行结果

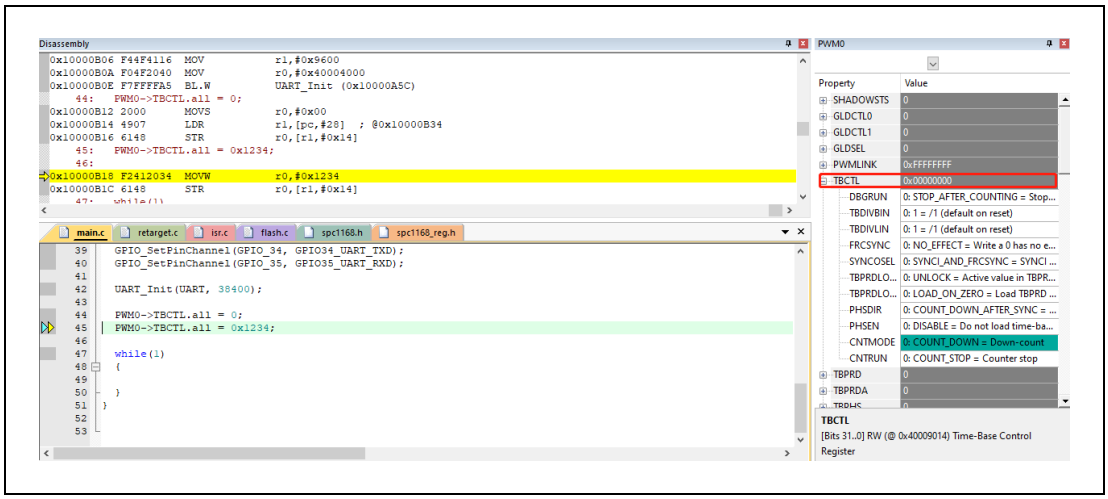
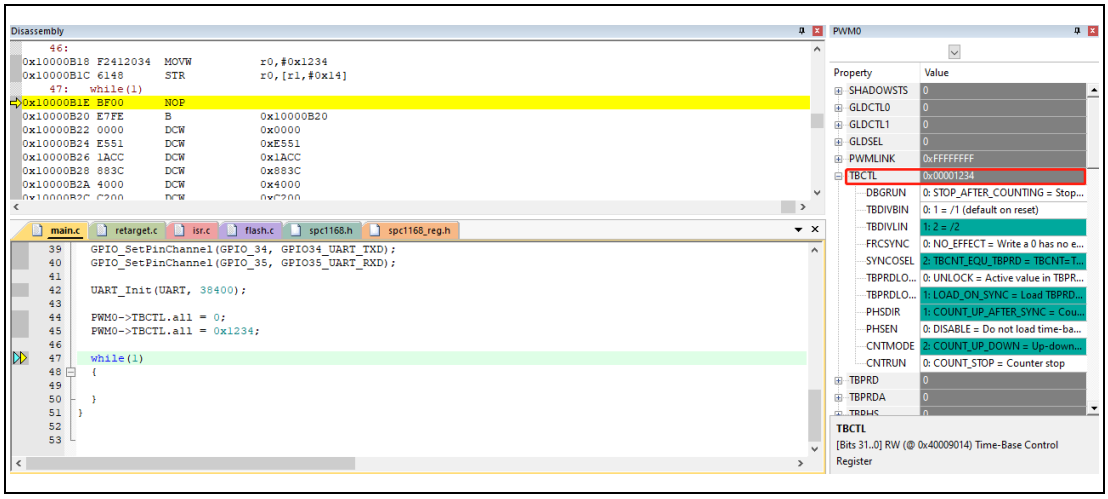


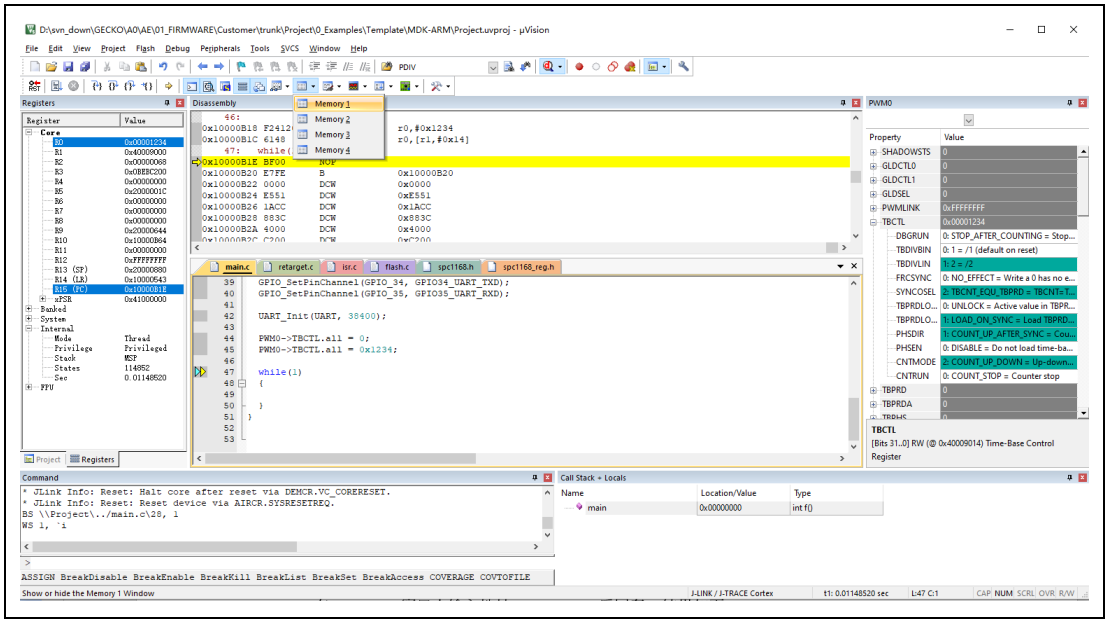
图 3-13. TBCTL=0x1234 执行结果



3.5 Memory 窗口

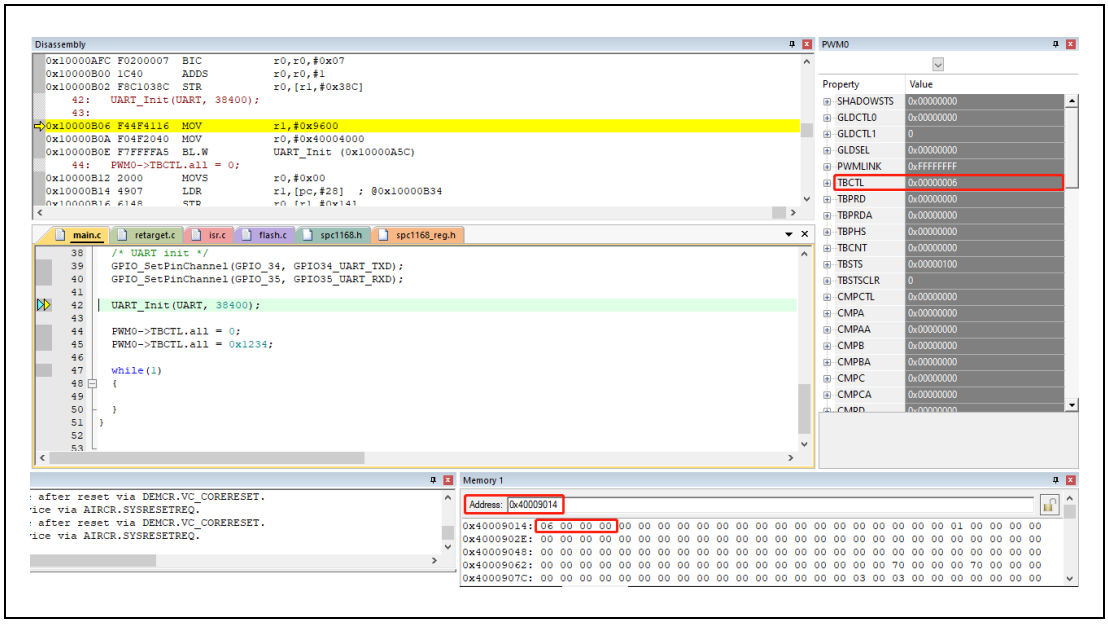
在 Debug 程序的过程中，我们还可以通过 Memory 窗口观察芯片内任一存储单元的地址。我们以章节 3.4 中的程序为例，其中 SPC11X8/SPD11X8 芯片 PWM0 模块 TBCTL 寄存器的地址为 0x40009014。首先，打开一个 Memory 观察窗口（Memory1），如图 3-14 所示。

图 3-14. 打开 Memory 观察窗口



在 Memory1 窗口中输入地址 0x40009014 后回车，结果如下：

图 3-15. Memory 窗口中观察到的 TBCTL 初始值



分别单步执行图 3-15 中 main 函数的程序，分别将 TBCTL 寄存器赋值为 0 和 0x1234，结果分别如图 3-16 和图 3-17 所示。

图 3-16. Memory 窗口结果（TBCTL=0）

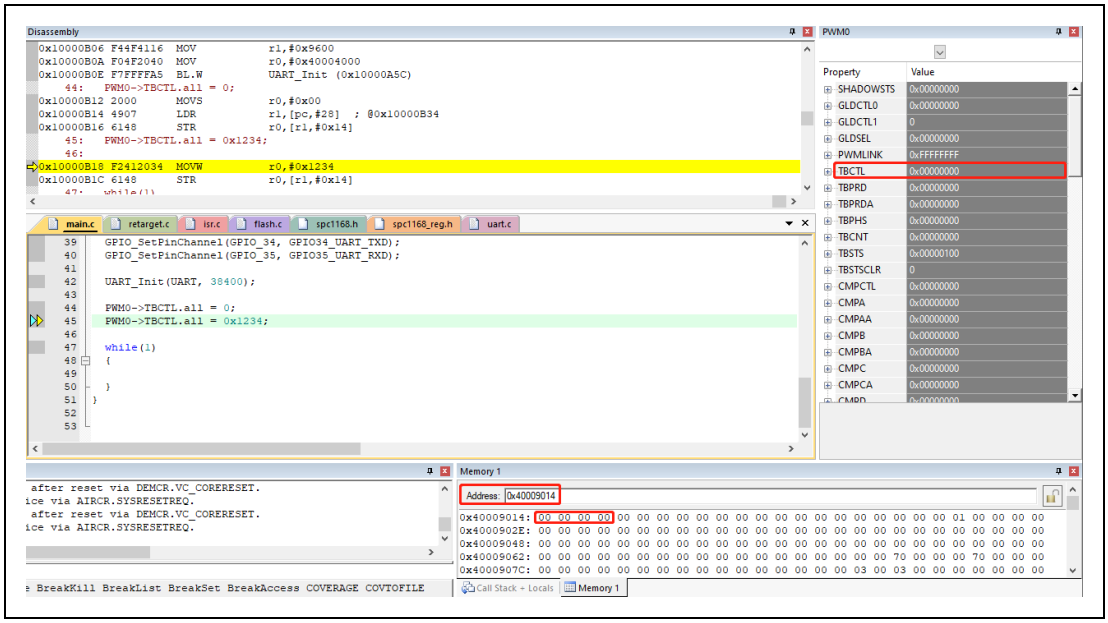
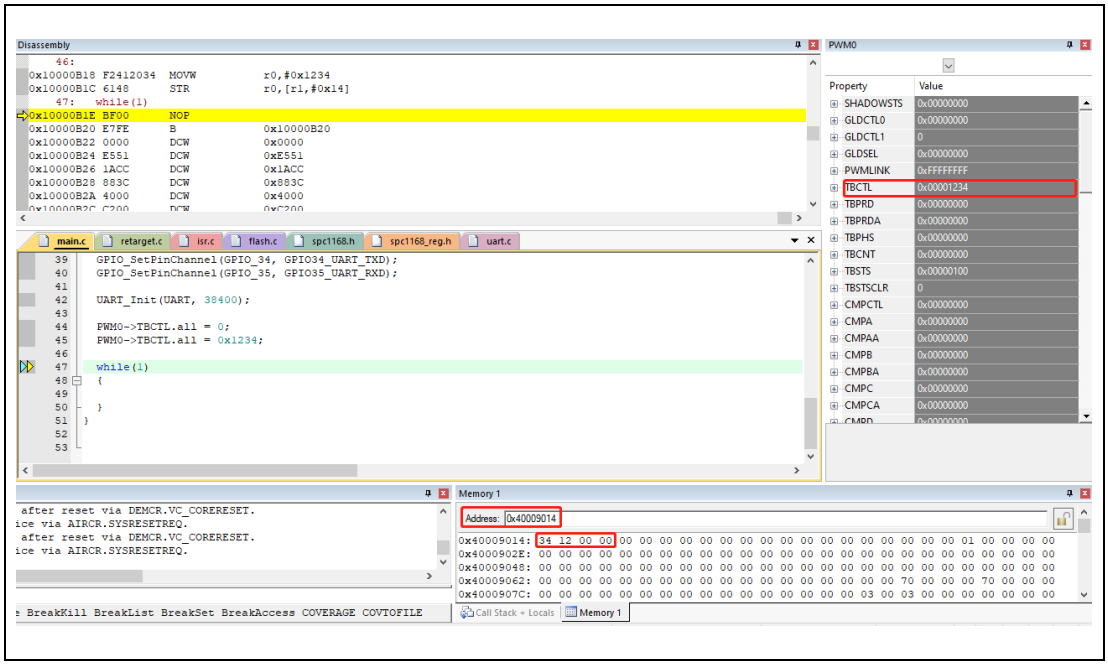


图 3-17. Memory 窗口结果（TBCTL=0x1234）



4 修订记录

表 4-1. 文档修订记录

日期	版本	修改内容
2021-01-23	1	初始版本