

概述

本手册适用范围：

适用范围	
SPC1125 系列	SPC1125, SPC1128, SPD1121
SPC1168 系列	SPC1155, SPC1156, SPC1158, SPC1168, SPD1148, SPD1178, SPD1188, SPD1163, SPM1173
SPC2168 系列	SPC2168, SPC2165, SPC2166, SPC1198
SPC1169 系列	SPC1169, SPD1179, SPD1176, SPD1177, SPD1179B
SPC2188 系列	SPC1185, SPC2188
SPC1198B 系列	SPC1198B

本手册中有关 Flash 及 RAM 地址范围的描述需要根据不同的产品进行设定。

目录

1	安装和配置软件.....	6
2	J-LINK 与目标板连接.....	6
3	KEIL 配置.....	9
4	程序调试.....	13
4.1	单步调试	15
4.2	断点设置	15
4.3	观察变量值	16
4.4	观察外设寄存器.....	18
4.5	观察 Memory.....	20
5	代码运行到 RAM.....	22

图片列表

图 2-1: J-LINK 接口	6
图 2-2: J-LINK 与 MCU 调试板接口	7
图 3-1: Options for Target 对话框	9
图 3-2: Debug 配置界面	9
图 3-3: J-LINK 设置对话框	10
图 3-4: Flash Download 设置	11
图 3-5: Add Flash Programming Algorithm	11
图 3-6: Build Output 窗口信息	12
图 4-1: Update Target before Debugging 设置	13
图 4-2: 启动 Debug 后的界面	14
图 4-3: 设置断点	15
图 4-4: 程序执行到断点	15
图 4-5: 添加变量到观察窗口	16
图 4-6: 添加变量到观察窗口的结果	16
图 4-7: i=5 执行结果	17
图 4-8: i++ 执行结果	17
图 4-9: System Viewer File 设置界面	18
图 4-10: 添加 UART0 到 System Viewer 窗口	19
图 4-11: 添加 UART0 到 System Viewer 窗口的结果	19
图 4-12: UART_Init 执行后结果	20
图 4-13: 开启 Memory1	20
图 4-14: UART_Init 执行后结果	21
图 4-15: UART_Init 执行后, 结果显示异常	21

表格列表

表 2-1: SWD 接口信号定义	6
表 2-2: 芯片与 BOOT 电平	7
表 2-3: 芯片调试接口电平	8
表 2-4: J-LINK 与各型号芯片的 SWD 管脚连接.....	8
表 4-1: Debug 菜单与指令	14

SPIN TROL

版本历史

版本	日期	作者	状态	变更
A/0	2023-11-03	HangSu	Outdated	1. 首次发布。
A/1	2023-11-29	HangSu	Outdated	1. 更新 章节 4.4 2. 增加 章节 4.5
C/0	2024-05-15	HangSu	Outdated	1. 增加 章节 2
C/1	2024-07-15	AngeloZhu	Outdated	1. 删除 章节 6 。 2. 更新 章节 2 。 3. 更新 章节 3 。 4. 修改为全系列通用文档。
C/2	2025-03-31	HangSu	Released	1. 增加 SPC1198B 系列。 2. 适用范围增加。

1 安装和配置软件

在开始使用 KEIL 软件之前，首先需要安装 KEIL，本文安装的版本是 MDK523。

2 J-LINK 与目标板连接

J-LINK 适配器支持 2 种接口，如图 2-1 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 2-1 所示。

图 2-1: J-LINK 接口

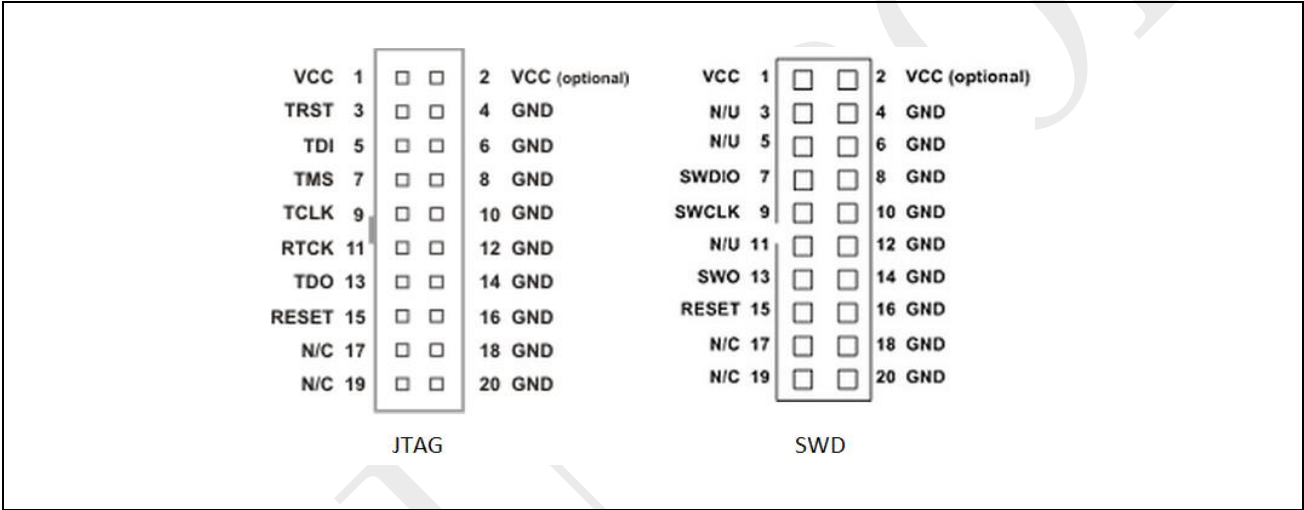
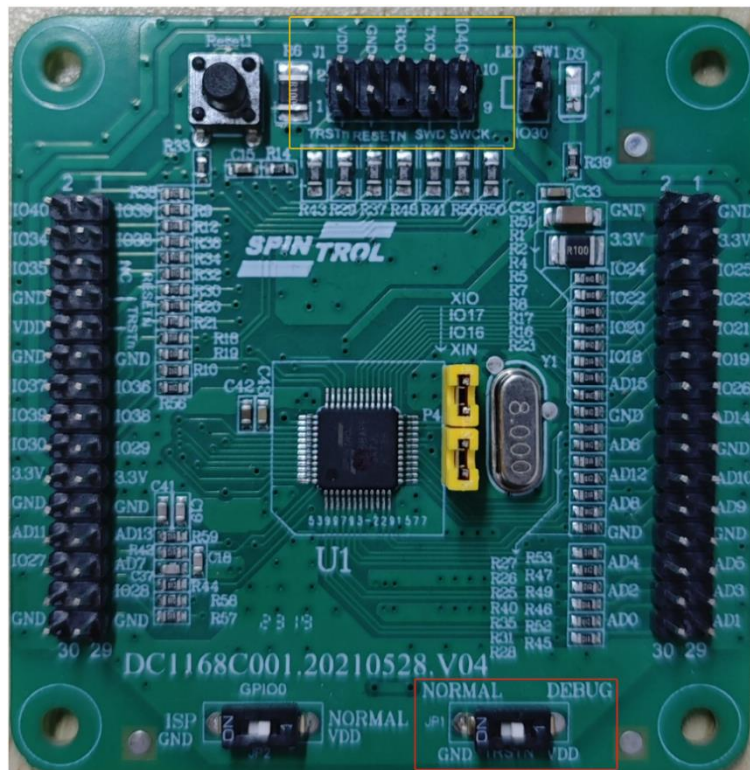


表 2-1: SWD 接口信号定义

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.

在使用芯片进行应用开发的过程中，需要经常使用 J-LINK 进行程序的调试。MCU 的硬件板卡与 JLINK 连接的接口如图 2-2 所示，表 2-4 中为具体的 PIN 脚连接关系。

图 2-2: J-LINK 与 MCU 调试板接口



图中所示为 SPC1168 调试板，红框内开关用于 MCU 的 TRSTN 引脚电平选择。
黄框内为调试板的 SWD 接口。

注意： 1. J-LINK 调试时，TRSTN 和 BOOT 电平必须与表 2-2 一致。
J-LINK 下载器端口电压需要与芯片端口电压表 2-3 一致。

表 2-2: 芯片与 BOOT 电平

芯片型号	BOOT	TRSTN
SPC1169 系列	X	高
SPC1168 系列	高	高
SPC2168 系列	高	高
SPC2188 系列	X	高
SPC1125 系列	X	高
SPC1198B 系列	X	/

[1] 图中 X 代表高低电平都可以。

[2] SPC1198B 系列没有 TRSTN 引脚。

注意： 芯片对应的 BOOT 和 TRSTN 管脚号，可通过查询对应的数据手册进行确认。

表 2-3： 芯片调试接口电平

芯片	调试接口电平
SPC1168 系列， SPC1125 系列， SPC2188 系列， SPC2168 系列， SPC1198B 系列	3.3V
SPC1169 系列	5V

注意： 1. 给 J-Link 引脚 1（VCC）接入表 2-3 参考电压前，需要确保其上无电压（部分 J-Link 调试器默认给引脚 1（VCC）接入 3.3V 电平），否则会烧芯片。

2. 具有默认电压（3.3V）的 J-Link 引脚 1（VCC），可以和调试接口电平为 3.3V 的芯片正常通信。

表 2-4： J-LINK 与各型号芯片的 SWD 管脚连接

芯片型号	SWD 引脚	
	SWDIO	SWCLK
SPC1169 系列	GPIO17	GPIO18
SPC1168 系列	GPIO38	GPIO39
SPC2168_CPU， SPC1198， SPC2166_CPU， SPC2165_CPU，	GPIO49	GPIO48
SPC2168_CAU	GPIO51	GPIO50
SPC2188_CPU0	GPIO80	GPIO81
SPC2188_CPU1	GPIO78	GPIO79
SPC1185	GPIO80	GPIO81
SPC1125 系列	GPIO38	GPIO39
SPC1198B 系列	SWD （芯片固定 Pad）	SWCK （芯片固定 Pad）

3 KEIL 配置


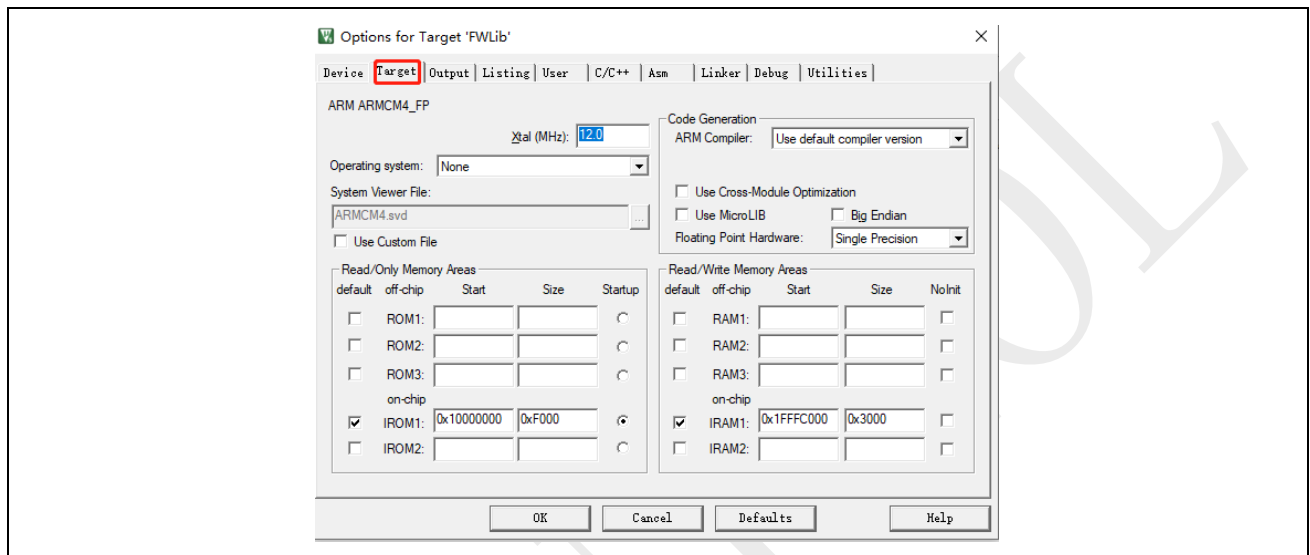
在安装 KEIL MDK 时，软件会默认安装 J-LINK 设备的驱动。按照图 2-2 将 J-LINK 与目标板连接，然后给芯片上电。这时打开 KEIL 软件，鼠标左键单击图标，弹出界面如下：

图 3-1: Options for Target 对话框



选择 Debug 选项卡，会看到如图 3-2 所示的界面。Settings 对应的红色矩形框标记的内容是 Debug 时需要设置的选项。Load Application at Startup 选择框用于指定在启动调试会话时是否自动加载应用程序到目标产品中，通常来说，为了保持调试时产品中的程序是最新的程序，此选项要勾选；Run to main 选择框指定是否在调试开始时 PC 运行至 main 函数入口，是否勾选根据开发者视具体需要而定。

图 3-2: Debug 配置界面

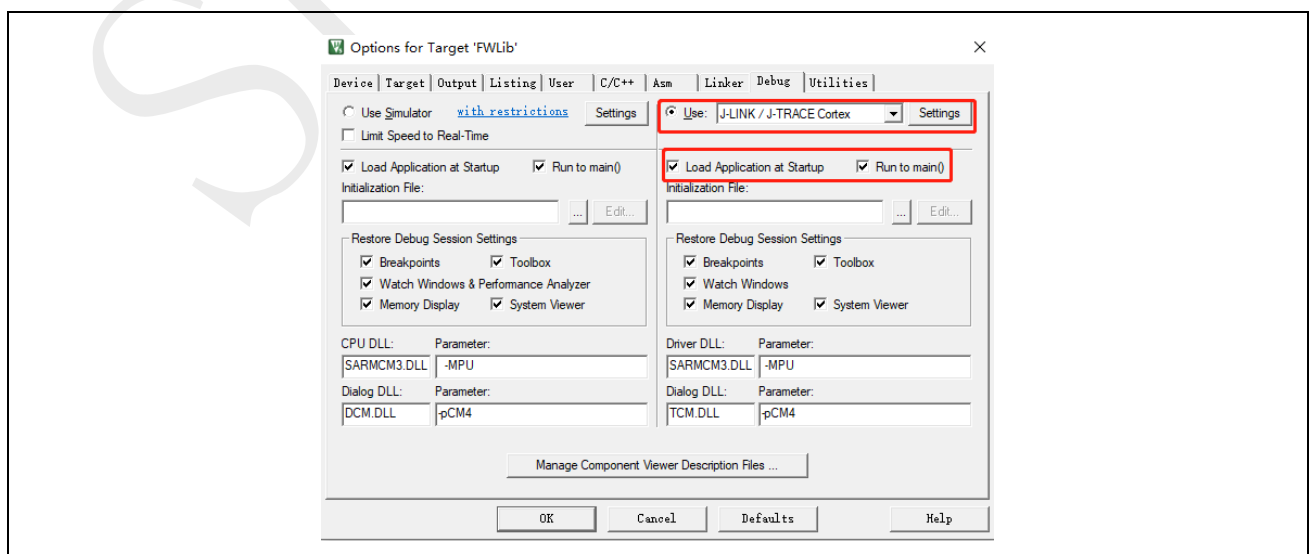
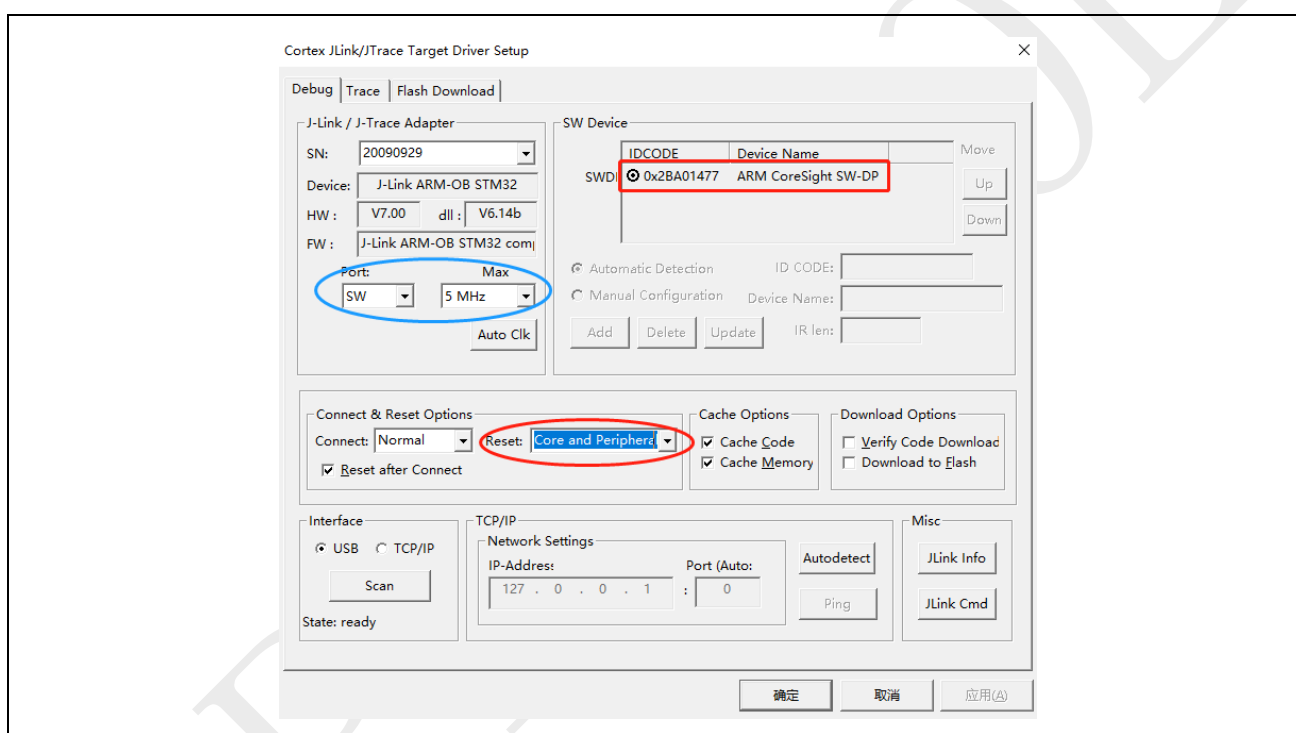


图 3-2 所示界面中，左侧是仿真调试相关的配置选项，右侧则是与硬件调试相关的选项。根据实际情形，选择使用 J-LINK/J TRACE Cortex 选项。单击按钮，会弹出与 J-LINK 相关的设置，如图 3-3 所示。可以看到，红色矩形框中出现 IDCODE 信息，表明 J-LINK 设备此时工作正常，否则表明 J-LINK 设备不可用。因此，在用 J-LINK 调试程序时，常常用此方法检查 J-LINK 设备是否正常。建议用户按照图 3-3 配置 Connect & Reset Options，Reset 方式选择 Core and Peripheral。此外，需要根据实际接线情况选择 JTAG 和 SWD 两种 Debug 协议其中的一种。

注意： J-LINK 下载器端口电压需要与芯片端口电压一致，否则可能无法出现红色矩形框中 IDCODE 信息。

图 3-3: J-LINK 设置对话框



- [1] 图中蓝色圈内选择调试接口和速率
- [2] 图中红色圈内选择 Reset 方式

在使用 J-LINK 调试程序之前，还需要设置 Flash Download 选项，如图 3-4 所示。其中 Programming Algorithm 可以通过点击 Add 按钮来添加，如图 3-5 所示。

注意： 需在开始 Debug 之前将 IDE_Support\MDK-ARM 目录下的 FLM 文件复制到 KEIL 软件安装路径下的目录 Keil_v5\ARM\Firmware，否则点击“Add”按钮之后看不到对应产品的 Flash 算法文件。

图 3-4: Flash Download 设置

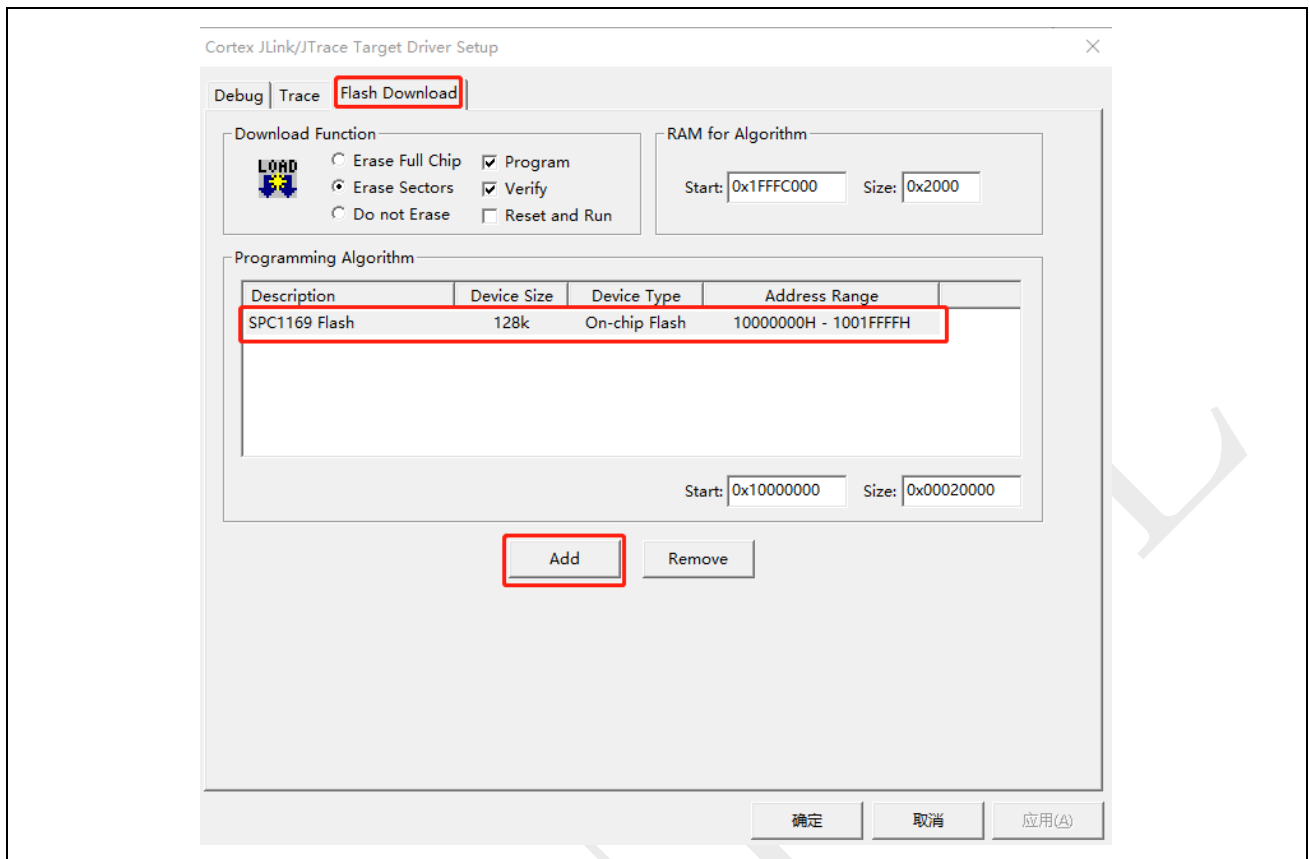
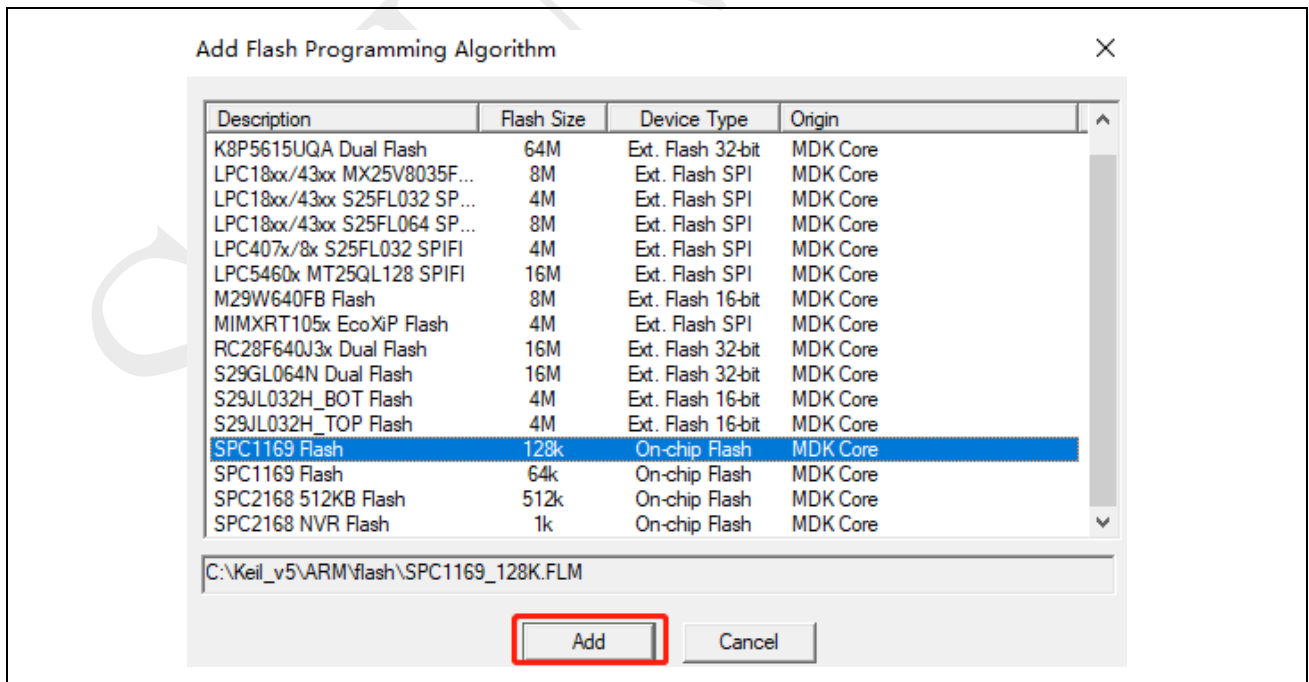


图 3-5: Add Flash Programming Algorithm




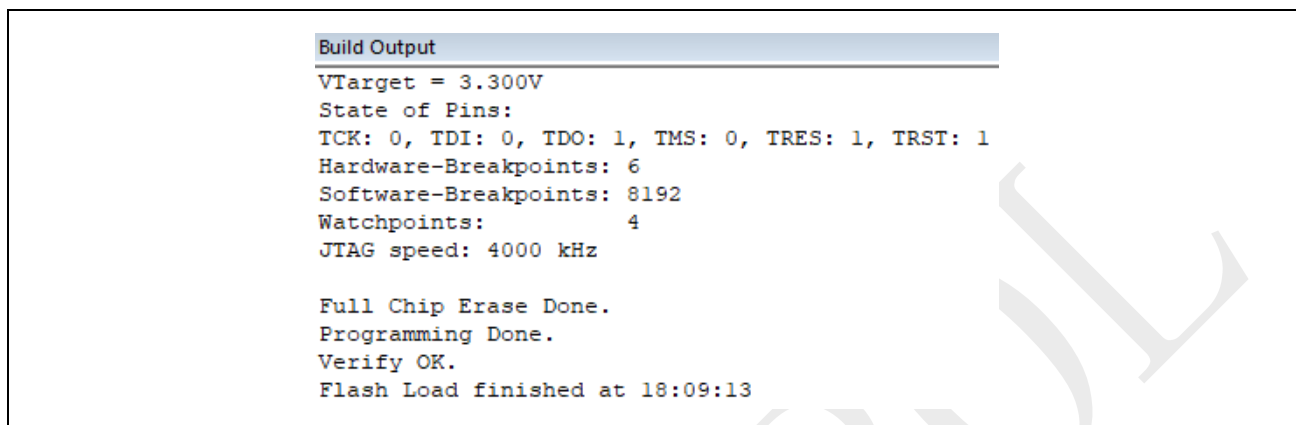
Flash Download 设置完成之后，将应用程序编译，然后点击 KEIL 软件工具栏上的  按钮，就可以将应用程序下载到芯片中。用户可以在 Build Output 窗口中查看具体的 Download 过程信息，如图 3-6 所示。

图 3-6: Build Output 窗口信息



```
Build Output
VTarget = 3.300V
State of Pins:
TCK: 0, TDI: 0, TDO: 1, TMS: 0, TRES: 1, TRST: 1
Hardware-Breakpoints: 6
Software-Breakpoints: 8192
Watchpoints: 4
JTAG speed: 4000 kHz

Full Chip Erase Done.
Programming Done.
Verify OK.
Flash Load finished at 18:09:13
```

4 程序调试



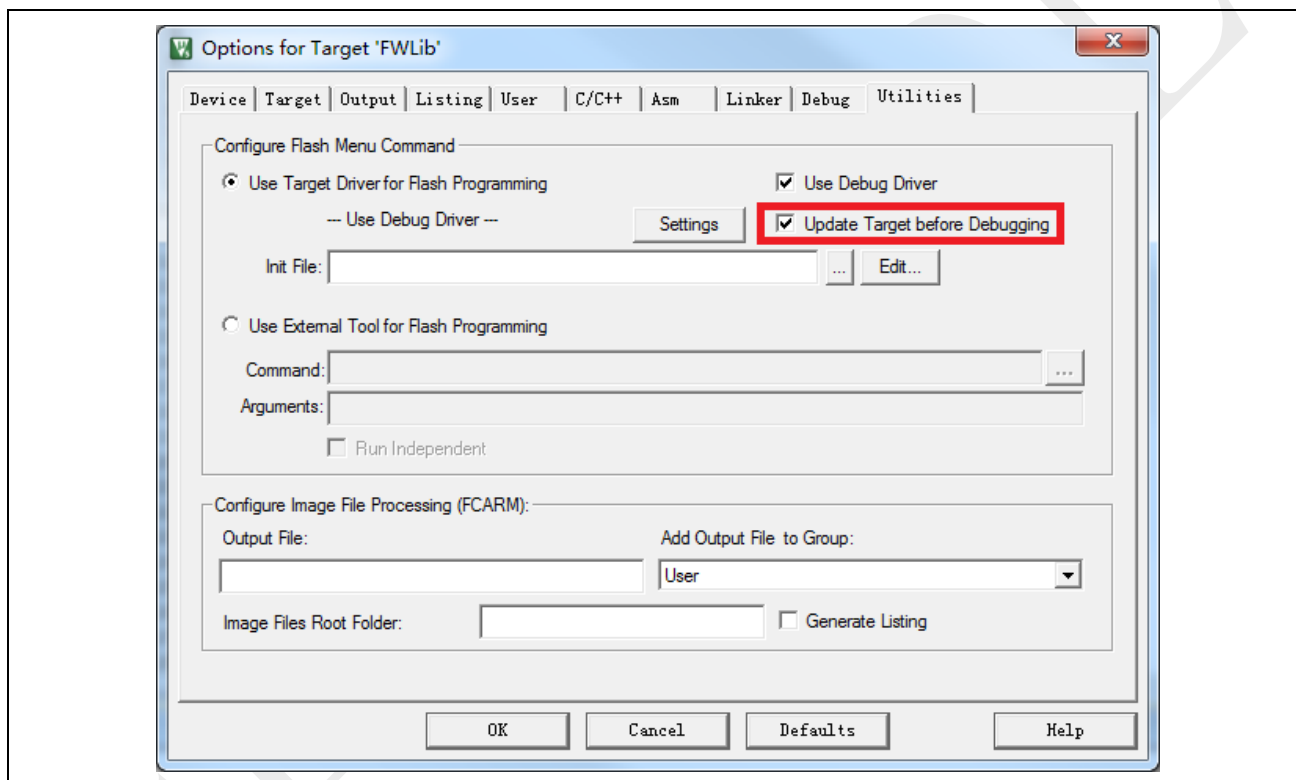

根据前面的介绍，将 J-LINK 设备与目标芯片正确连接后，按照图 3-2、图 3-3 以及图 3-4 设置 Debug 的相关选项，就可以使用 J-LINK 设备调试程序了。使用 J-LINK 调试程序时，必须保证 Flash 存储器中的程序与当前程序一致，这就需要用户每次修改代码后，都需要点击  按钮进行编译，随后点击  按钮将程序下载到 Flash 存储器中。值得一提的是，KEIL 软件提供了一个功能，可以自动上述动作，如图 4-1 所示。用户只需勾选 Update Target before Debugging 选项，那么在每次启动 Debug 会话时，KEIL 软件会自动通过 J-LINK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

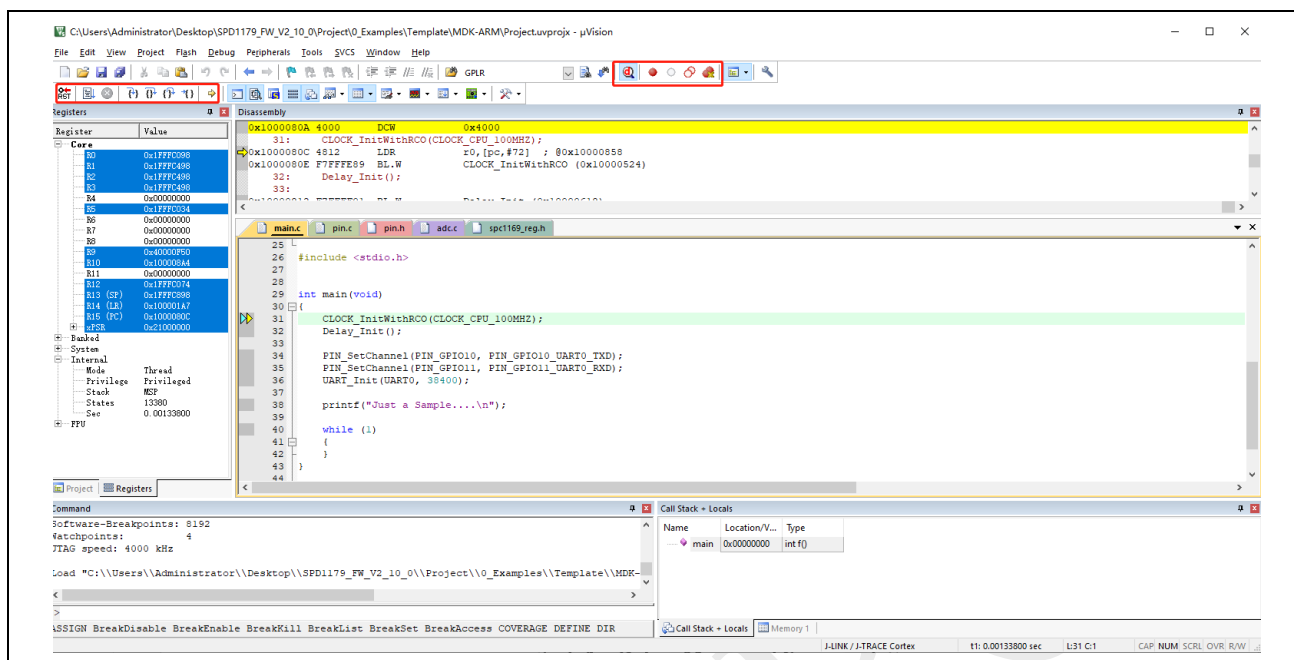




















表 4-1: Debug 菜单与指令

Debug 菜单	工具栏	快捷键	描述
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.

4.1 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。

4.2 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 4-3 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 4-4 所示。

图 4-3：设置断点

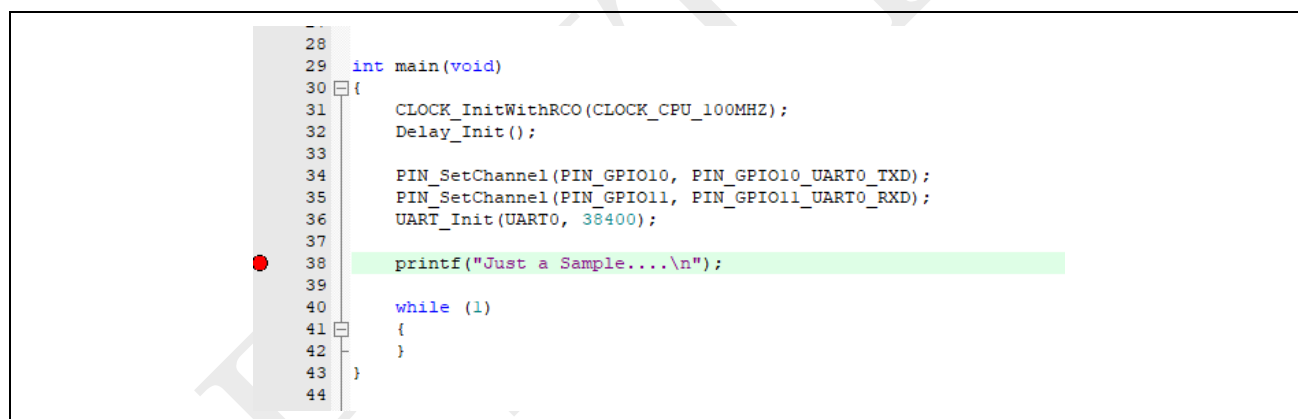
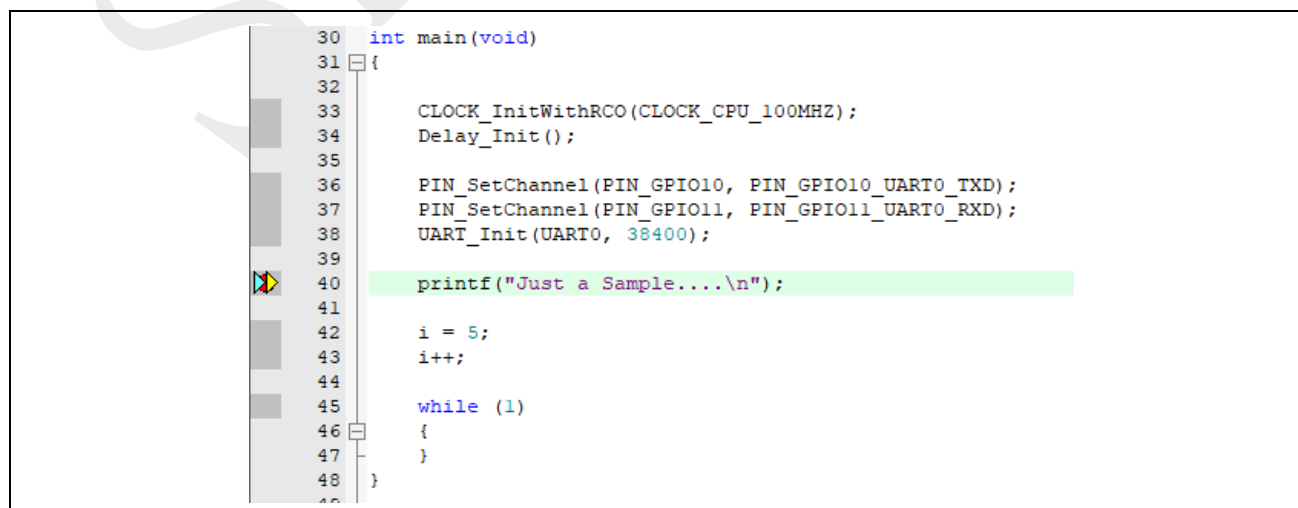


图 4-4：程序执行到断点



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。

4.3 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 4-5 所示。

从图 4-5 可以看出，我们将变量 i 添加到观察窗口 Watch1 里，结果如图 4-6 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 4-5：添加变量到观察窗口

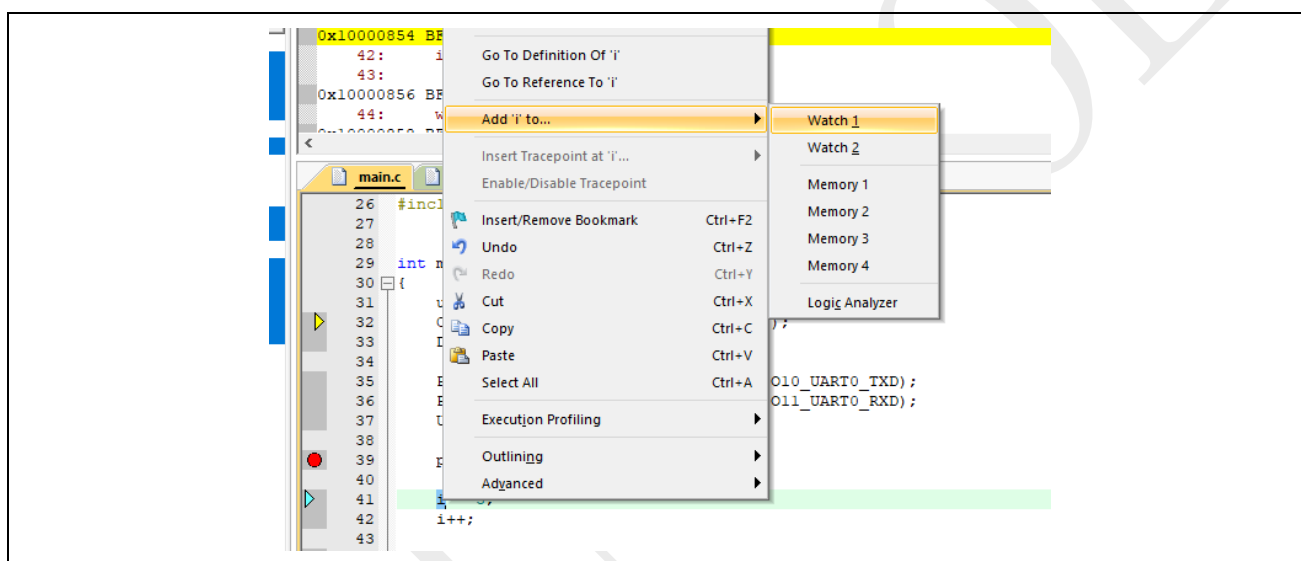
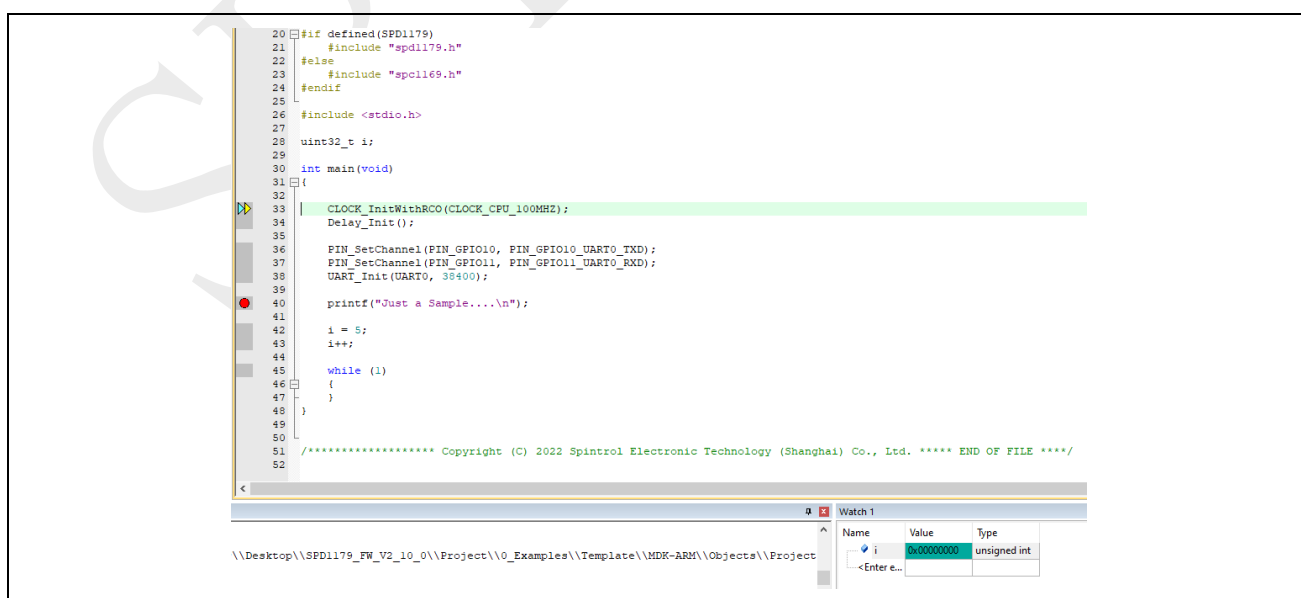


图 4-6：添加变量到观察窗口的结果



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 4-7 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 4-8 所示，可以看出变量 i 的值变为 6。

图 4-7: $i=5$ 执行结果

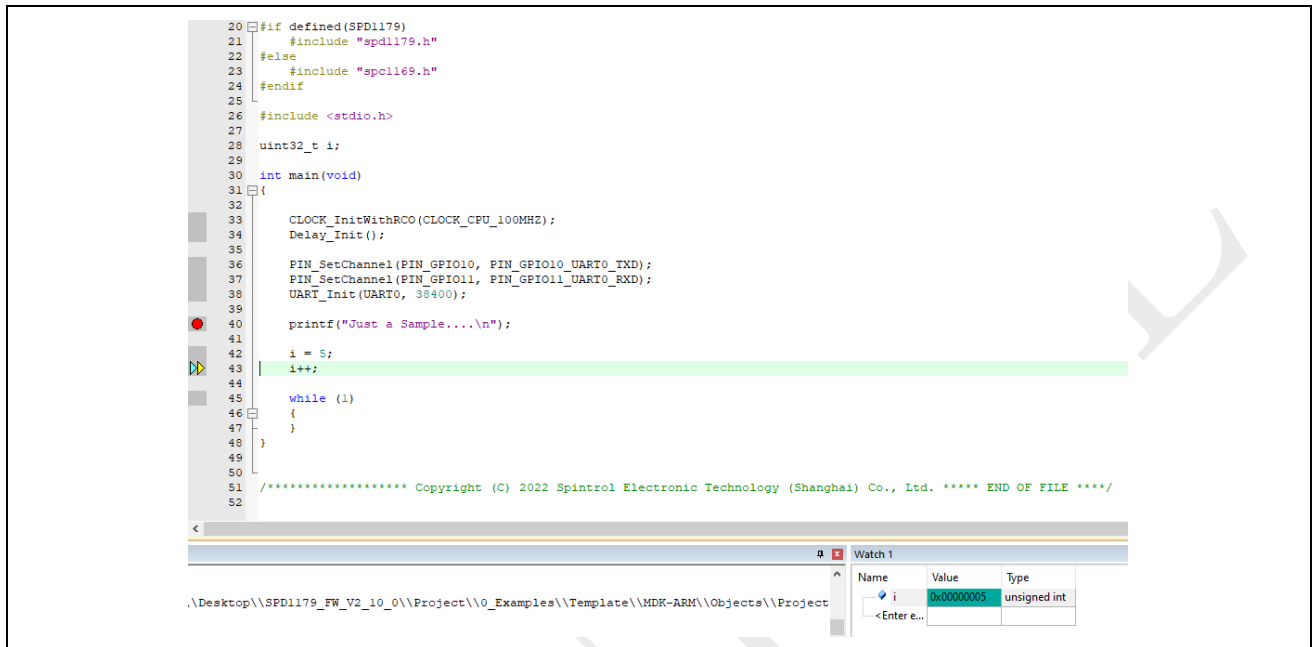
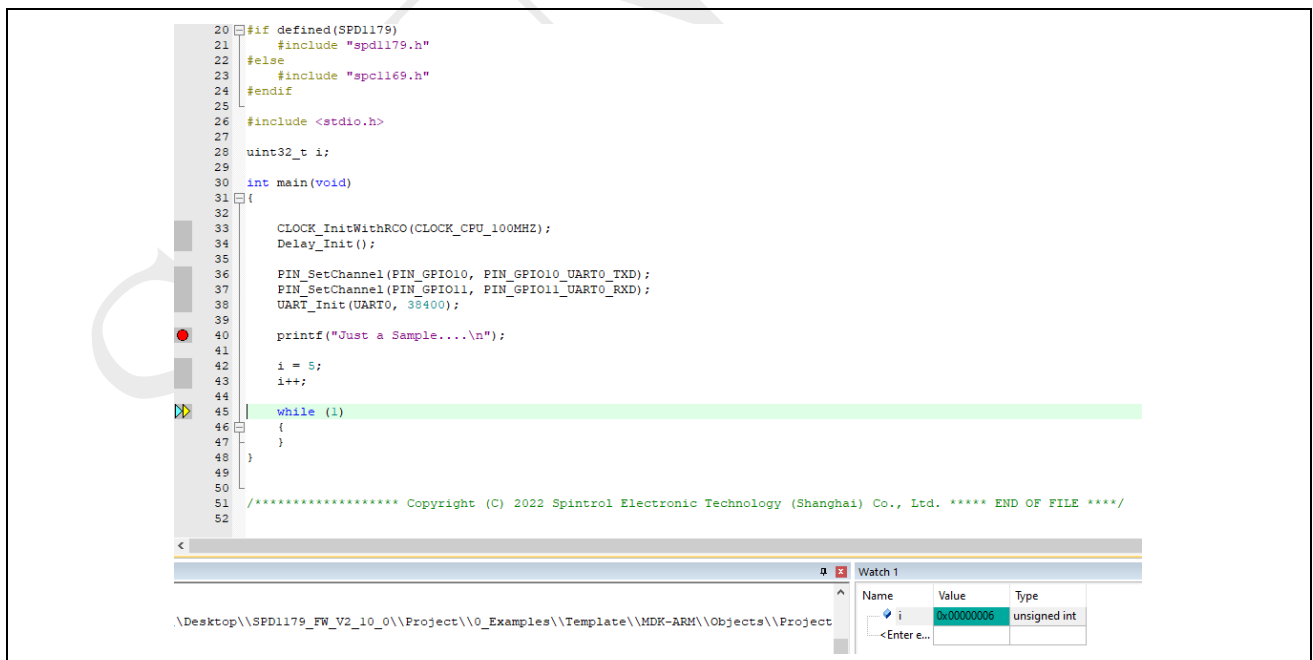





图 4-8: $i++$ 执行结果



4.4 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。本节以芯片外设模块 UART0 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 SFR 文件，位于目录 IDE_Support\MDK-ARM 中。设置结果如图 4-9 所示。
2. 单击按钮，进入 Debug 模式，将芯片外设 UART0 添加到 System Viewer 窗口，如图 4-10 所示。添加后的结果如图 4-11 所示。从图 4-11 可以看出，在 System Viewer 窗口中，不仅可以看到 UART0 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 UART0 添加到 System Viewer 窗口后，就可以在 Debug 程序的过程中观察到 UART0 各个寄存器的值。我们单步执行图 4-11 中 main 函数的程序，结果分别如图 4-12。

图 4-9: System Viewer File 设置界面

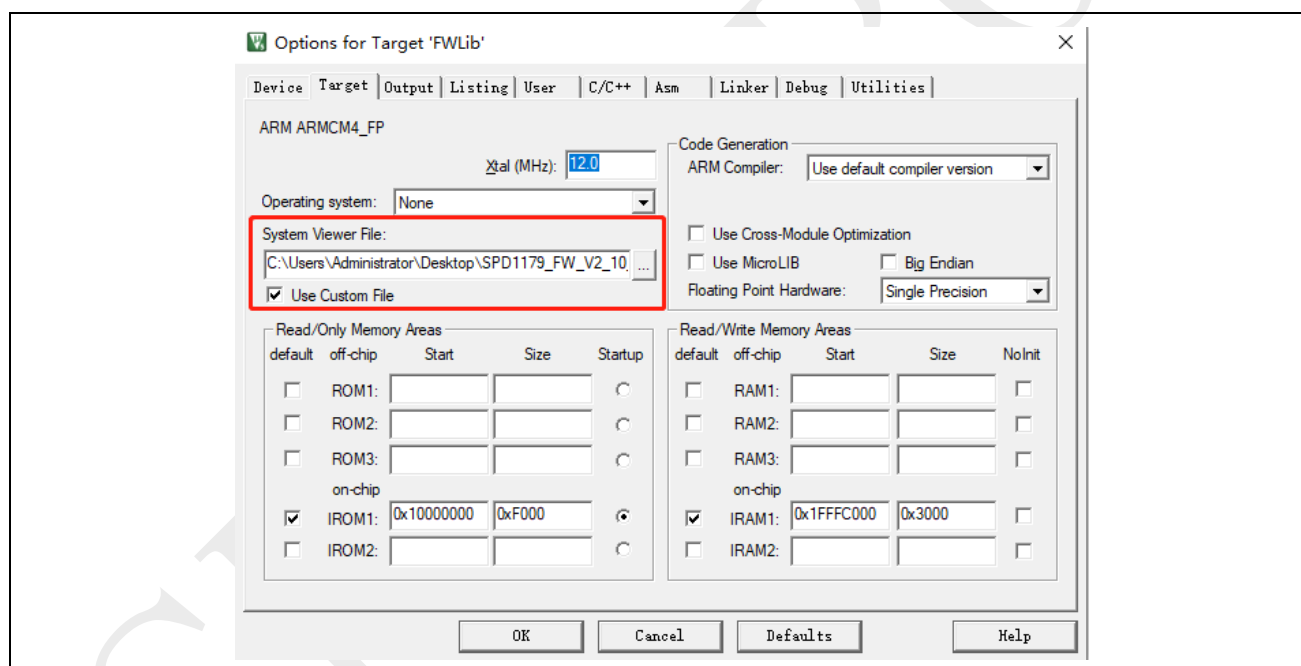


图 4-10: 添加 UART0 到 System Viewer 窗口

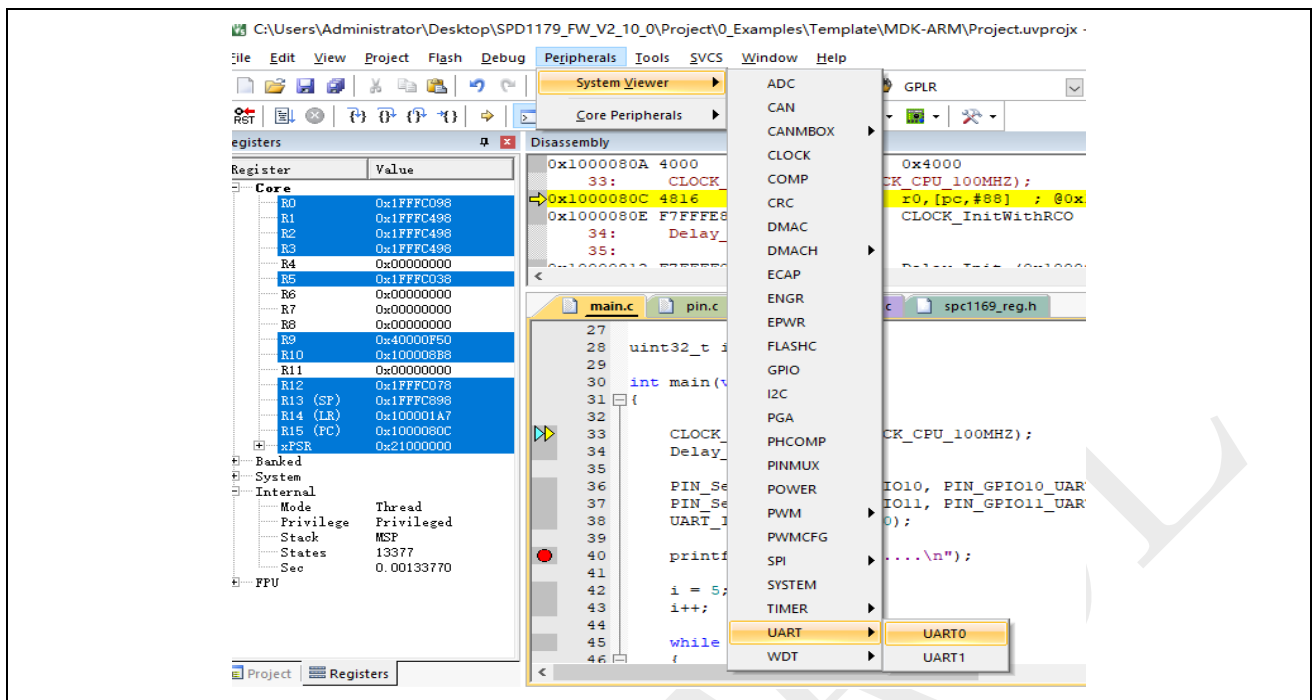


图 4-11: 添加 UART0 到 System Viewer 窗口的结果

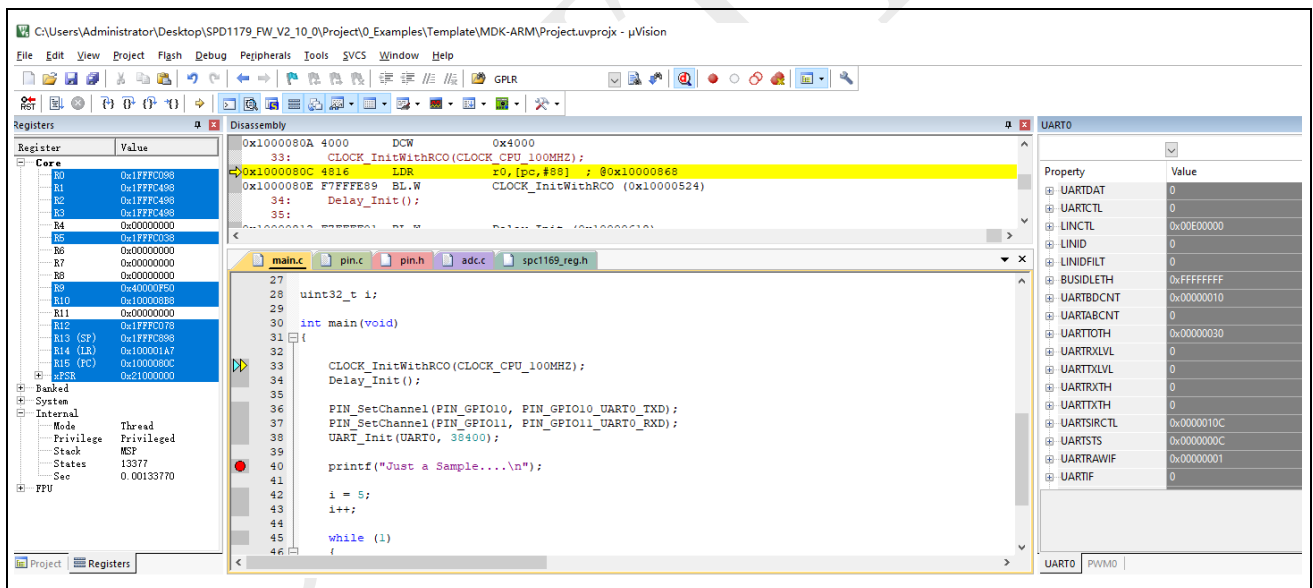
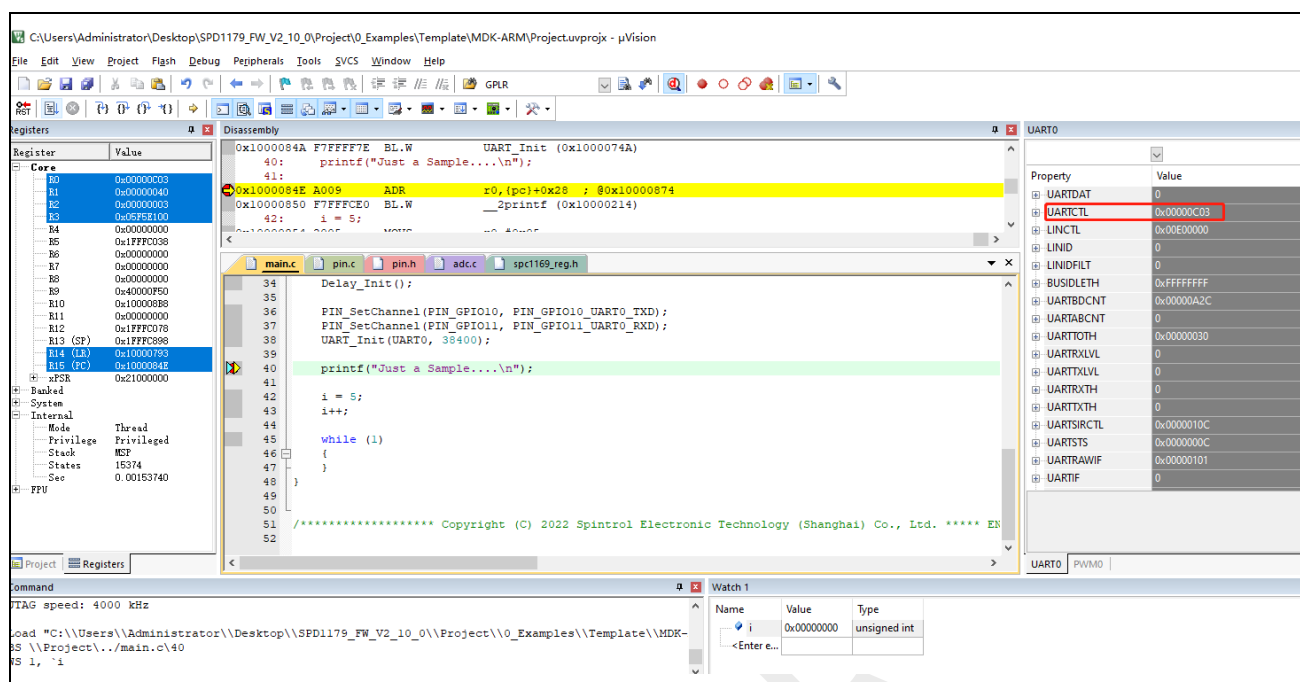


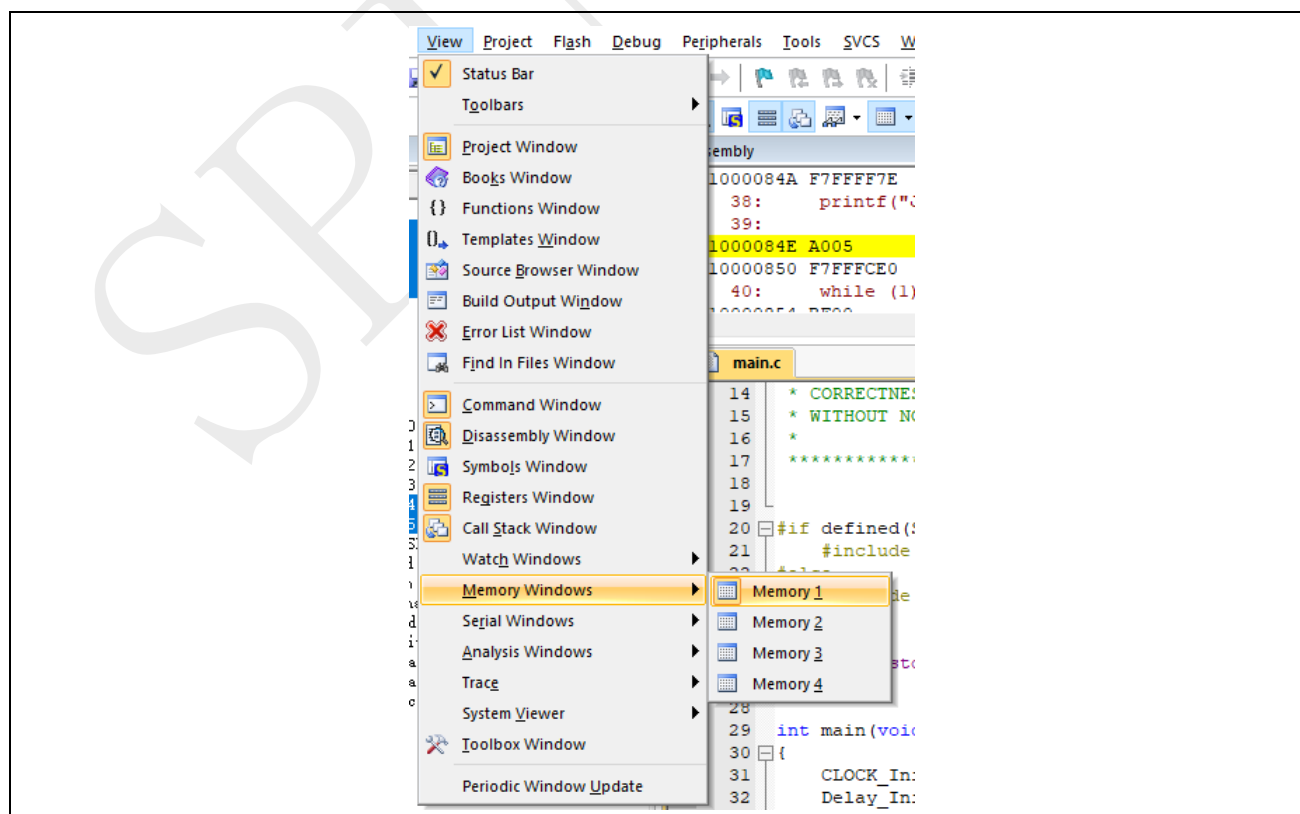
图 4-12: UART_Init 执行后结果



4.5 观察 Memory

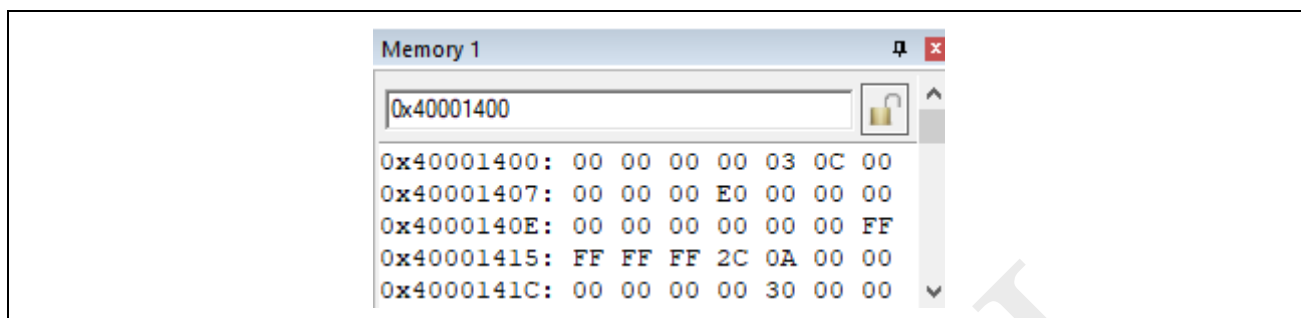
在调试程序的时候，也可以通过 Memory 查看外设 Register 的值，如图 4-13 所示。本节以芯片外设模块 UART0 为例介绍实现过程。

图 4-13: 开启 Memory1



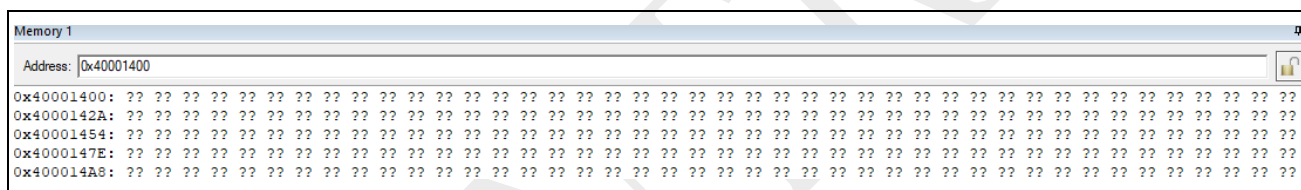
输入 0x40001400, 运行到 UART_Init 后, 执行结果如图 4-14 所示(与章节 4.4 完全相同)。

图 4-14: UART_Init 执行后结果



注意：Memory 窗口内地址范围不能超过 UART0 地址范围（0x40001400 - 0x40001458）。如图 4-15 所示，如果 Memory 窗口显示异常，可缩小 Memory 窗口到 UART0 地址范围（0x40001400 - 0x40001458）内解决。

图 4-15: UART_Init 执行后, 结果显示异常



5 代码运行到 RAM

详细信息请参考文档《RC-032_RAM 程序调试使用指南》。

SPIN TROL