

1 介绍

1.1 概述

RT600 是面向嵌入式应用的双核微控制器系列，配有 Arm® Cortex®-M33 CPU 和 Cadence Xtensa HiFi4 高级音频数字信号处理器 CPU。Cortex-M33 包括两个硬件协处理器，能够为一揽子复杂算法提供增强的性能。Arm Cortex-M33 是基于 Armv8-M 架构的新一代内核，它提供了系统增强功能，如 Arm TrustZone® 安全、单周期数字信号处理，以及紧耦合的协处理器接口，同时结合了低功耗、增强型调试功能和更高水平的模块集成的支持。

Cadence Xtensa HiFi4 音频 DSP 引擎是一款高度优化的音频处理器，专为高效执行音频和语音编解码器，以及预处理和后处理模块而设计。

RT600 提供高达 4.5 MB 的片上 SRAM（另外还有 128 Kb 紧耦合的 HiFi4 ram）以及多个高带宽接口来访问片外闪存。

RT600 的设计允许 Cortex-M33 以高达 300 MHz 的频率运行，而 HiFi4 DSP 以高达 600 MHz 的频率运行。

1.2 双核的基本机制

Cortex-M33（处理器 A）出厂时设置为正常工作。另一方面，为了最大限度地降低功耗，RT600 启动时，HiFi4 DSP（处理器 B）不会上电。要运行或调试 DSP 应用，需要在 Cortex-M33（处理器 A）上执行代码来初始化 HiFi4 DSP（处理器 B）。

在双核运行模式下，Cortex-M33 和 DSP 需要相互通信。RT600 提供了两种简单的方式，即消息单元（MU）和信号量块（Semaphore Block），来实现这个任务。

• MU 模块

它使得 SoC 内的两个处理器能够通过 MU 接口传递消息（例如数据、状态和控制），进行通信和协作。

MU 还可以让一个处理器使用中断向另一个处理器发出信号。

• 信号量块

— 信号量块提供了在多核系统中实现信号量所需的硬件支持。

— 信号量块还提供了一种简单的机制，通过单次写入访问来实现**锁定/解锁**操作。

目录

1 介绍	1
1.1 概述	1
1.2 双核的基本机制	1
1.3 相关的系统资源	1
2 消息单元（MU）	2
2.1 特性	3
2.2 基本配置	3
2.3 中断模式下的消息传输协议	4
2.4 轮询模式下的消息传输协议	5
3 信号量模式	6
3.1 特性	6
3.2 基本配置	6
3.3 功能说明	6
4 软件实现	7
4.1 引脚设置	7
4.2 中断模式下的消息传输的实现	7
4.3 轮询模式下的消息传输的实现	10
4.4 信号量块的实现	13
5 调试双核项目	15
5.1 调试系统	15
5.2 基本配置	15
5.3 双核项目调试	16
5.4 调试和运行双核应用	17



注意

本应用笔记中有关处理器 A 的指代对应于 Cortex-M33，而有关处理器 B 的指代对应于 HiFi4 DSP。

1.3 相关的系统资源

为了在双核模式下使用共享内存和外设时获得更好的性能，采用了以下的机制。

1.3.1 共享系统 SRAM

整个系统的 SRAM 空间多达 4.5 MB，被划分为多达 30 个单独的分区，两个 CPU、两个 DMA 引擎和所有其他 AHB 总线主机都可以访问这些分区。HiFi4 CPU 通过专用的 256 位接口访问 RAM。所有其他主机，包括 Cortex-M33 处理器和 DMA 引擎，可通过主 32 位 AHB 总线访问 RAM。硬件接口模块在 HiFi4 和 AHB 总线之间对每个 RAM 分区进行访问仲裁。

1.3.2 AHB 多层矩阵

一个多层 AHB 矩阵采用灵活的方式将 CPU 总线及其他总线主机连接到外设，允许不同的总线主机同时访问矩阵中的不同从机端口上的外设，从而优化了性能。图 1 显示了多层 AHB 矩阵的框图。

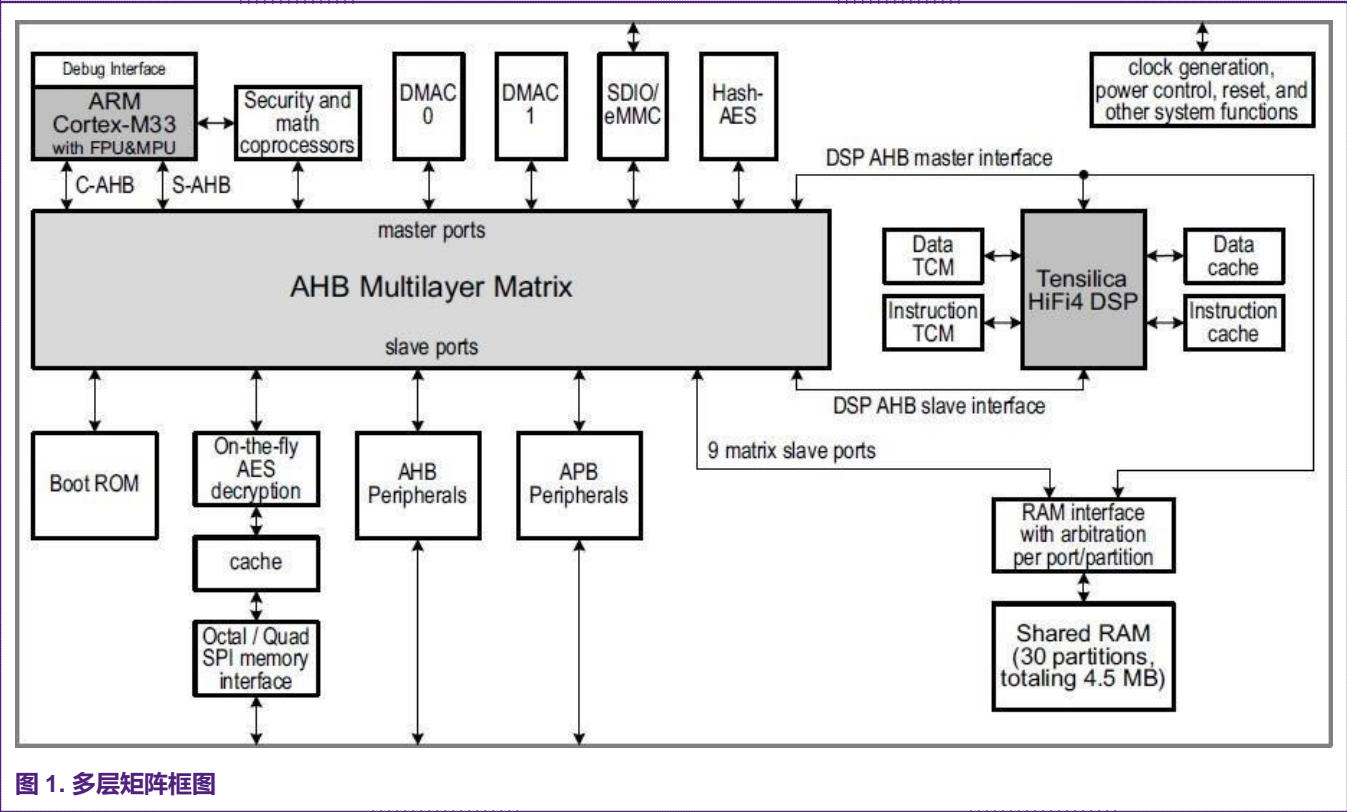
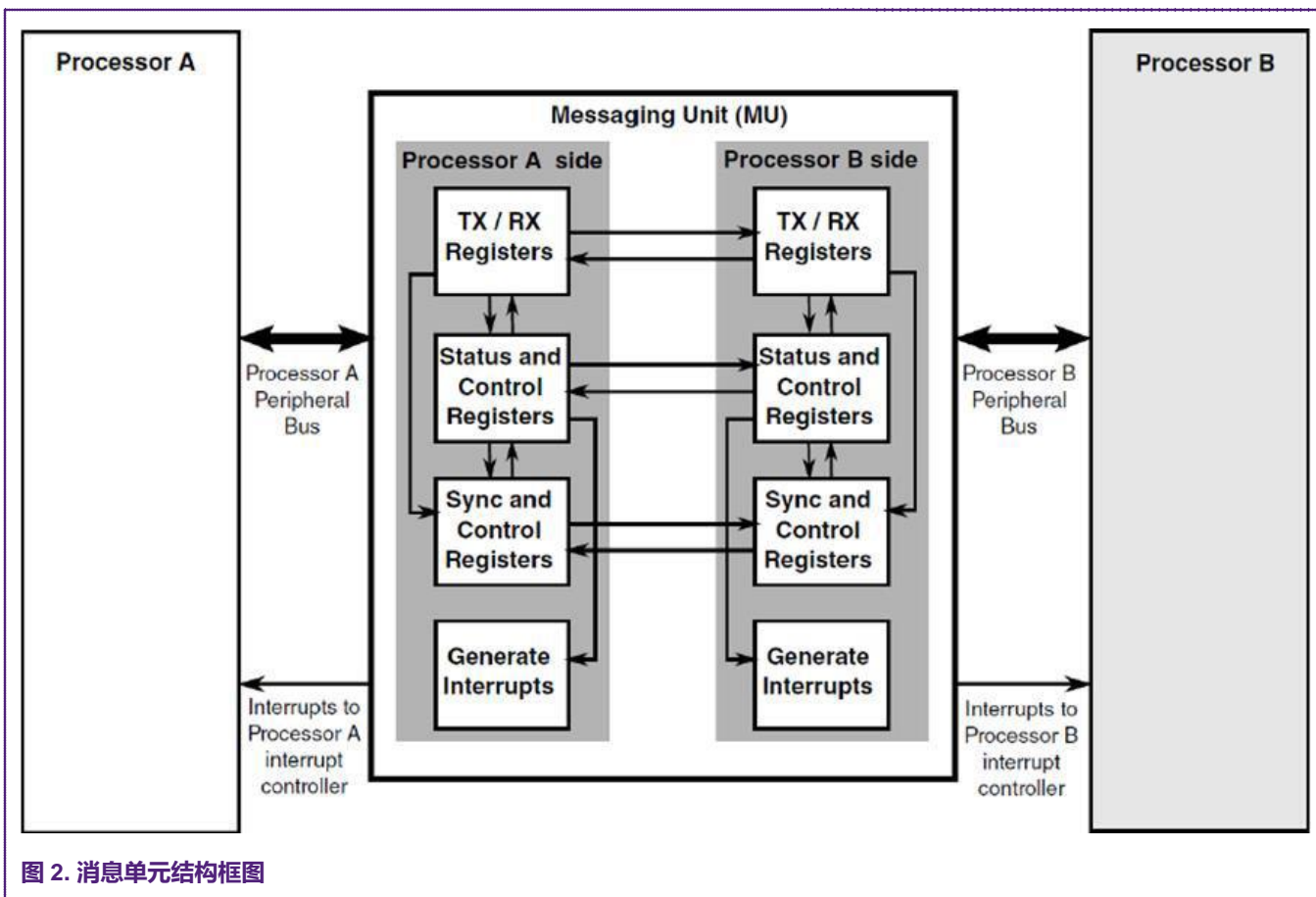


图 1. 多层矩阵框图

2 消息单元 (MU)

消息单元是一个共享的外设，具有 32 位 IP 总线接口以及对应于每个处理器的中断请求信号。消息单元向每个处理器公开一组寄存器，通过 32 位的数据字、中断和标志实现处理器间的通信。图 2 显示了消息单元的框图。



2.1 特性

- 双端口：一个用于 Cortex-M33，一个用于 HiFi4 DSP。
- 通过中断或轮询来控制消息传输。
- 对称的处理器接口，每侧都支持以下功能：
 - 可反映到另一侧的三个通用标志。
 - 可反映到另一侧的四个通用中断请求。
 - 四个接收寄存器，带有可屏蔽的中断。
 - 四个发送寄存器，带有可屏蔽的中断。
- Cortex-M33 可以通过触发其中一个 HiFi4 DSP 中断，使 HiFi4 DSP 内核脱离低功耗模式，反之亦然。

2.2 基本配置

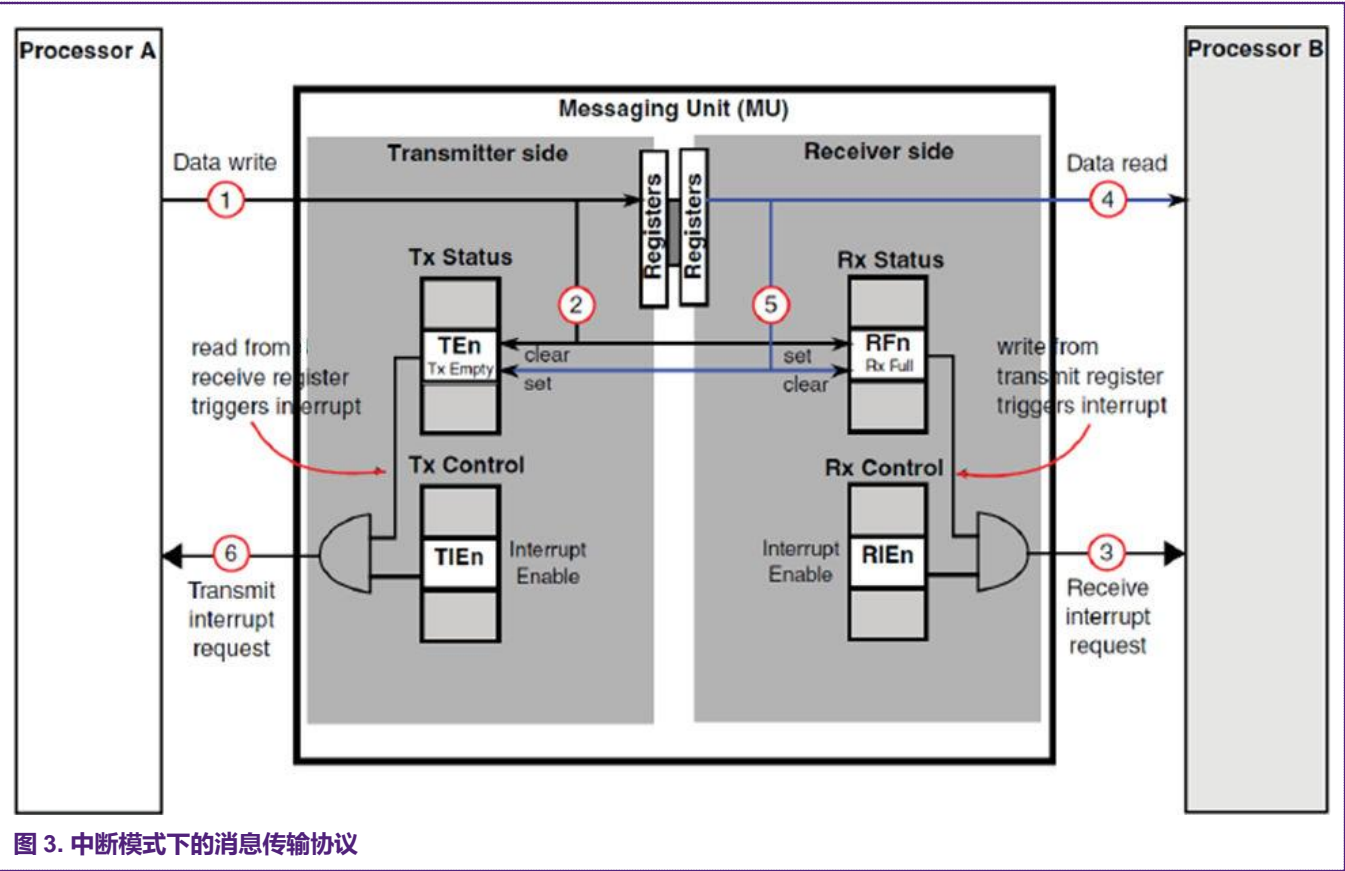
消息单元的初始配置可以通过以下步骤完成：

- 在 CLKCTL1_PSCCTL1 寄存器中启用消息单元的时钟。这将启用寄存器接口和外设功能的时钟。
- 向 RSTCTL1_PRSTCTL1_CLR 寄存器写入内容，可清除 RSTCTL1_PRSTCTL1 寄存器中消息单元外设的复位状态。

2.3 中断模式下的消息传输协议

以下步骤和图 3 描述了使用发送/接收寄存器 (TR/RR) 和中断方式的消息传输模型。这个例子描述了启用了发送和接收中断，从处理器 A 发送到处理器 B 的消息传输顺序。

- 1. **处理器 A 数据写入。**
处理器 A 将消息写入发送寄存器 (TRn)。这个消息立即反映在处理器 B 的接收寄存器 (RRn) 中。
- 2. **清除 Tx 空标志位，并设置 Rx 满标志位。**
处理器 A 对 TRn 寄存器的数据写入会清除处理器 A 状态寄存器中的发送器空标志位 (TEn)。除此之外，处理器 A 对 TRn 寄存器的写入还会导致处理器 B 中的接收器满标志位 (RFn) 被置位。
- 3. **生成接收中断请求。**
RFn 位的置位会生成处理器 B 的接收中断请求。
- 4. **处理器 B 的数据读取。**
触发接收中断请求后，处理器 B 对其 RRn 寄存器执行数据读取。
- 5. **清除 Rx 满标志位并设置 Tx 空标志位。**
从 RRn 寄存器中读出数据，会清除处理器 B 中的接收器满标志位 (RFn)，并设置处理器 A 中的发送器空标志位 (TEn)。
- 6. **生成发送中断请求。**
当处理器 A 中的发送器空标志位 (TEn) 置位时，它会生成一个发送中断请求。此时，如果还需要发送的消息，则重复此过程。



2.4 轮询模式下的消息传输协议

以下步骤和图 4 显示了使用发送/接收寄存器和轮询消息传输协议的消息传输模型。这个例子描述了从处理器 A 发送到处理器 B 的消息传输顺序。

1. 等待 TEn 寄存器为空。

处理器 A 需要等待，直到状态寄存器（SR）中的 TEn 位为空。

2. 处理器 A 数据写入。

处理器 A 将消息写入 TRn 寄存器，该消息立即反映在处理器 B 的接收寄存器（RRn）中。

3. 清除 Tx 空标志位并设置 Rx 满标志位。

处理器 A 对 TRn 寄存器的数据写入会清除处理器 A 状态寄存器中的发送空标志位（TEn）。一旦处理器 A 将消息写入 TRn 寄存器，就会设置处理器 B 中的接收满标志位（RFn）。

4. 等待，直到状态寄存器上的 RFn 位置位。

处理器 B 将轮询状态寄存器上的 RFn 位，直到它被置位。

5. 处理器 B 数据读取。

RFn 位置位后，处理器 B 执行对 RRn 寄存器的数据读取。

6. 清除 Rx 满标志位，并设置 Tx 空标志位。

从 RRn 寄存器中读出数据会清除处理器 B 中的接收满标志位（RFn），并设置处理器 A 中的发送空标志位（TEn）。此时，如果还需要发送的消息，则重复此过程。

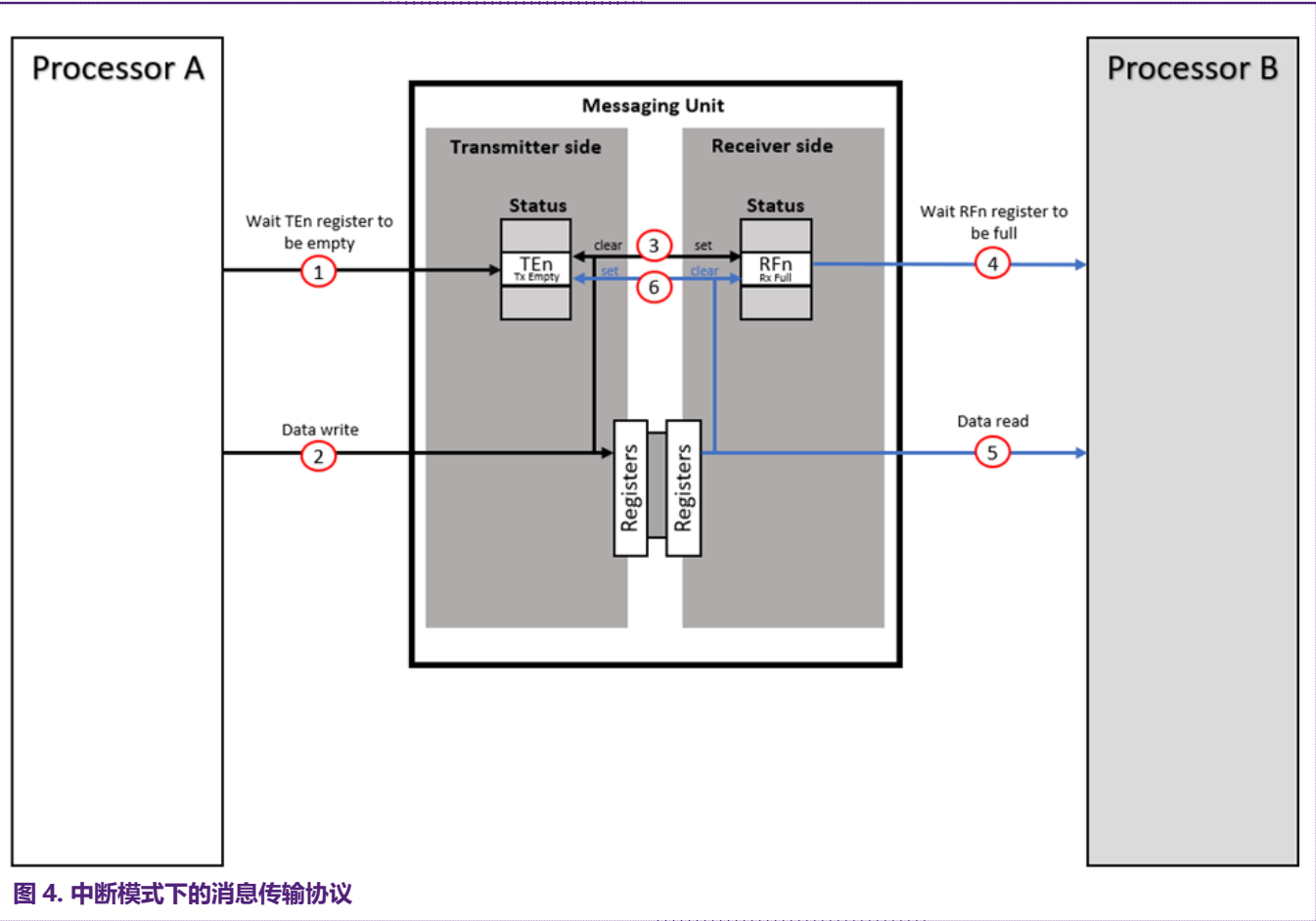


图 4. 中断模式下的消息传输协议

3 信号量模式

多处理器系统，如 RT600，需要一种用来提供锁定机制的方法，让系统采用该机制对共享数据、共享硬件资源等的访问进行控制。这些门控机制用于防止竞争状态，并保持不同进程和处理器之间的内存一致性。信号量模块提供了在 RT600 中实现这一目标所需的硬件支持。

3.1 特性

- 在多处理器配置中支持 16 个硬件实现的门。其中 `cp0` 表示 Cortex-M33，`cp1` 表示 HiFi4。而 `cpX` 表示内核处理器 X。
- 这些门表现为一个 16 入口的，字节大小的数组，可进行读写访问。
 - 处理器通过向相应的门写入 `processor_number+1` 来锁定门，并必须读取门值来验证锁定操作是否成功。
 - 门锁定后，由将其锁定的处理器写入 0 来解锁。
- 每个硬件门表现为一个 16 个状态、4 位的状态机。
 - 16 个状态分别对应以下含义：
 - 如果门值为 0x0，那么该门处于未锁定状态。
 - 如果门值为 0x1，那么该门状态由处理器（主机）0 锁定。
 - 如果门值为 0x2，那么该门状态由处理器（主机）1 锁定。
 - ...
 - 如果门值为 0xF，那么该门状态由处理器（主机）14 锁定。
 - 使用逻辑总线主机号作为参考属性，并使用指定的数据模式来验证所有写入操作。
- 支持安全复位机制来清除各个门的内容，以及全部清零功能。

3.2 基本配置

信号量块的初始配置可以通过以下步骤完成：

- 在 `CLKCTL1_PSCCTL1` 寄存器中启用信号量块的时钟。这会启用寄存器接口和外设功能的时钟。
- 通过写入 `RSTCTL1_PRSTCTL1_CLR` 寄存器，清除 `RSTCTL1_PRSTCTL1` 寄存器中的信号量块外设的复位标志。

3.3 功能说明

以下步骤和图 5 描述了处理器 A 在修改共享数据之前阻塞门的过程。

- **等待门解锁。**
如果门被处理器 B 或一个进程锁定，处理器 A 将等待直到门解锁。
- **锁定门。**
处理器 A 通过写入 `processor_number+1` 来锁定门。
- **读取门值。**
处理器 A 必须读取门值来验证锁定操作是否成功。
- **通知处理器 B 门已锁定。**
处理器 A 必须通知处理器 B 门已锁定，这是通过消息单元完成的。

- 访问共享数据。
处理器 A 修改共享数据。
- 解锁门。
最后，处理器 A 解锁门。

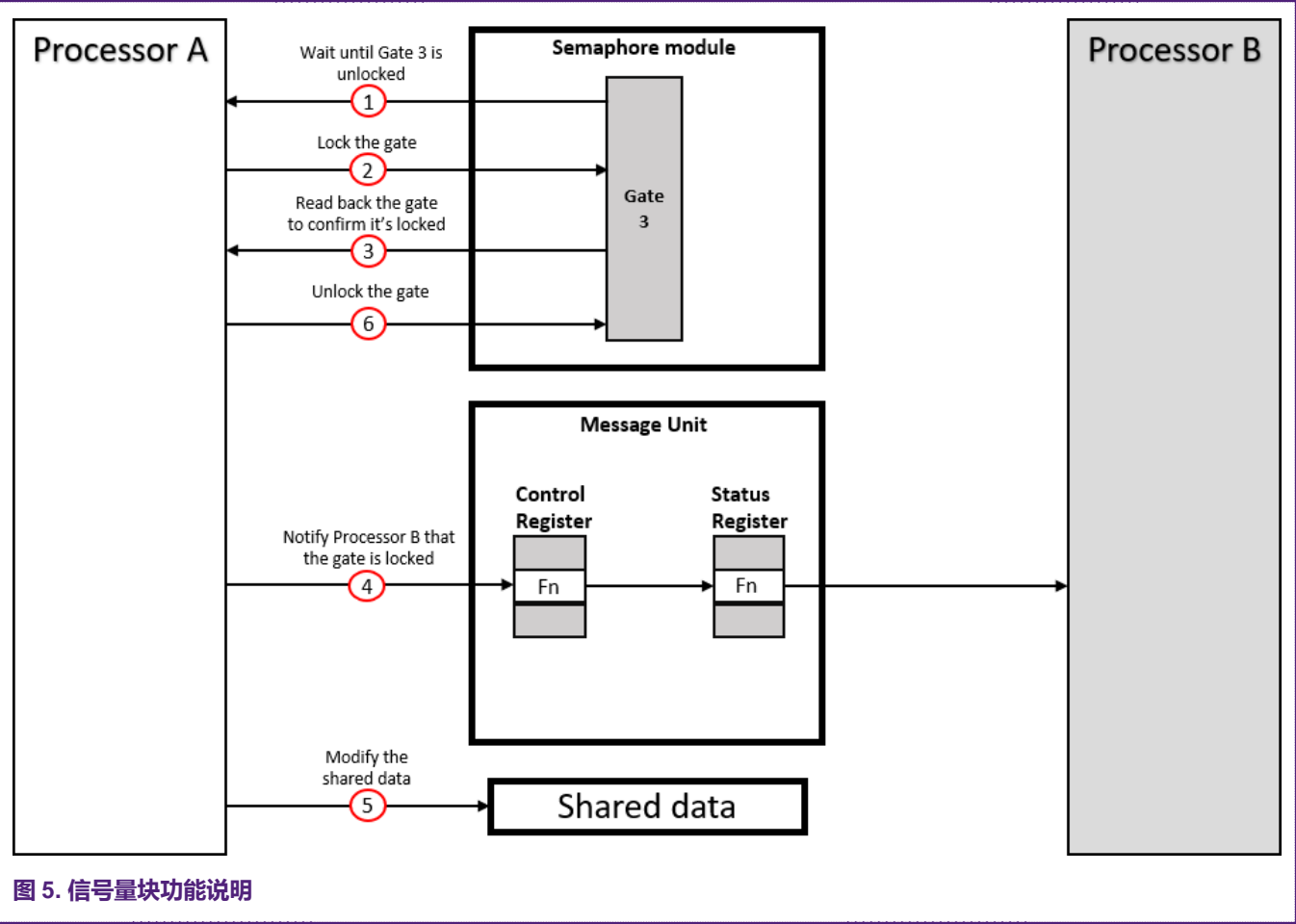


图 5. 信号量块功能说明

4 软件实现

RT600-EVK 的 SDK 提供了消息单元和信号量块的示例。SDK 可以从 <https://mcuxpresso.nxp.com> 下载。

为了更好地了解如何在轮询和中断模式下正确配置信号量块和消息单元，本节简要浏览一下 SDK 中的 `mu_interrupt`、`mu_polling` 和 `sema42` 等示例。

4.1 引脚设置

消息单元和信号量功能与任何设备引脚无关。

4.2 中断模式下的消息传输实现

SDK 中提供的 `dsp_mu_interrupt` 演示应用展示了如何在处理器 A 和处理器 B 之间使用中断模式进行双向消息发送，即从 A 处理器到 B 处理器，或者反过来。

这个示例执行以下操作：

- 处理器 A 以中断模式依次向处理器 B 发送 32 条消息。每个发送的消息会触发处理器 B 上的接收满（Receive Full）中断。
- 处理器 B 以中断模式接收消息。读取消息会触发处理器 A 上的发送寄存器空（Transmit Empty）中断。
- 当处理器 B 接收完所有消息后，它会以中断模式将接收到的 32 条消息发送给处理器 A，每条消息都会在处理器 A 上触发接收满中断。
- 处理器 A 以中断模式接收消息。读取消息将触发处理器 B 上的发送寄存器空中断。
- 当处理器 A 接收完所有消息后，它会将接收到的消息与它发送的消息进行比较，以确定通信是否成功。

图 6 以高层概括的形式描述了 SDK 示例的工作原理。有关这个示例的更多详细内容，请参阅 SDK 中的项目。

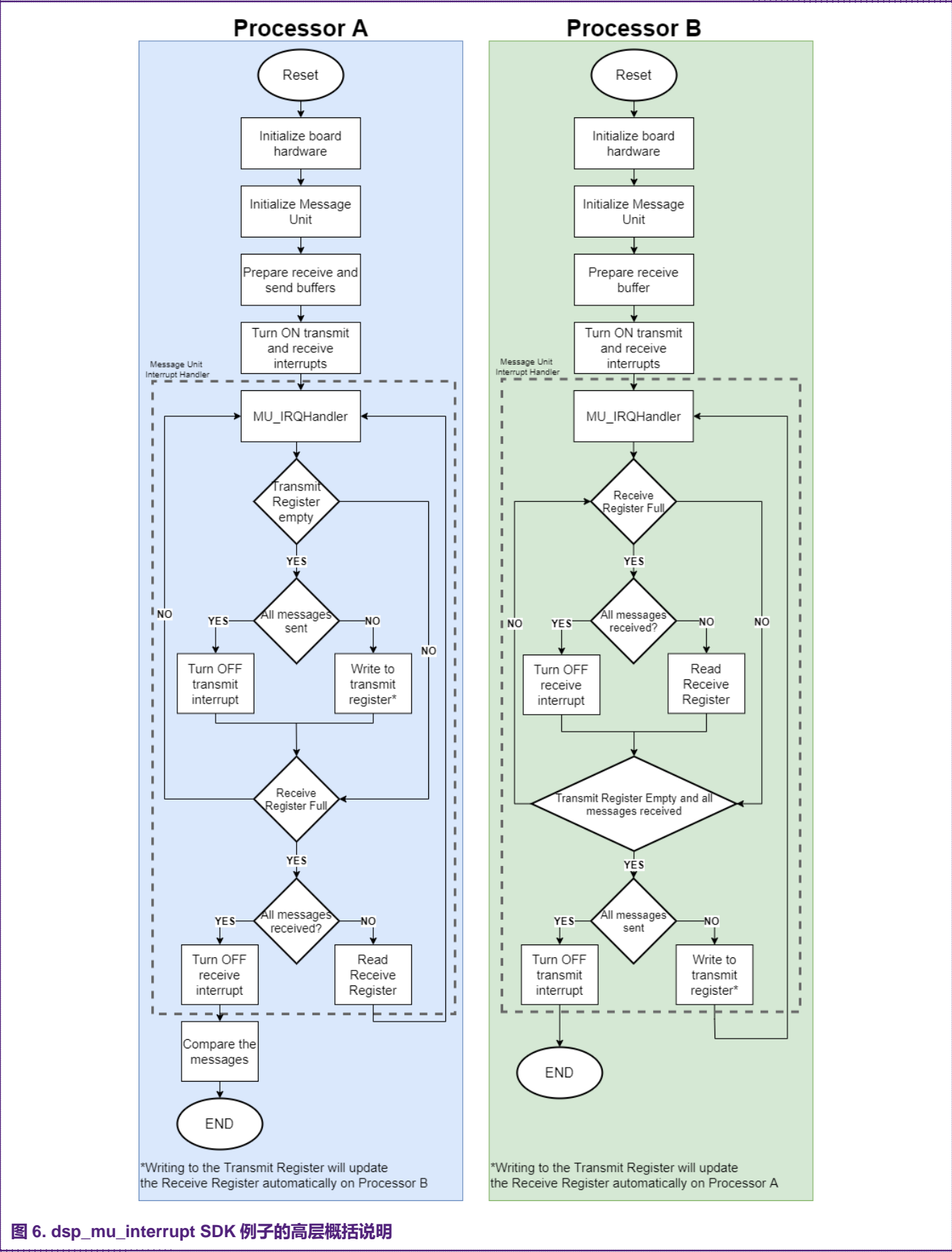


图 7 显示了将在终端中打印的项目运行结果。

```
MU example interrupt!  
Send: 1. Receive 1  
Send: 2. Receive 2  
Send: 3. Receive 3  
Send: 4. Receive 4  
Send: 5. Receive 5  
Send: 6. Receive 6  
Send: 7. Receive 7  
Send: 8. Receive 8  
Send: 9. Receive 9  
Send: 10. Receive 10  
Send: 11. Receive 11  
Send: 12. Receive 12  
Send: 13. Receive 13  
Send: 14. Receive 14  
Send: 15. Receive 15  
Send: 16. Receive 16  
Send: 17. Receive 17  
Send: 18. Receive 18  
Send: 19. Receive 19  
Send: 20. Receive 20  
Send: 21. Receive 21  
Send: 22. Receive 22  
Send: 23. Receive 23  
Send: 24. Receive 24  
Send: 25. Receive 25  
Send: 26. Receive 26  
Send: 27. Receive 27  
Send: 28. Receive 28  
Send: 29. Receive 29  
Send: 30. Receive 30  
Send: 31. Receive 31  
Send: 32. Receive 32  
MU example run succeed!
```

图 7. 在串行终端上输出的消息单元中断方式消息传输示例

4.3 轮询模式下的消息传输实现

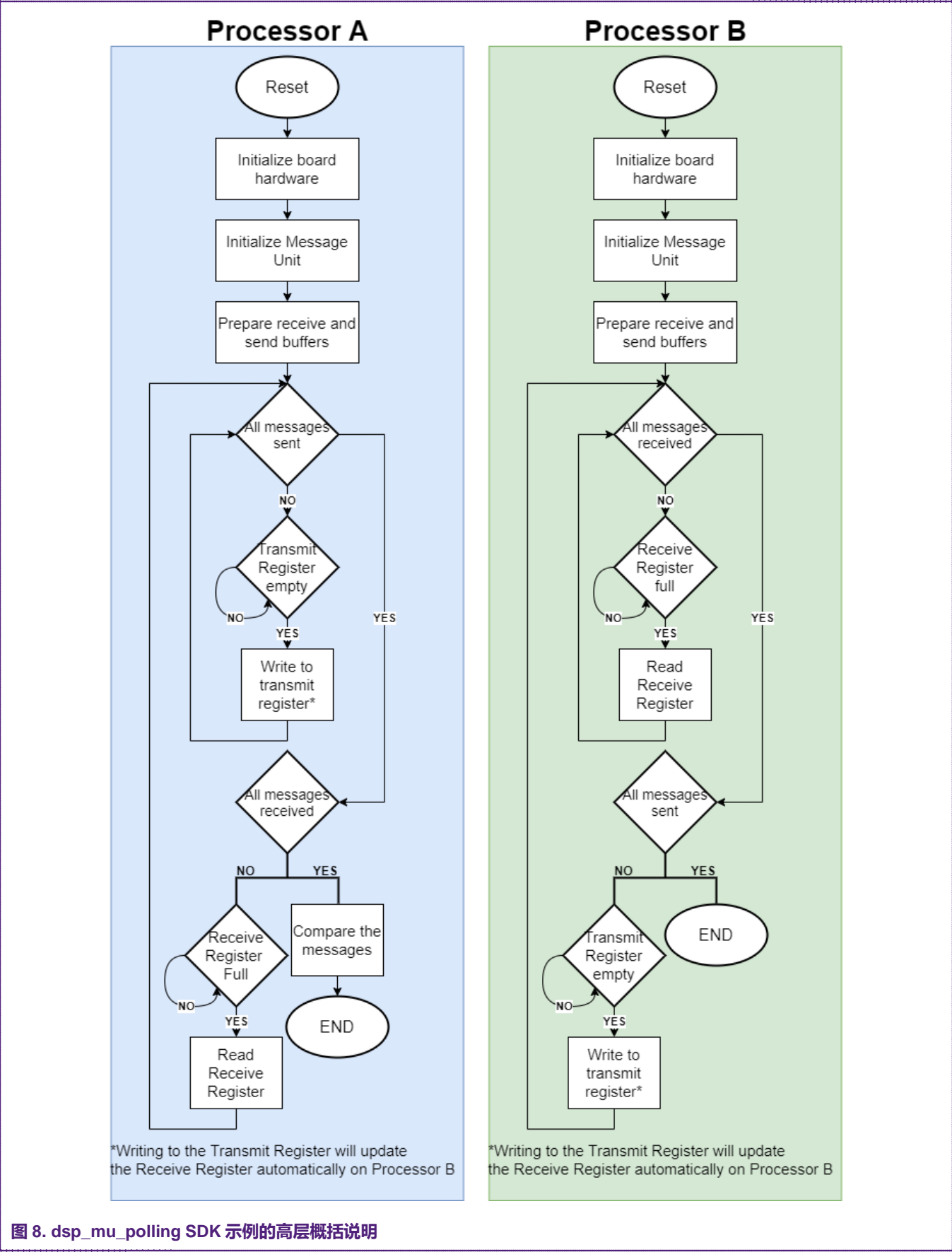
SDK 中提供的 `dsp_mu_polling` 演示应用采用轮询而不是中断方式，在两个处理器之间发送和接收 32 条消息。

这个示例执行以下操作：

- 处理器 A 轮询其发送寄存器。当它为空地，它会将消息一条一条地发送给处理器 B。
- 处理器 B 轮询其接收寄存器。当它已满时，将读取接收到的消息。

- 当处理器 B 接收完所有消息后，会轮询其发送寄存器。当它为空时，开始向处理器 A 发送收到的 32 条消息。
- 处理器 A 轮询其接收寄存器。当它已满时，将读取接收到的消息。
- 当处理器 A 接收完所有消息后，它将比较接收到的消息与它发送的消息，确认通信是否成功。

图 8 以高层概括的形式描述了这个 SDK 示例的工作原理。有关这个示例的更多详细信息，请参阅 SDK 中的项目。



4.4 信号量块的实现

SDK 中提供的 `dsp_sema42` 演示应用展示了如何使用信号量门来控制 EVK 上的 LED。

这个示例执行以下操作：

- 处理器 A 内核点亮 EVK 上的 LED D9，并锁住信号量门。
- 处理器 A 解锁信号量门之前，处理器 B 将等待。
- 当用户在终端窗口中按下任意键后，处理器 A 将解锁信号量门。
- 处理器 B 将锁定信号量门，并关闭 LED。

图 9 以高层概括的形式描述了这个 SDK 示例的工作原理。有关这个示例的更多详细信息，请参阅 SDK 中的项目。

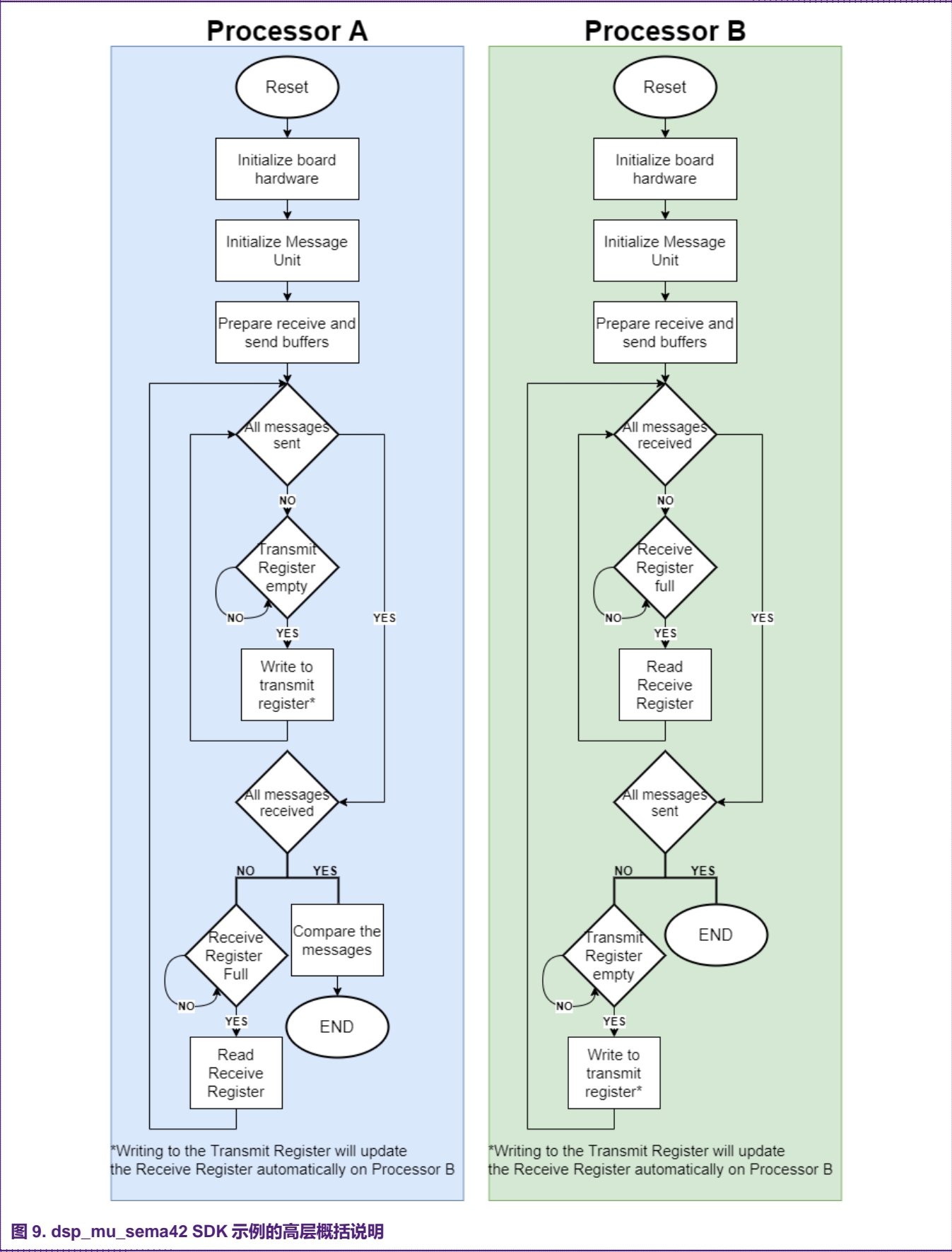


图 10 显示了将在终端中打印的项目运行结果。

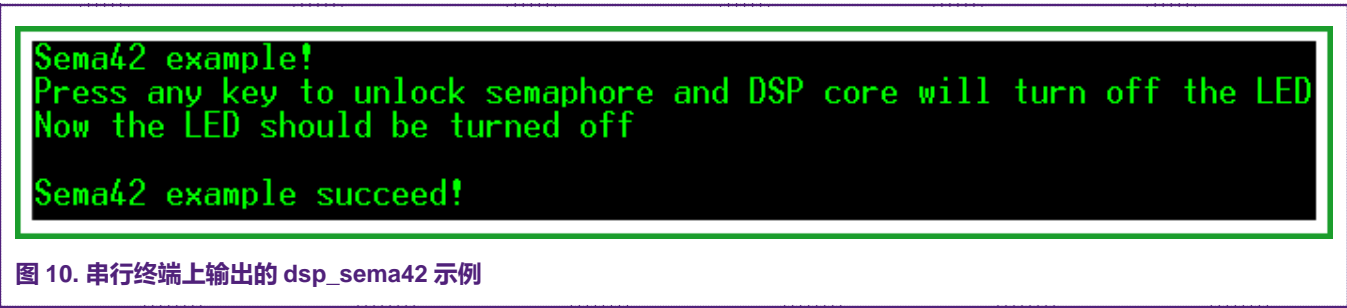


图 10. 串行终端上输出的 dsp_sema42 示例

5 调试双核项目

5.1 调试系统

RT600 系列的双核调试系统具有以下特点：

- 支持针对 Cortex-M33 和 HiFi4 DSP 的 Arm 串行线调试（SWD）模式。
- 跟踪端口提供了 Cortex-M33 CPU 指令跟踪功能。通过串行线查看器进行输出。
- 直接调试访问所有存储器、寄存器和外设。
- 调试会话不需要目标资源。
- Cortex-M33 包括指令断点，也可用于代码补丁的指令地址重映射。
- Cortex-M33 包括数据监视点，也可用作触发器。
- 支持 JTAG 边界扫描。
- 指令跟踪宏单元允许对 Cortex-M33 进行额外的软件控制的跟踪。
- HiFi4 DSP 也包括地址和数据断点以及跟踪功能。

图 11 显示了顶层调试端口和连接。

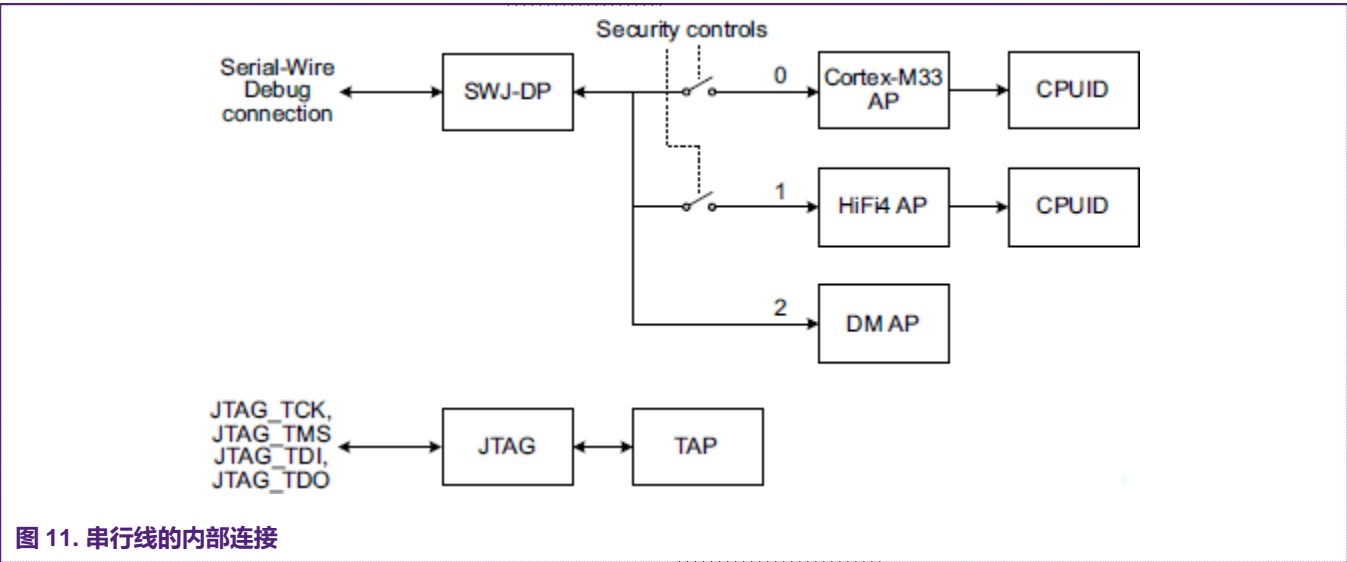


图 11. 串行线的内部连接

5.2 基本配置

串行线调试是 PIO2_25 和 PIO2_26 引脚的默认功能，允许通过复位进行调试。

5.3 双核项目调试

5.3.1 前提条件

本节假设您的个人电脑上已经安装了以下工具和软件：

- Xtensa Xplorer IDE
- 面向 Xtensa Xplorer IDE 的 RT600 DSP Build Configuration
- Xtensa 片上调试器 Daemon
- MCUXpresso IDE V11.1.1 或更新版本
- RT600-EVK 的 SDK v2.7.0 或更新版本
- Jlink 软件 V6.62c 或更新版本

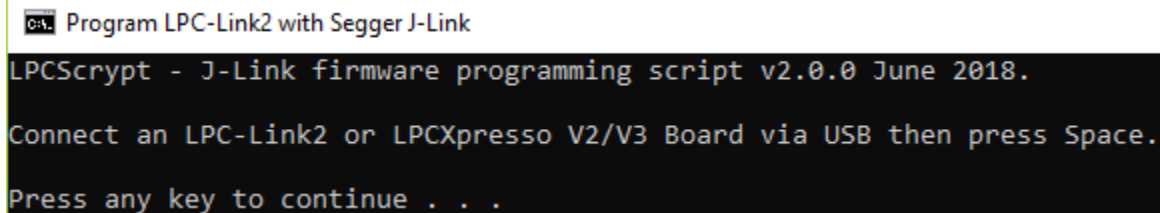
5.3.2 采用 Segger-Jlink 对 LPC-Link2 进行编程

独立的调试器可用于通过 SWD 调试 HiFi4 DSP 和 Cortex-M33。但是，使用 EVK 上的板载 LPC-Link2 调试器更方便。RT600 EVK 上有一个 LPC4300 MCU，预置了 CMSIS-DAP 固件。CMSIS-DAP 与 HiFi4 不兼容；因此，我们需要将 J-Link 固件写入板载调试器。

首先，我们需要安装 LPCScript。它可以从[此链接](#)下载。请确保是 2.1.0 或更高版本。

按照以下步骤对 LPC-Link2 调试器进行编程：

- 连接 EVK 上的跳接线 JP1。
- 通过 USB J5 为 EVK 供电。
- 运行位于 `LPCScript_Installation_Path\LPCScript_2.1.1_15\scripts` 的 `program_JLINK.cmd` 脚本。你将看到图 12 所示的消息。
- 按任意键，开始 Segger-JLink 固件的编程脚本。
- 如果脚本运行成功，你将看到图 13 所示的内容。
- 断开电路板的电源。
- 移除 JP1 跳接线。



```
Program LPC-Link2 with Segger J-Link
LPCScript - J-Link firmware programming script v2.0.0 June 2018.
Connect an LPC-Link2 or LPCXpresso V2/V3 Board via USB then press Space.
Press any key to continue . . .
```

图 12. JLINK 编程脚本

```
Program LPC-Link2 with Segger J-Link
LPCScript - J-Link firmware programming script v2.0.0 June 2018.
Connect an LPC-Link2 or LPCXpresso V2/V3 Board via USB then press Space.
Press any key to continue . . .
Booting LPCScript target with "LPCScript_227.bin.hdr"
LPCScript target booted
.
Programming LPCXpresso V2/V3 with "Firmware_JLink_LPCXpressoV2_20190404.bin"
LPCXpresso V2/V3 programmed successfully:
- To use: remove DFU link and reboot.
Connect Next Board then press Space (or CTRL-C to Quit)
Press any key to continue . . .
```

图 13. JLINK 编程脚本成功运行

要检查 J-Link 固件的编程是否成功，可以运行 Jlink 命令窗口。如果一切顺利，打开 Jlink 命令窗口时，应该能够看到调试器的信息，如图 14 所示。每个 LPC-Link2 都有一个不同的 JLink 序列号，请记住这个号码，因为之后会用到。

```
Select J-Link Commander V6.46k
SEGGER J-Link Commander V6.54c (Compiled Nov 7 2019 17:01:56)
DLL version V6.54c, compiled Nov 7 2019 17:01:02

Connecting to J-Link via USB...O.K.
Firmware: J-Link LPCXpresso V2 compiled Apr 4 2019 16:54:03
Hardware version: V1.00
S/N: 729312828
VTref=3.300V

Type "connect" to establish a target connection, '?' for help
J-Link>
```

图 14. JLINK 序列号

5.4 调试和运行双核应用

5.4.1 调试和运行 MCUXpresso 项目

默认情况下，RT600 启动时不会给 DSP 内核供电。要运行或调试 DSP 应用程序，首先需要在 M33 内核上执行一些代码来初始化 DSP。

MCUXpresso SDK 中包含的 DSP 演示项目由在 Arm 内核和 DSP 内核上运行的两个独立的应用程序组成。例如，要在 DSP 上调试 dsp_mu_interrupt 项目，首先需要用所选的环境来构建并执行 M33 应用程序。本文档中所用的是 MCUXpresso IDE。

首先，将 `dsp_mu_interrupt` 示例导入到您的工作区。默认情况下，MCUXpresso IDE 上的 dsp 示例会将 DSP 预编译的应用程序链接到 RAM 中。当尝试调试多核应用程序时，情况就不是这样了，因为 DSP 映像将从 Xtensa IDE 进行加载和调试。要在 MCUXpresso 上修改这一点，请打开项目属性并访问以下路径：C/C++ Build > Settings > MCU C Compiler > Preprocessor。在 Preprocessor（预处理器）部分，将 `DSP_IMAGE_COPY_TO_RAM=0` 设置为 0，如图 15 所示。

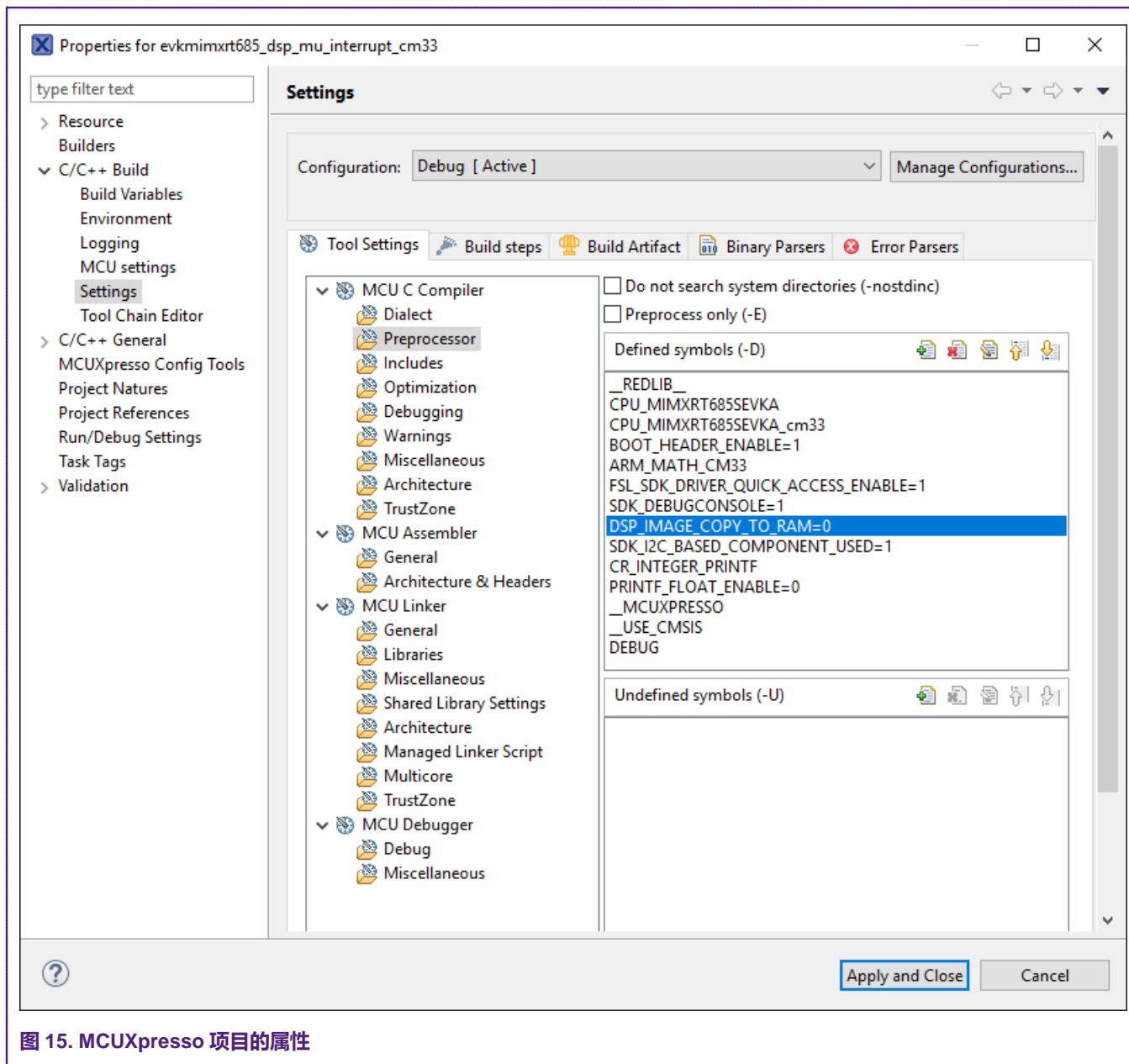


图 15. MCUXpresso 项目的属性

通过这种修改，M33 内核仍然会初始化 DSP，但它不会将预编译的二进制文件加载到 RAM 中。要启动 Xtensa Daemon 调试器，需要让 DSP 运行起来。

进行这个修改后，您可以正常地构建和启动调试会话。调试会话将一直停留在一个循环中，直到 DSP 启动为止，如图 16 所示。

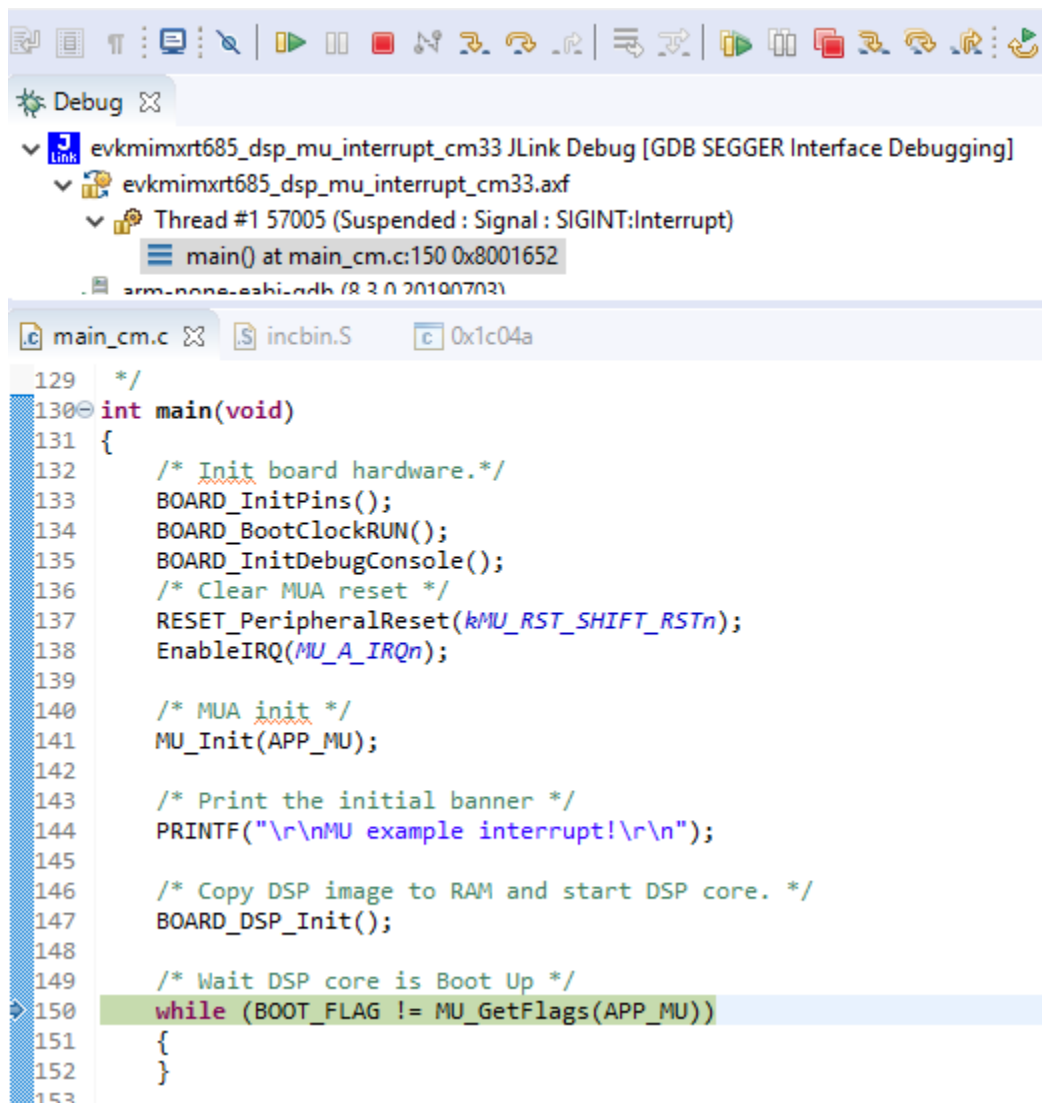


图 16. MCUXpresso IDE 上进行的调试会话

为了与 RT600 兼容，需要 SEGGER J-Link 软件的 6.62c 或更新的版本。MCUXpresso IDE 自带的可能是较旧的版本，可以在 MCUXpresso IDE 的偏好设置中更改版本。可进入 Window > Preferences > MCUXpresso IDE > Debug Options > J-Link Options，如图 17 所示。

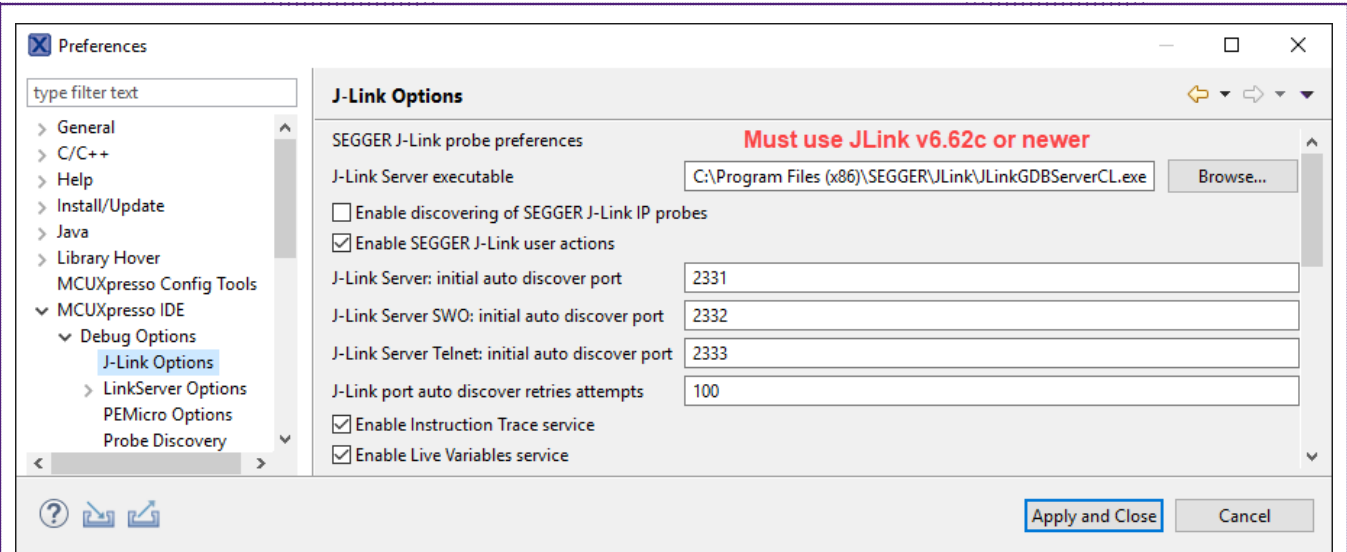


图 17. 在 MCUXpresso IDE 上更改 JLink 软件版本

5.4.2 启动 Xtensa 调试器 Daemon

解释这个过程不属于本文件的范围。请参考位于<SDK_ROOT>/docs/下的 *Getting Started with Xplorer for MIMXRT600.pdf*，了解如何启动 Xtensa 调试器 Daemon 的信息。

注意

对于这一步，需要掌握 JLink 序列号。例如，本示例中使用的调试器如图 14 所示。

5.4.3 调试和运行 Xtensa 项目

一旦调试器 Daemon 启动并运行，即可在 Xtensa IDE 上启动调试配置。

- 1. 将 `dsp_mu_interrupt` 导入 Xtensa IDE。请参阅位于<SDK_ROOT>/docs/下的 *Getting Started with Xplorer for MIMXRT600.pdf*，了解如何将 SDK 项目导入到 Xtensa IDE 中。
- 2. 在项目资源管理器中，你会看到 `dsp_hello_world_hifi4`。使用菜单栏上的下拉按钮，为项目和硬件目标配置进行构建选择，如图 18 所示。

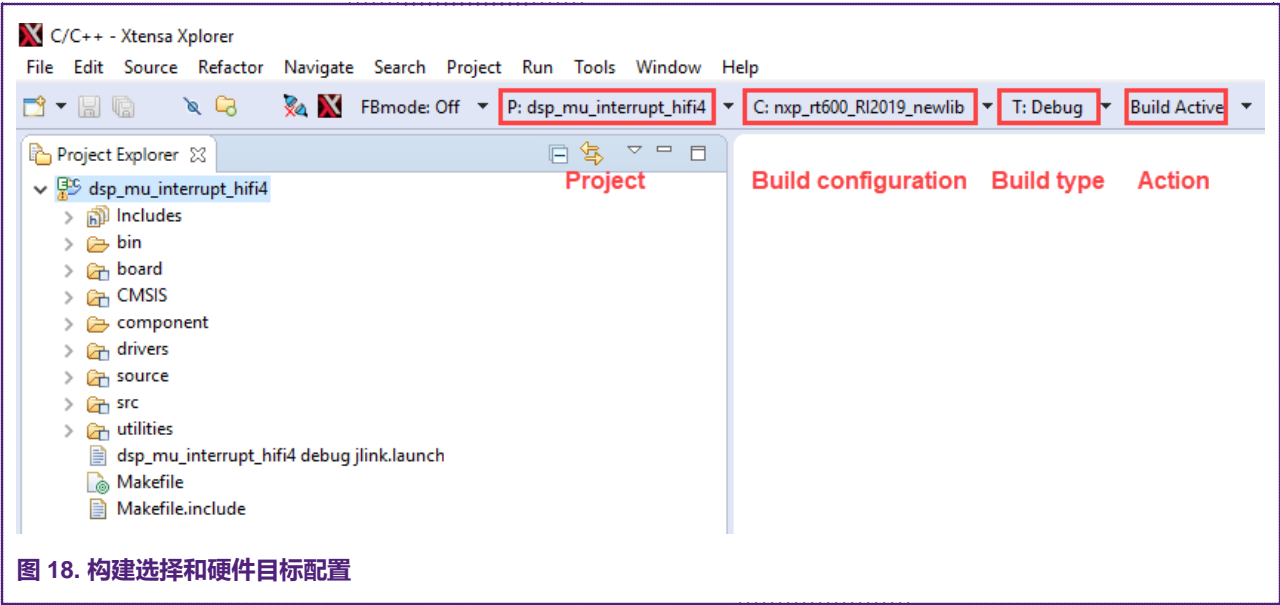


图 18. 构建选择和硬件目标配置

- 3. 点击动作按钮（Build Active）来构建项目。
- 4. 使用菜单栏右侧的 Debug（调试）动作操作按钮来启动调试会话。SDK 项目提供了默认的调试配置，即使用片上调试器。

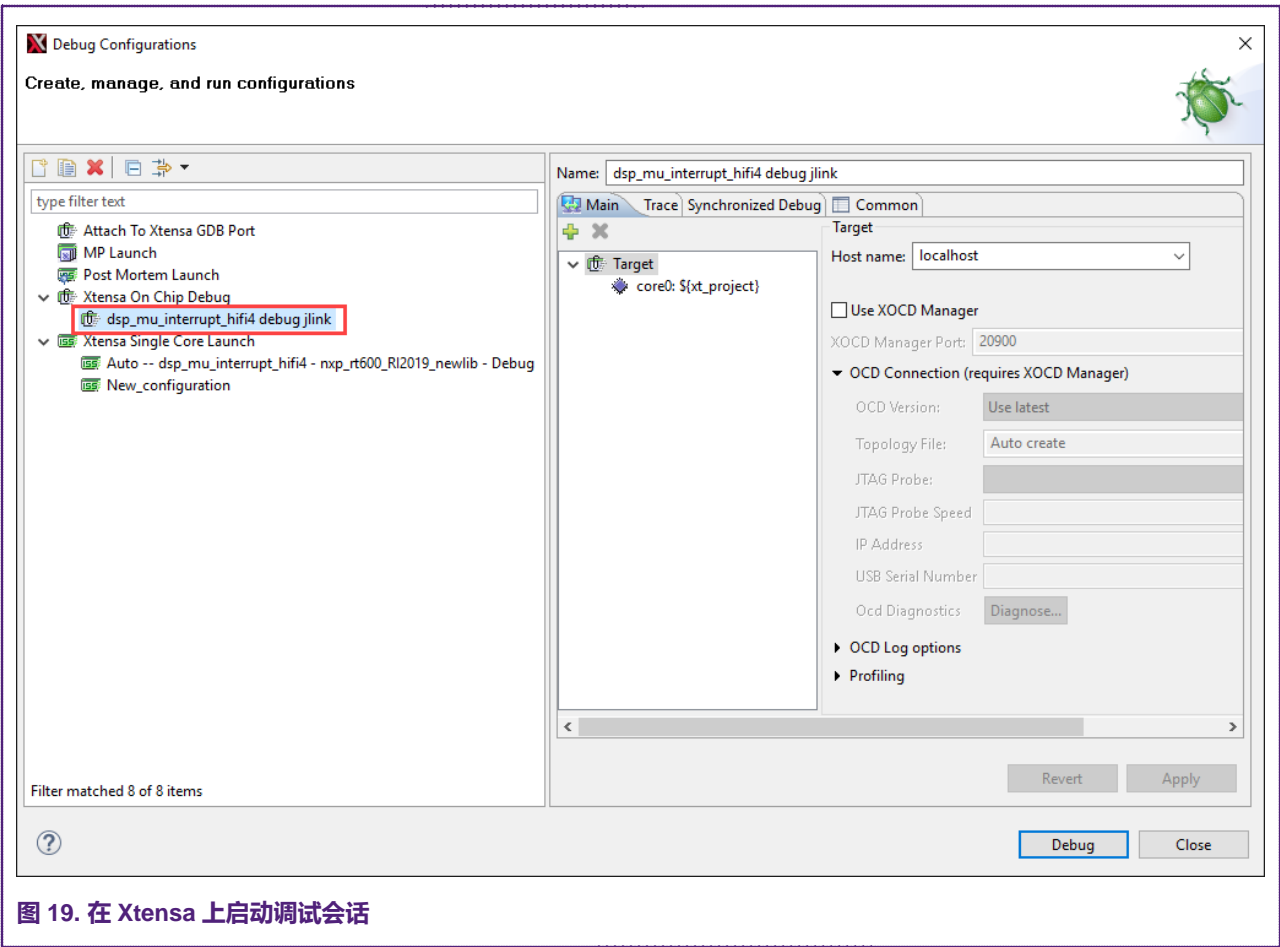
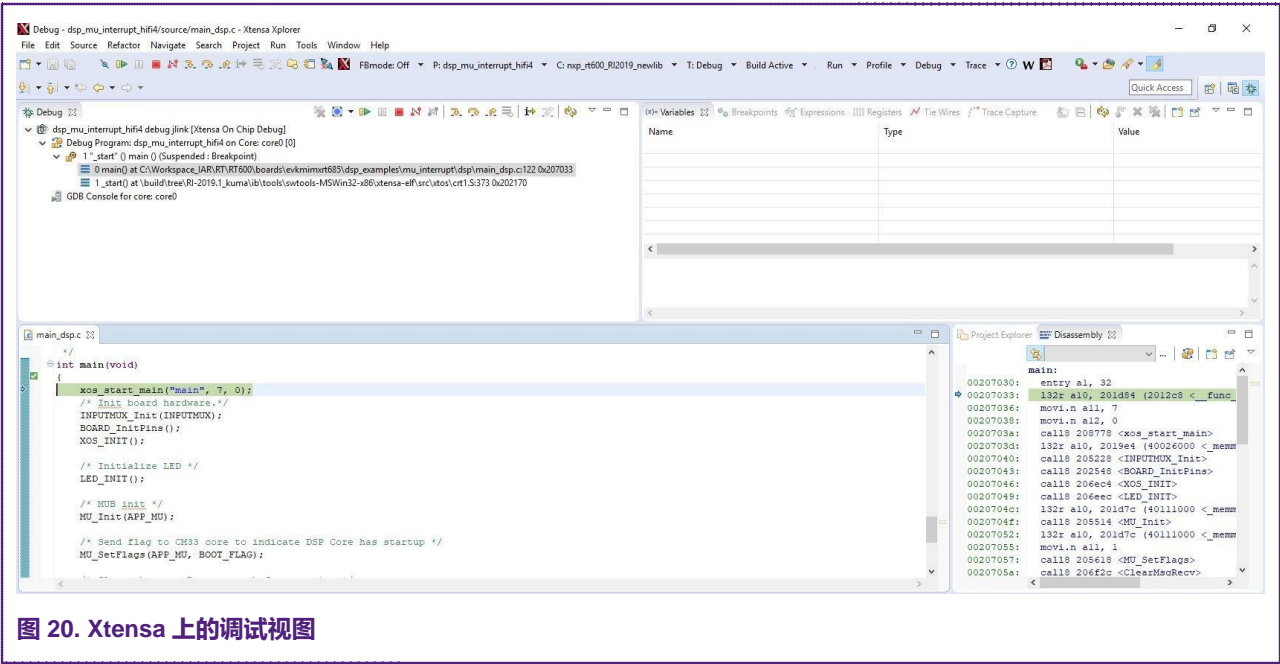


图 19. 在 Xtensa 上启动调试会话

5. 一旦选择了 Debug（调试）按钮，将启动实际的片上调试。Xplorer 会询问是否要将二进制文件下载到硬件上。选择 **Yes**。二进制文件下载完成后，Xplorer IDE 将切换到调试视图。调试视图如图 20 所示。



现在两个调试会话都已经启动并运行了。

How To Reach Us

Home Page:

nxp.com.cn

Web Support:

nxp.com.cn/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com.cn/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, UMEMS, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© NXP B.V. 2020.

All rights reserved.

For more information, please visit: <http://www.nxp.com.cn>

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: March 20, 2020

Document Identifier: AN12789

The logo for Arm, consisting of the word "arm" in a lowercase, blue, sans-serif font.