AN13234

在 i.MX RT10xx 中使用 SWO TRACE 功能

版本 1 — 2022 年 7 月 6 日 应用笔记

目录

1 简介

i.MX RT10xx 系列是 NXP 采用先进 Cortex-M7 内核的跨界处理器，能够提供优秀的 CPU 性能和实时响应。Cortex-M7 内核的优势之一是它在 CoreSight 架构中包含了指令跟踪宏单元（ITM）。ITM 是一种全新的单片机调试跟踪方法，支持串行线输出跟踪（SWO trace）功能。

本文介绍了 SWO trace 的原理及优点，并且提供了在 i.MX RT10xx 系列 MCU 上使用 SWO trace 功能的详细指导，包括五款开发板上不同的硬件和软件配置。此外，本文还介绍了在不同的 IDE 开发环境下，使用 SWO trace 基本功能（ITM 调试输出）的具体步骤。

2 SWO trace 概述

2.1 CoreSight 组件

CoreSight 是 ARM 公司开发的用于 MCU 调试和跟踪的架构，该架构由众多组件组成。图 1 展示了常规的 CoreSight 架构模型，主要包括两个 ARM 内核处理器、一个 DSP 模块以及其他组件。图 1 的模型中共包含三个通路，分别是跟踪通路、调试通路和触发通路。本文涉及的内容主要与调试通路相关，如图 1 中的红色方框所示，包括 ITM 模块和 SWO 模块。

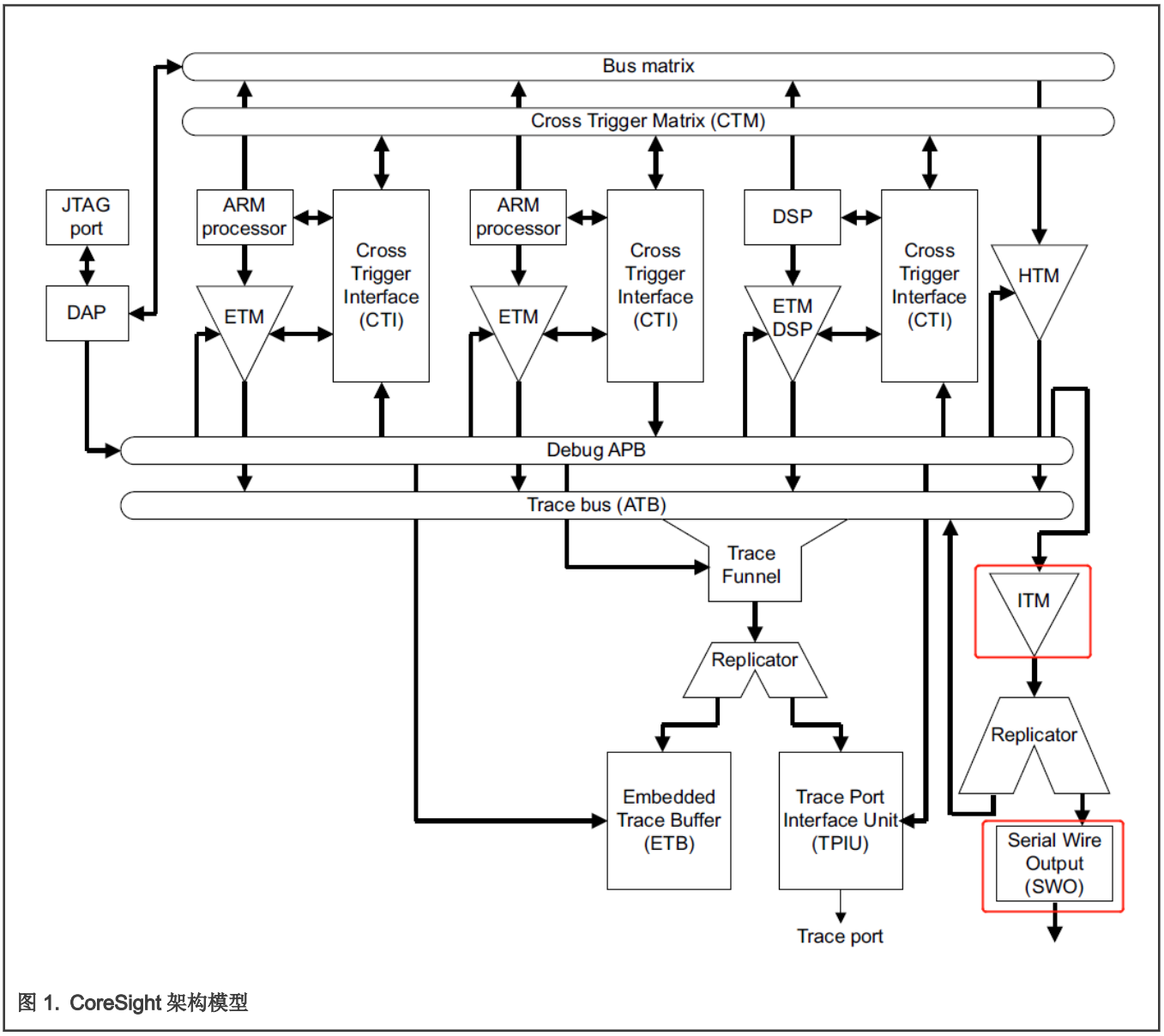

图 1. CoreSight 架构模型

2.2 串行线输出（SWO）

串行线输出（Serial Wire Output, SWO）是一个单引脚跟踪接口，它作为 Cortex-M 内核中 CoreSight 架构的一部分，充当片上跟踪数据与跟踪端口分析器（TPA）之间的桥梁。

SWO trace 支持对正在运行的 MCU 的内存的访问，而不需要暂停 CPU 的运行。一些采用 Cortex-M3/M4/M7 等架构的 MCU 内核支持 SWO Trace 功能，SWO trace 利用串行线调试（SWD）接口中的 SWO 引脚输出调试时产生的跟踪信息。SWD 调试接口是 ARM 公司提出的一种调试接口，相对于传统的 JTAG 接口，使用更少的信号引脚。图 2 是 SWO 的模型框图，除标准的 SWD 连接外，SWO trace 只需要一个额外的引脚。

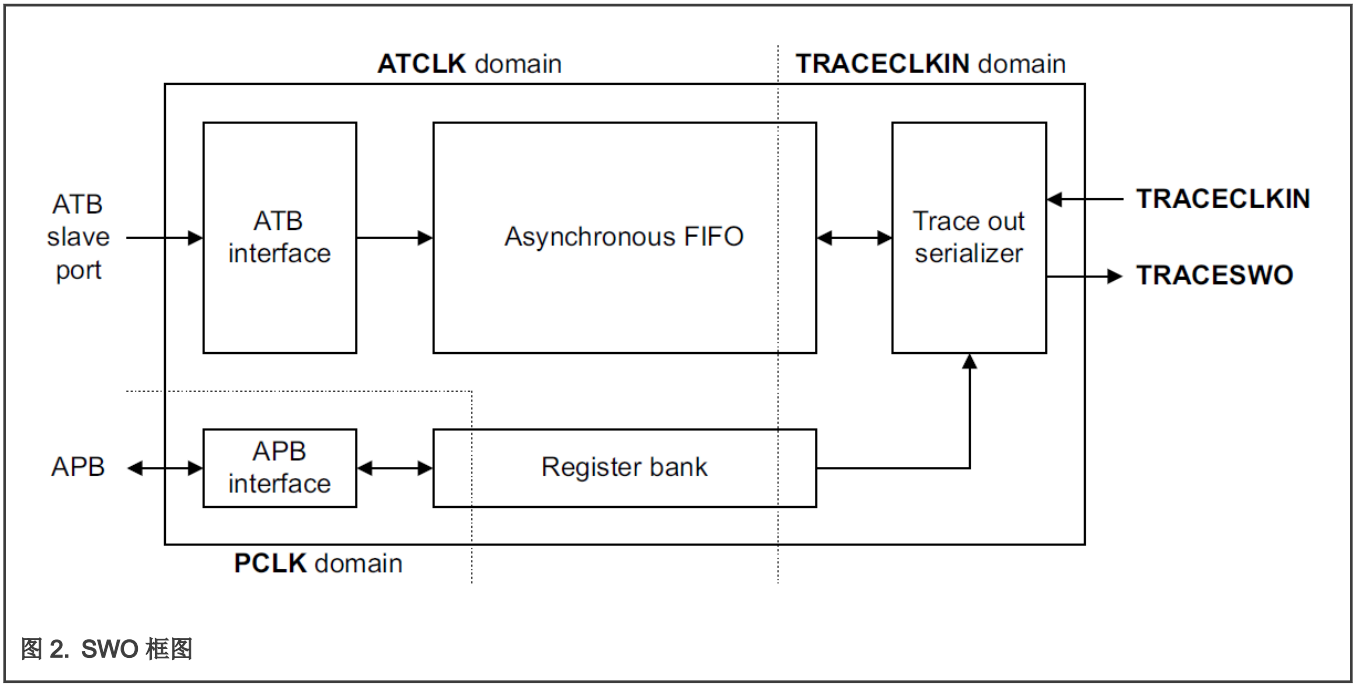

图 2. SWO 框图

SWO trace 接口与 TPIU（Trace Port Interface Unit, 跟踪端口接口单元）类似，但它仅支持 TPIU 接口的部分功能。

SWO trace 主要调试功能列举如下：

- 以字符串的形式发送调试消息
- 监控中断进入/退出
- 监控函数进入/退出
- PC（程序计数器）周期性抽样
- 事件通知
- 变量值随时间的变化

其中，SWO trace 最基本和常用的功能是输出调试信息。本文 IAR 中使用 SWO 功能和 Keil 中使用 SWO 功能将详细介绍该功能的使用方法。

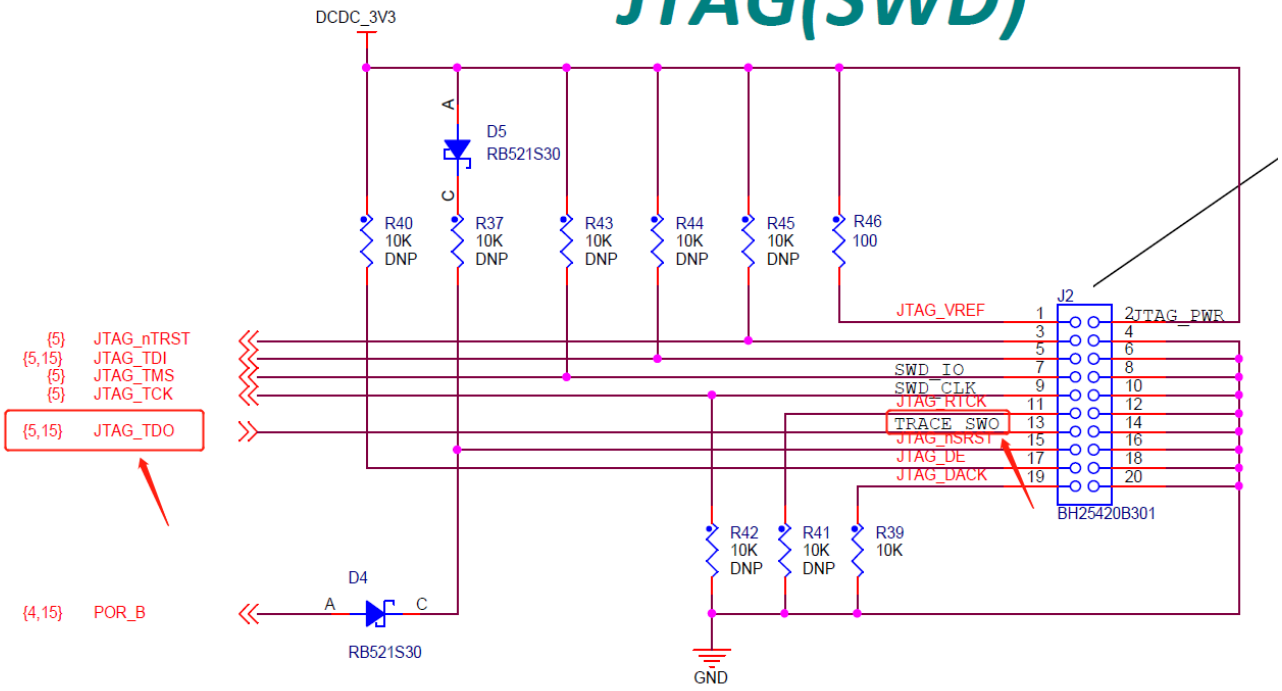

图 3. SWO 引脚

注意

如图 3 所示，SWO 引脚与 JTAG_TDO 引脚复用。因此串行线调试（SWD）模式和 JTAG 模式不能同时使用，本文讨论的 SWO trace 功能都基于串行线调试模式。

2.3 指令跟踪宏单元（ITM）

指令跟踪宏单元（Instrumentation Trace Macrocell, ITM）以包的形式生成跟踪信息，并且它提供了一种通过 SWO 端口将数据从终端发送到调试器的机制。ITM 的跟踪数据首先被传输到 SWO 接口，然后 SWO 将数据流传输到跟踪端口分析器。

ITM 包含 32 个激励端口，且允许不同的软件把数据输出到不同的端口，从而调试主机可以将各路数据分开。ITM 的一个主要用途，就是支持调试信息的输出（例如 printf 格式输出）。在 32 个激励端口中，端口 0 默认用于输出调试信息。不同于基于 UART 的数据输出，使用 ITM 输出不会对程序执行造成很大的延迟，因为在 ITM 内部有一个 FIFO，使得写入的输出数据得到缓冲。

ITM 主要用于：

- 支持 printf 风格的调试
- 跟踪操作系统和应用程序事件
- 发送系统诊断信息

2.4 调试器

i.MX RT10xx 系列 MCU 支持多种调试器，例如 CMSIS DAP, J-Link, PE micro，但是并不是每个调试器都支持 SWO trace 功能。本节将介绍两个常见的支持 SWO trace 功能的调试器。

- **LinkServer LPC-Link2**

LPC-Link2 是一款可扩展、独立运行的硬件调试器，可使用各种可下载的固件映像来支持各种开发工具和 IDE。LPC-Link2 支持部分 LPCXpresso 和 MIMXRT10xx 开发板的调试。

LinkServer 调试支持在 MCU 保持运行的过程中，启动或停止 SWO trace 功能。这意味着当使用 LPC-Link2 调试器进行 SWO trace 时，MCU 的性能不受影响。SWO trace 的丰富功能已经在 MCUXpresso IDE 和 LPC-Link2 固件上进行了充分的测试。如需详细使用教程，请参考 *MCUXpresso IDE SWO Trace Guide*。

- **SEGGER J-Link**

  J-Link 是一款支持多种 MCU 内核的调试器，当使用 J-Link 实现 SWO trace 功能时，如果需要启动或停止 SWO trace，CPU 必须暂时停止运行。

  为了正确计算内部采样率，必须对目标时钟速度进行正确采样。IAR 中使用 SWO 功能和 Keil 中使用 SWO 功能将详细介绍在不同 IDE 开发环境下，如何正确使用 J-Link 调试器实现 SWO trace 功能。

3 项目和硬件配置

SDK 中的官方项目示例以及部分型号的 EVK 开发板默认没有开启 SWO trace 功能。因此为了使能 SWO trace，需要对部分硬件电路和项目源文件进行修改。本章介绍了使能 SWO trace 功能所需的软硬件配置，包括以下五个型号的开发板：

- MIMXRT1010-EVK
- MIMXRT1020-EVK
- MIMXRT1050-EVK
- MIMXRT1060-EVK
- MIMXRT1060-EVKB

注意

源代码的修改示例基于最新的 SDK (版本 2.9.1) 。

3.1 MIMXRT1010-EVK

1. 配置 `pin_mux.c` 文件

   在项目文件夹中，选择 **board** 子文件夹，打开 `pin_mux.c` 文件，在函数 `BOARD_InitPins` 中添加以下内容：

```
IOMUXC_SetPinMux(                /* 添加以下代码*/
    IOMUXC_GPIO_AD_09_ARM_TRACE_SWO,
    0U);
IOMUXC_SetPinConfig(
    IOMUXC_GPIO_AD_09_ARM_TRACE_SWO,
    0x00F9U);
```

2. 配置 *clock_config.c* 文件

   在项目文件夹中，选择 **board** 子文件夹，打开 `clock_config.c` 文件，修改函数 `BOARD_BootClockRUN` 的部分内容，如下所示：

```
/* Disable TRACE clock gate. */            /* 定位到此处*/
CLOCK_DisableClock(kCLOCK_Trace);
/* Set TRACE_PODF. */
CLOCK_SetDiv(kCLOCK_TraceDiv, 3);
/* Set Trace clock source. */
CLOCK_SetMux(kCLOCK_TraceMux, 0);

CLOCK_EnableClock(kCLOCK_Trace);        /* 代码修改如下*/
/* Set TRACE_PODF. */
```

```
CLOCK_SetDiv(kCLOCK_TraceDiv, 0);
/* Set Trace clock source. */
CLOCK_SetMux(kCLOCK_TraceMux, 0);
```

3.2 MIMXRT1020-EVK

1. 连接 SWO 引脚和 JTAG-TDO 引脚。

   连接 SWO (J19-3 号引脚) 和 JTAG-TDO (J16-13 号引脚) 。

2. 移除 R116

   如图 4 所示，JTAG-TDO 引脚与 ENET-RESET 信号复用，其中电容 (C137) 对信号产生影响，因此移除 R116。

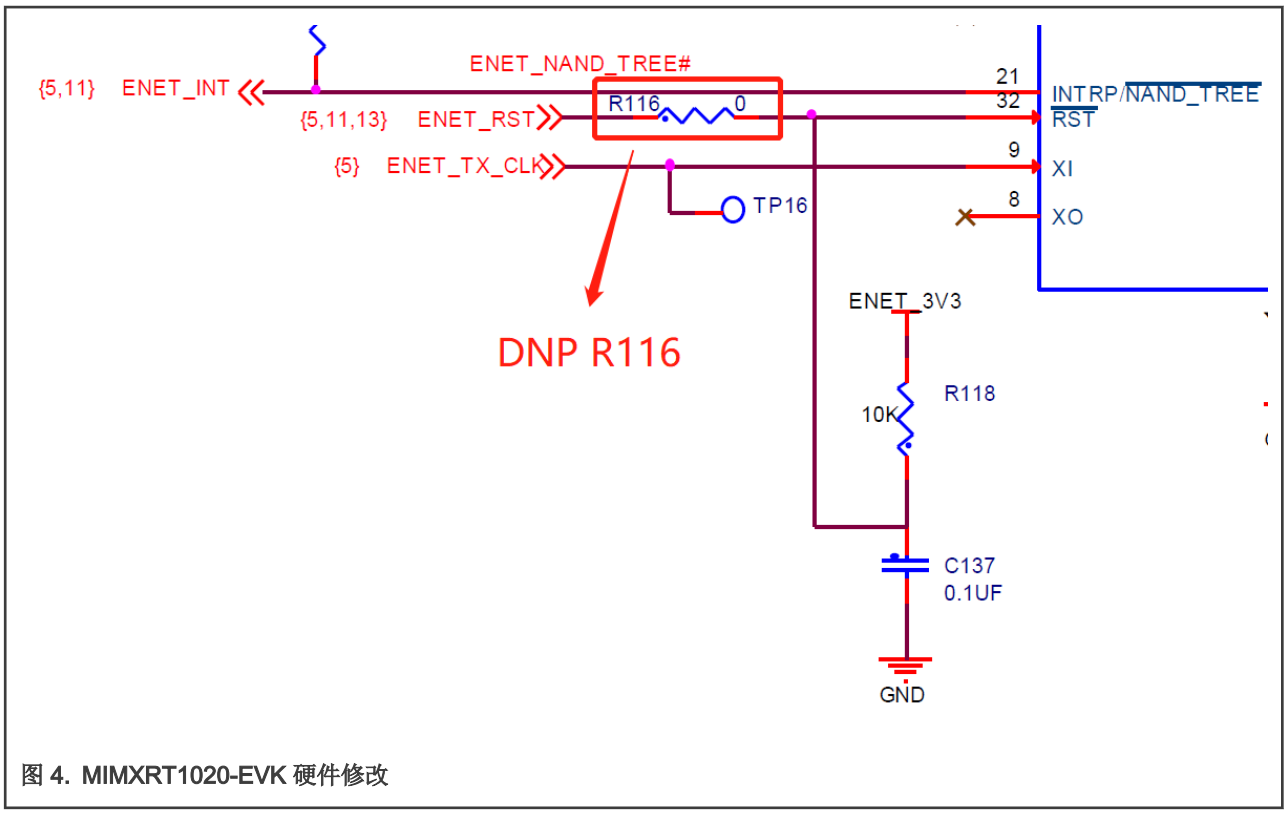

图 4. MIMXRT1020-EVK 硬件修改

3. 配置 `pin_mux.c` 文件

   在项目文件夹中，选择 **board** 子文件夹，打开 `pin_mux.c` 文件，在函数 `BOARD_InitPins` 中添加以下内容：

```
IOMUXC_SetPinMux(               /* 添加以下代码*/
    IOMUXC_GPIO_AD_B0_11_ARM_CM7_TRACE_SWO,
    0U);
IOMUXC_SetPinConfig(                  /* 添加以下代码*/
    IOMUXC_GPIO_AD_B0_11_ARM_CM7_TRACE_SWO,
    0x00F9U);
```

4. 配置 `clock_config.c` 文件

   与 MIMXRT1010-EVK 的时钟配置相同，详见 MIMXRT1010-EVK。

3.3 MIMXRT1050-EVK

1. 连接 R256 与 J21 的 13 号引脚。

2. 配置 pin_mux.c 文件。

   在项目文件夹中，选择 **board** 子文件夹，打开 pin_mux.c 文件，在函数 BOARD_InitPins 中添加以下内容：

```
IOMUXC_SetPinMux(          /* 添加以下代码*/
   IOMUXC_GPIO_B0_13_ARM_CM7_TRACE_SWO,
   0U);
IOMUXC_SetPinConfig(        /* 添加以下代码*/
   IOMUXC_GPIO_B0_13_ARM_CM7_TRACE_SWO,
   0x00F9U);
```

3. 配置 clock_config.c 文件。

   在项目文件夹中，选择 **board** 子文件夹，打开 clock_config.c 文件，修改函数 BOARD_BootClockRUN 的部分内容，如下所示：

```
/* Disable TRACE clock gate. */        /* 定位到此处*/
CLOCK_DisableClock(kCLOCK_Trace);
/* Set TRACE_PODF. */
CLOCK_SetDiv(kCLOCK_TraceDiv, 3);
/* Set Trace clock source. */
CLOCK_SetMux(kCLOCK_TraceMux, 0);

CLOCK_EnableClock(kCLOCK_Trace);    /* 代码修改如下*/
/* Set TRACE_PODF. */
CLOCK_SetDiv(kCLOCK_TraceDiv, 0);    /* 代码修改如下*/
/* Set Trace clock source. */
CLOCK_SetMux(kCLOCK_TraceMux, 3);    /* 代码修改如下*/
```

3.4 MIMXRT1060-EVK

1. 配置 pin_mux.c 文件。

   在项目文件夹中，选择 **board** 子文件夹，打开 pin_mux.c 文件，在函数 BOARD_InitPins 中添加以下内容：

```
IOMUXC_SetPinMux(IOMUXC_GPIO_AD_B0_10_ARM_TRACE_SWO, 0U);
IOMUXC_SetPinConfig(IOMUXC_GPIO_AD_B0_10_ARM_TRACE_SWO,0x00F9U);
```

2. 配置 clock_config.c 文件。

   与 MIMXRT1050-EVK 的时钟配置相同，详见 MIMXRT1050-EVK。

3.5 MIMXRT1060-EVKB

1. 移除 R173。
2. 配置 pin_mux.c 文件。

   与 MIMXRT1060-EVK 的引脚复用配置相同，详见 MIMXRT1060-EVK。

3. 配置 clock_config.c 文件。

   与 MIMXRT1060-EVK 的时钟配置相同，详见 MIMXRT1060-EVK。

注意

TRACE_ROOT_CLK 的频率应该与 CORE_CLK 的频率接近，请特别注意不同 MCU 之间的时钟配置差异。

4 IAR 中使用 SWO 功能

完成项目和硬件配置中列举的硬件和项目源文件的配置后，SWO trace 功能理论上已经可以使用。然而在不同的开发环境（IDE）中，为了支持 SWO trace 功能，仍需要进行相应的配置。本章主要介绍如何在 IAR 中使用 J-Link 调试器测试 SWO trace 的基本功能（ITM 输出调试信息）。配置步骤如下：

1. 准备工作：
   - 软件：MIMXRT1060-EVK SDK release (version: 2.9.1)
   - 硬件：MIMXRT1060-EVK 开发板
   - 调试器：J-Link Plus
   - IDE: IAR Embedded Workbench for Arm 8.50.9
   - ITM 输出调试信息的代码：

```
while (1)
{
   ITM_SendChar('A');          /* ITM 向指定端口发送字母 A */
   ITM_SendChar('B');
   ITM_SendChar('C');
}
```

   ITM 输出信息的默认端口是端口 0，关于更多 ITM 输出功能的相关信息，请参考 `core_cm7.h` 文件中的 `ITM_SendChar()` 函数。

2. 点击 **Project -> Options -> General Options -> Library Configuration**， 将 **library** 设置为 **Full**，并在 **stdout/stderr** 中选择 **Via SWO**， 如图 5 所示。

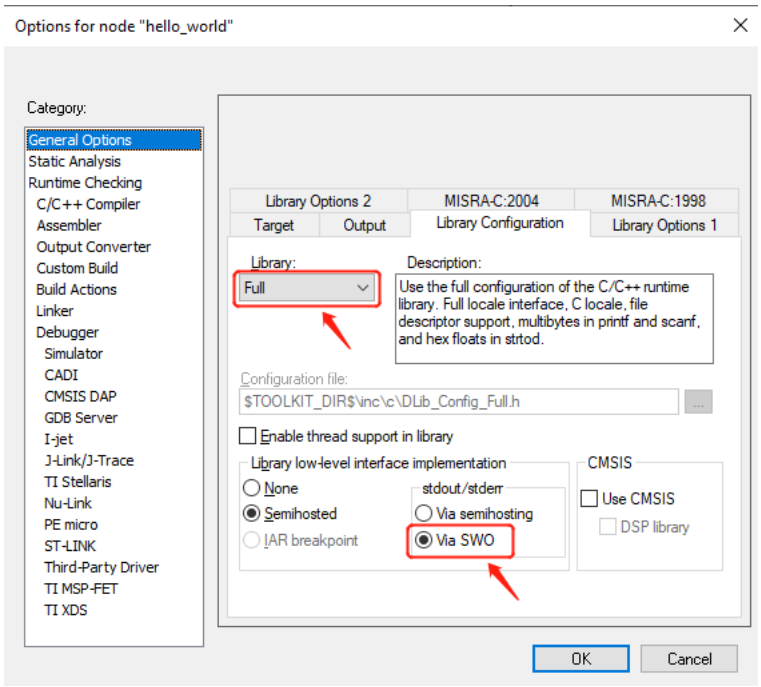

图 5. IAR 库配置

3. 点击 **Project -> Options -> Debugger**。设置调试器为 **J-Link/J-Trace**，如图 6 所示。

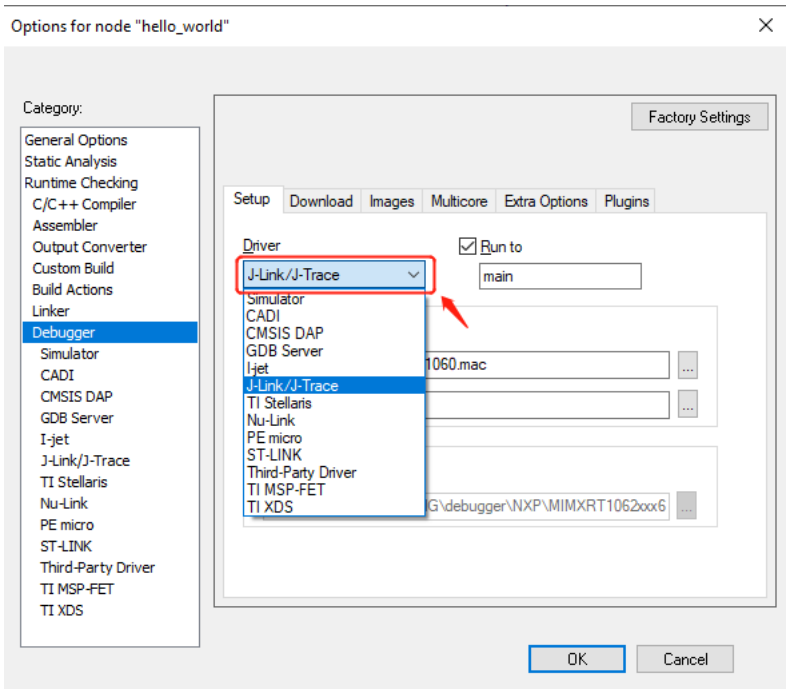

图 6. 将 debugger 设置为 J-Link

4. 点击 **Project -> Options -> Debugger -> J-Link/J-Trace**。在 **Setup** 菜单中设置复位方式为 **Normal**，CPU 时钟为 **600 MHz**。在 **Connection** 菜单中设置端口模式为 **SWD**， 如图 7 所示。

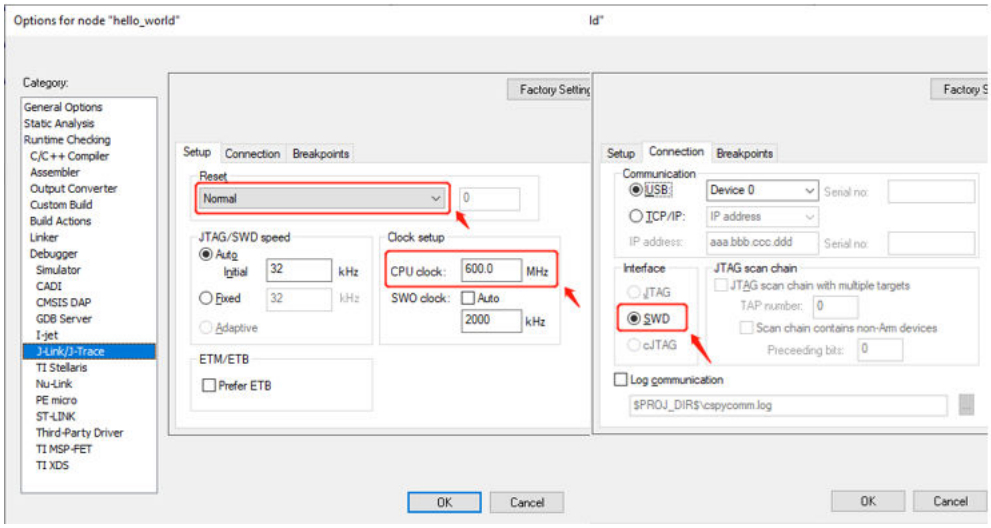

图 7. J-Link 调试器设置

5. 下载并调试程序，点击 **File Menu -> J-Link -> SWO Configuration**，进行如图 8 和图 9 所示的配置。

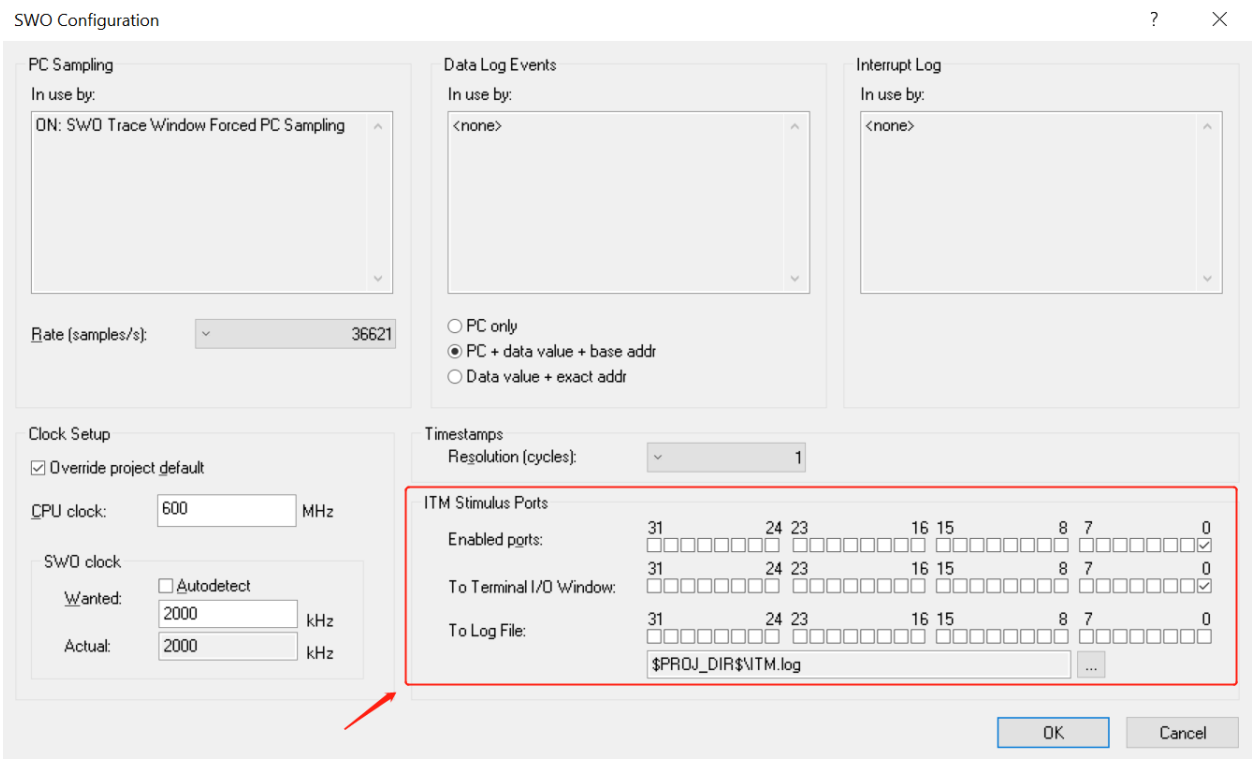

图 8. 设置 ITM 激励端口为 port0

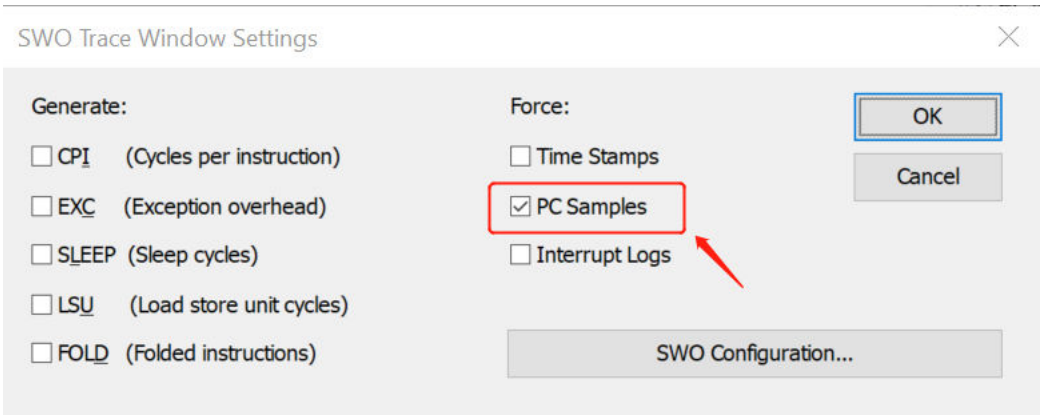

图 9. 将 SWO trace 模式设置为 PC samples

6. 点击 **File Menu -> J-Link -> SWO trace**，打开 SWO Trace 面板。然后点击  按钮以启用 SWO trace。
7. 点击 **View -> Terminal I/O** 打开 Terminal I/O 窗口，然后全速运行程序。ITM 输出信息如图 10 所示。

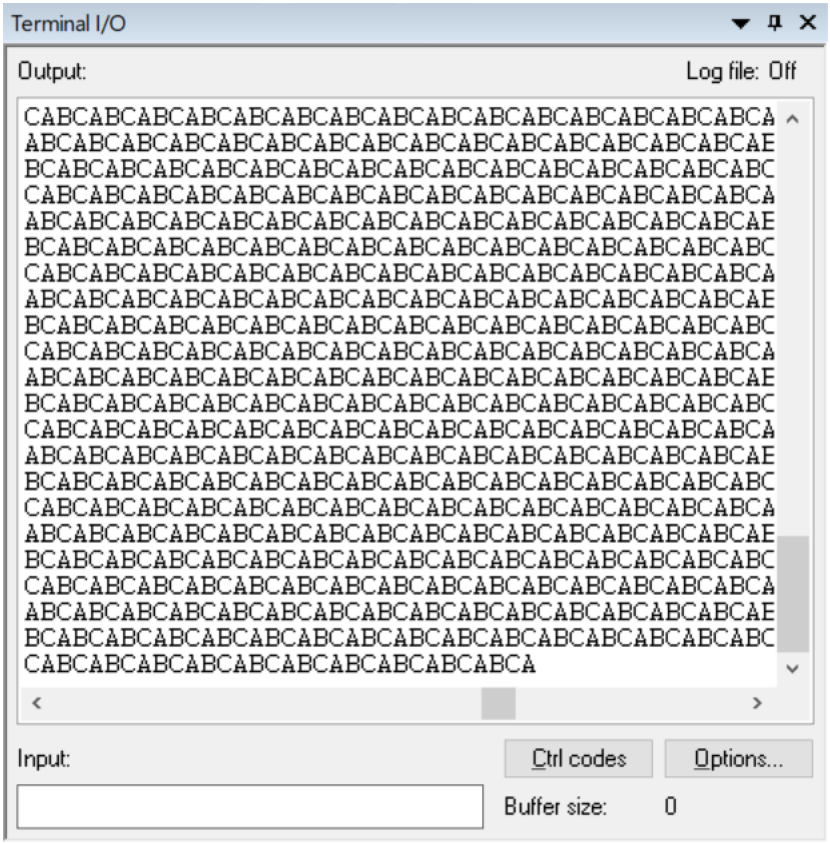

图 10. Terminal I/O 窗口输出

5 Keil 中使用 SWO 功能

本章介绍了使用 J-Link 调试器在 Keil 上实现 SWO trace 功能的步骤。配置步骤如下：

1. 准备工作
   - 软件：MIMXRT1060-EVK SDK release (version: 2.9.1)
   - 硬件：MIMXRT1060-EVK 开发板
   - 调试器：J-Link Plus
   - IDE: Keil uVision 5
   - ITM 输出调试信息的代码：

```
while (1)
{
  ITM_SendChar('A');     /* ITM 向指定端口发送字母 A*/
  ITM_SendChar('B');
  ITM_SendChar('C');
}
```

   ITM 输出信息的默认端口是端口 0，关于更多 ITM 输出功能的相关信息，请参考 `core_cm7.h` 文件中的 `ITM_SendChar()` 函数。

2. 点击 **Options for Target** 按钮，然后点击 **Target** 子菜单，并勾选 **Use MicroLIB**，如图 11 所示。

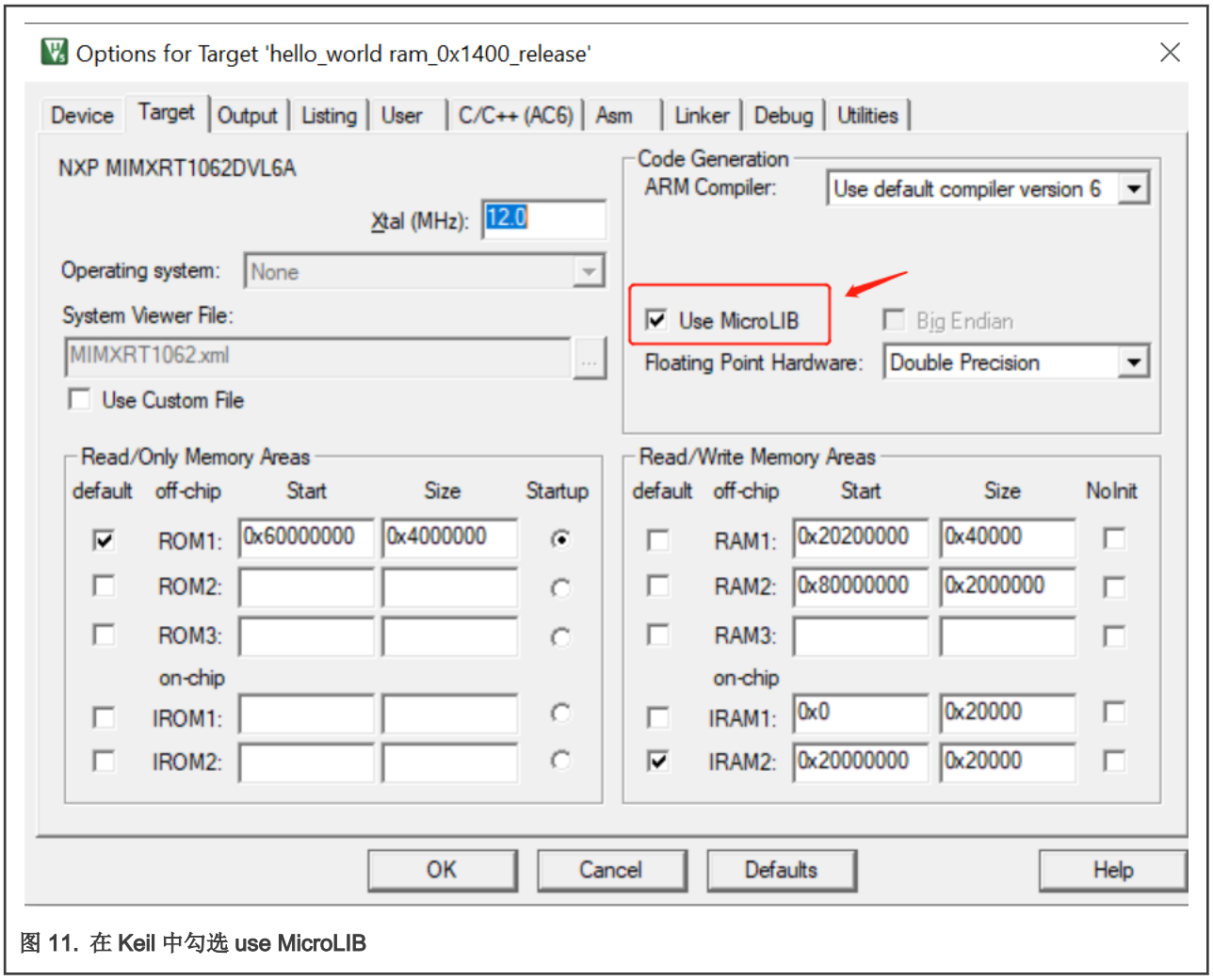

图 11. 在 Keil 中勾选 use MicroLIB

3. 在 **Debug** 菜单中，将调试器设置为 **J-Link/J-TRACE Cortex**，如图 12 所示。

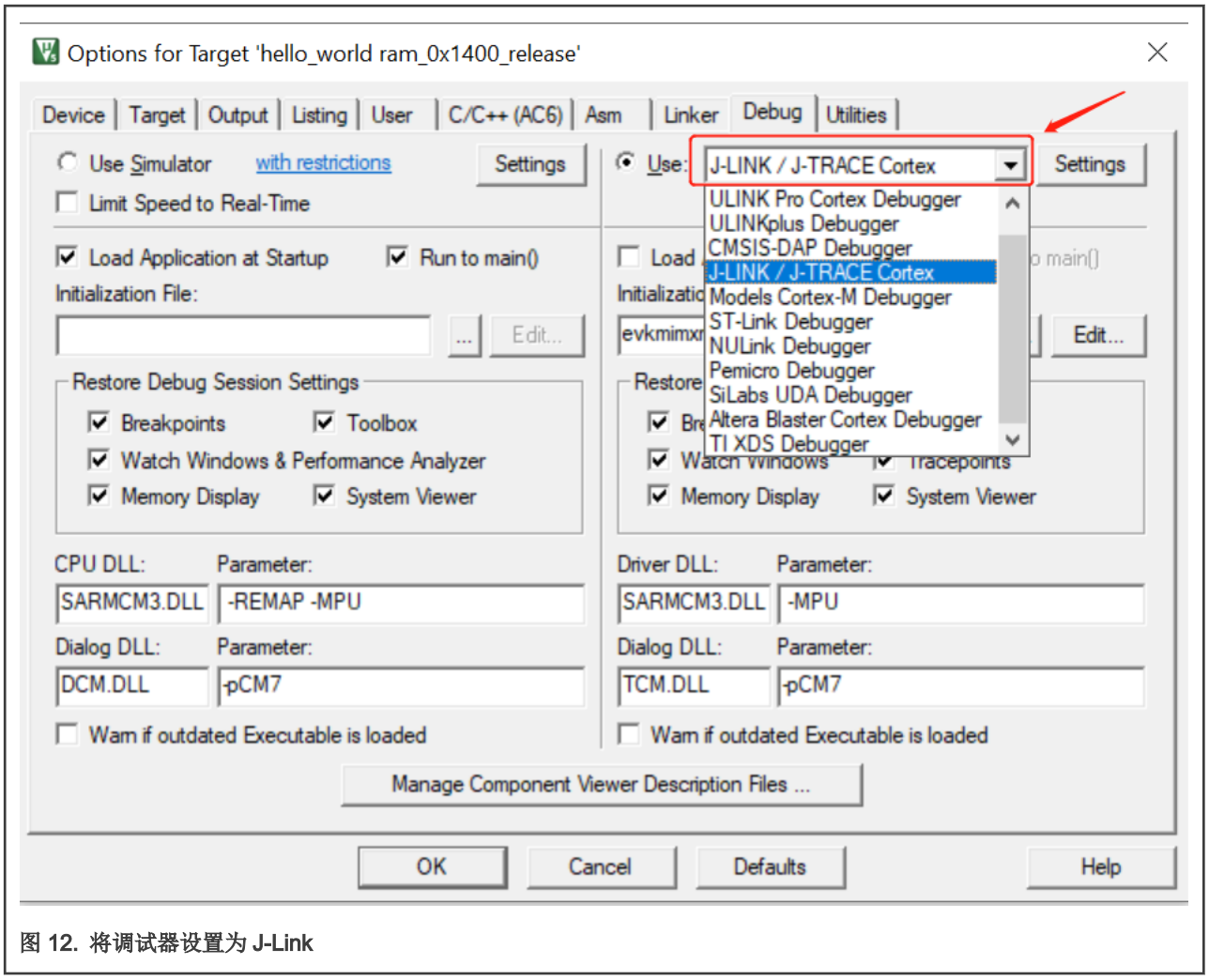

图 12. 将调试器设置为 J-Link

4. 点击上一步中的 **Settings** 按钮。在 J-Link 设置菜单中，设置端口为 **SW**，复位方式为 **Normal**，如图 13 所示。

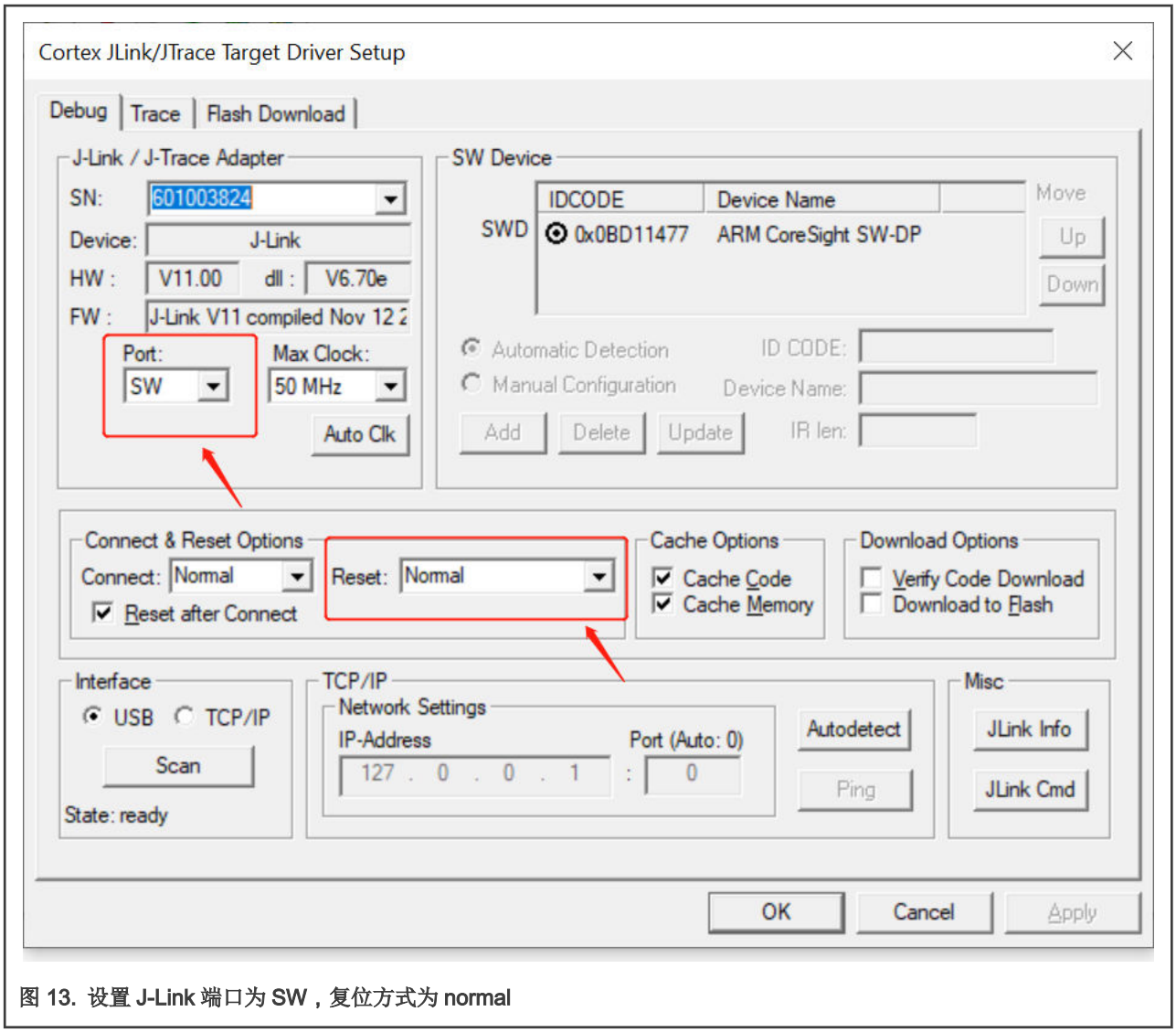

图 13. 设置 J-Link 端口为 SW，复位方式为 normal

5. 单击 **Trace** 菜单，进行如图 14 所示。

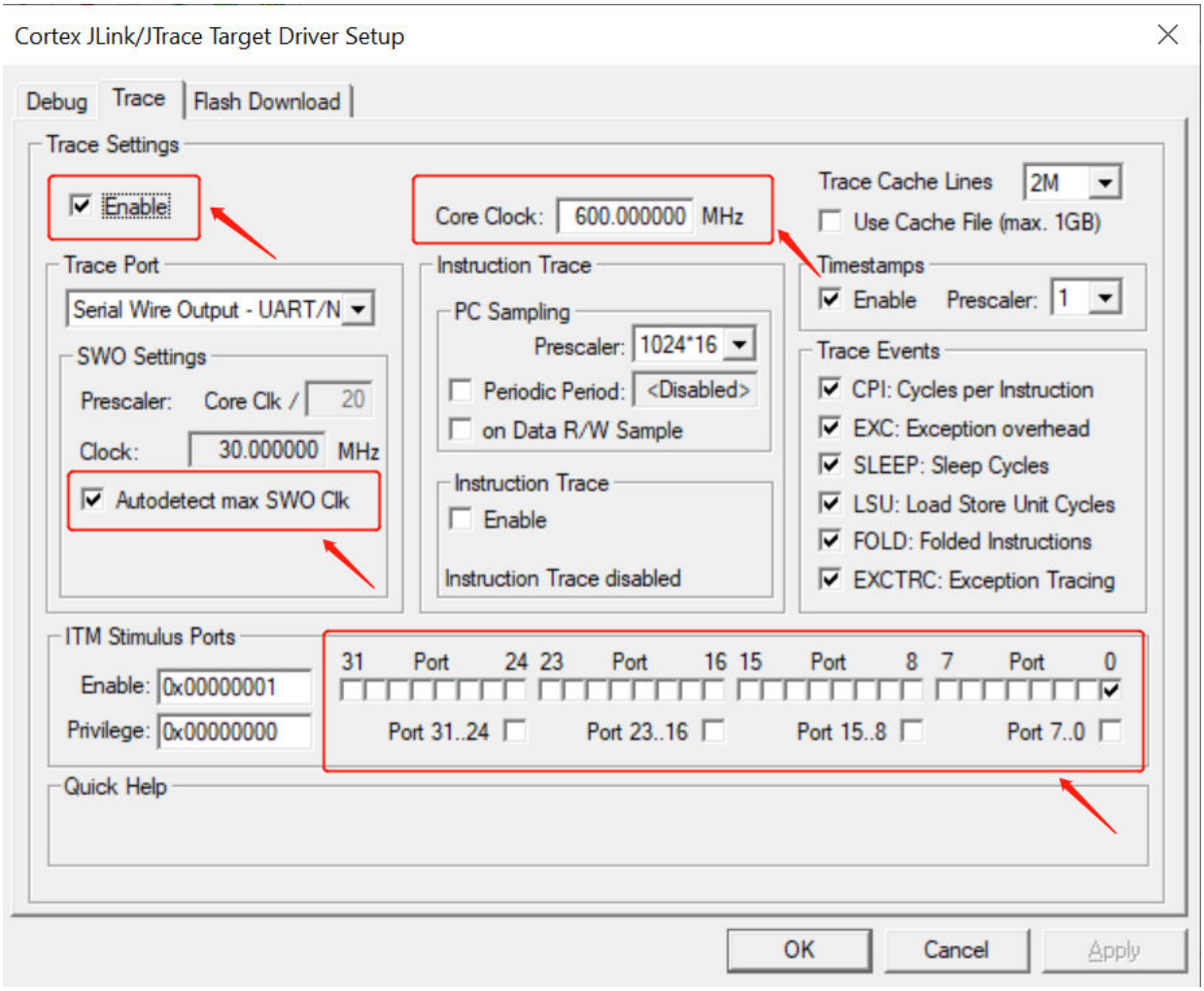

图 14. J-Link 跟踪配置

6. 下载并调试程序，点击 **View -> Serial Window -> Debug(printf) Viewer**。然后全速运行程序，ITM 输出信息如图 15 所示。

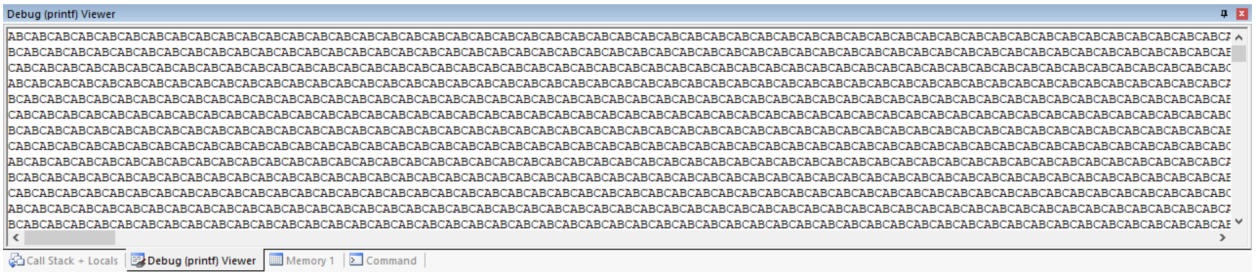

图 15. Keil 中 Debug(printf) Viewer 输出

6 本文总结

嵌入式软件开发中的一个基本需求是能通过终端输出调试信息，一般可通过两种方式实现：

- 一是使用串口线连接板上的 UART 接口和 PC 上的 COM 端口，然后通过 PC 上的终端来查看调试信息；
- 一是采用半主机机制，但有可能不被所用的工具链支持。

Cortex-M7 内核中的 CoreSight 架构突破了这种限制。Cortex-M7 内核提供了 ITM 接口，可调试由 SWO 引脚接收到的 ITM 数据。通过这种方式，可以在不配置串口和使用终端调试软件的情况下输出调试信息，在一定程度上减少了工作量。与普通串口打印相比，SWO 具有以下优势。

- 串口是 MCU 的片内外设，占用一个外设资源，而 SWO 不占用外设。
- 当使用不同的 MCU 时，需要重新编写串口的驱动，而 SWO 驱动程序可以兼容所有支持 Cortex-M 内核的 MCU。
- 串口输出一般使用中断方式发送，调试时可能需要在中断中观察某些信息，这样中断嵌套就容易出现问题，而 SWO 没有中断问题。
- SWO 比串行端口速度快，在时序要求较高的情况下对代码的影响较小，能够帮助提高调试效率。

除了最常用的调试信息输出功能外，SWO trace 还有许多强大的功能。例如，显示实时变量值的变化、监控代码执行时间、监控中断活动等。

7 参考文献

- *i.MX RT1010 Processor Reference Manual* (document IMXRT010RM)
- *i.MX RT1020 Processor Reference Manual* (document IMXRT1020RM)
- *i.MX RT1050 Processor Reference Manual* (document IMXRT1050RM)
- *i.MX RT1060 Processor Reference Manual* (document IMXRT1060RM)
- *Using the MIMXRT1060/4-EVK with MCUXpresso IDE*
- *MCUXpresso IDE SWO Trace Guide* (document MCUXPRESSO-SWO-TRACE)
- CoreSight Components Technical Reference Manual

8 修订记录

版本号	日期	主要更新
1	2022 年 7 月 6 日	更新 MIMXRT1060-EVK
0	2021 年 4 月	首次发布

This object is not available in the repository.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 2022 年 7 月 6 日
Document identifier: AN13234