User's Guide

TI 毫米波 xWRLx432 引导加载程序流程和热复位建议

TEXAS INSTRUMENTS

内容

插图清单

表格清单

商标

1 引言

本文档详细介绍了如何在各种不同用例中使用 ROM 引导加载程序 (RBL)，以及如何解决在这些场景中使用 RBL 可能出现的问题。此外，由于 RBL 具有一定的刚性（因其被编程到器件 ROM 中），本文档还介绍了辅助引导加载程序 (SBL) 及其加载方式（通过任何串行流接口），辅助引导加载程序提供更灵活的加载选项（主映像和辅助映像，或多个独特的用户应用程序）。最后，本文档介绍了复位行为，重点介绍了 xWRLx432 上的热复位行为，以及用户应用程序设计和硬件设计中需要注意的事项。

本文档更深入地介绍了应用程序软件如何与 RBL/SBL 交互，以及如何在引导和复位过程中利用 RBL 和 SBL 实现不同级别的主机控制。有关 RBL 和 SBL 的完整实现详细信息，请参阅 xWRLx432 TRM。

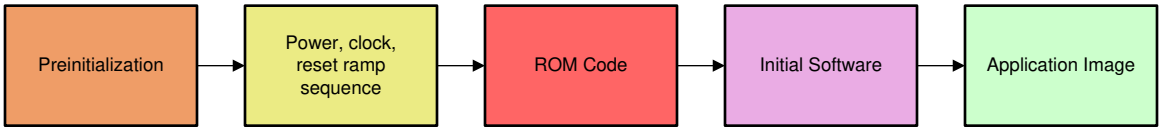

图 1-1. 器件初始化

上文概述了初始化过程，详细信息如下：

- 预初始化：必须保持电源、时钟和控制连接，并且引导配置引脚必须保持在所需的逻辑电平
- 电源、时钟、复位斜坡序列：电源管理芯片应用的特定序列
- ROM 引导加载程序 (RBL)：查找、下载和执行初始软件
- 初始软件：辅助引导加载程序/其他引导仿真软件
- 应用程序映像：在主内核/处理器上运行的应用程序

一旦器件处于功能模式，RBL 将确保应用程序映像正确传输到 RAM，并将器件的控制权交给用户应用程序。如果这个流程因任何常见故障而中断，器件将进入安全状态，且不会引导用户应用程序。

2 基本引导加载程序流程

本节主要关注功能模式下的引导加载程序流。图 2-1 显示了功能模式下从 RBL 执行开始的引导加载程序流程。

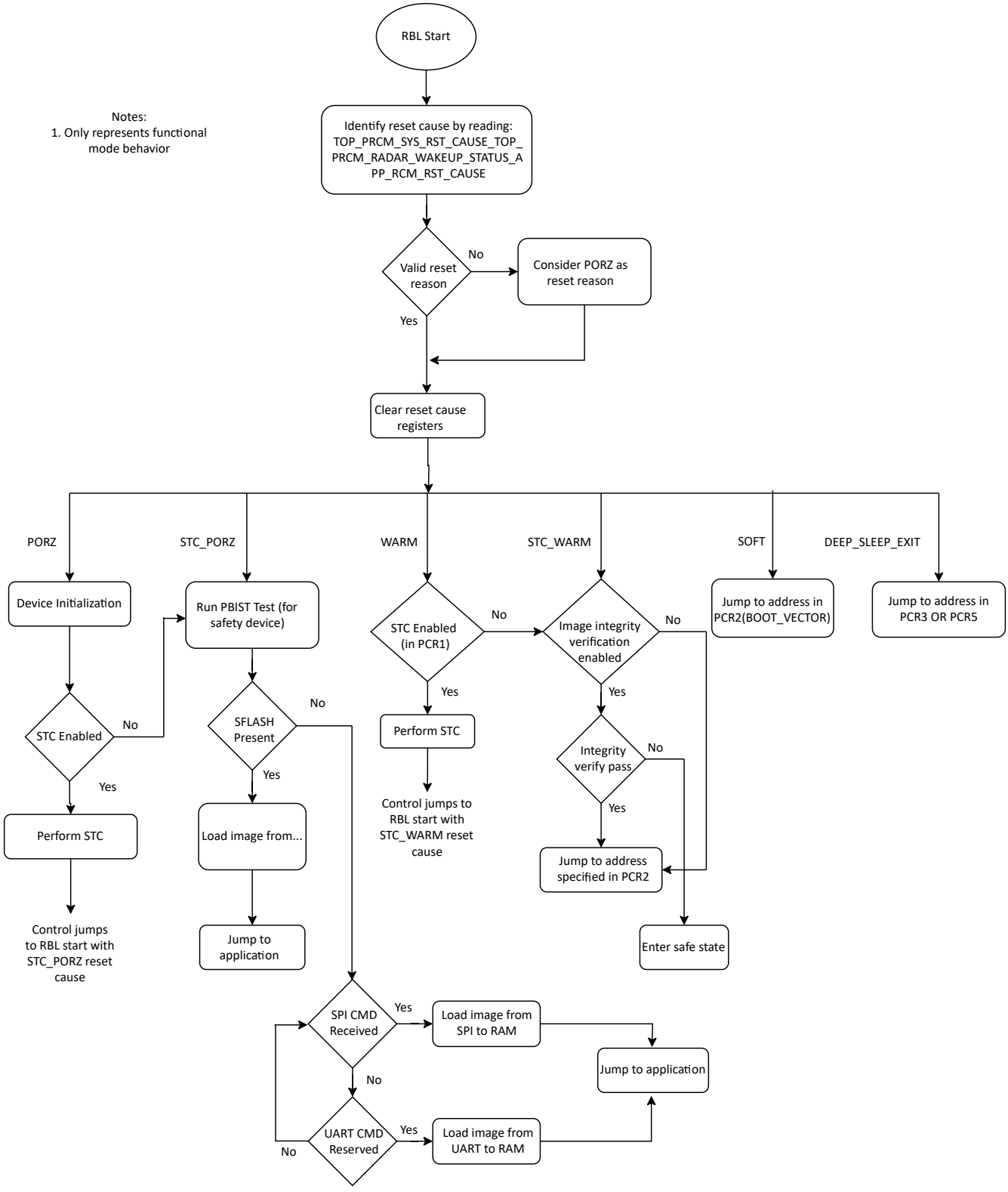

图 2-1. 基本引导加载程序流程图

备注

从 SFLASH 加载或通过 SPI/UART 下载仅在图 2-1 的 STC_PORZ 路径中执行。

2.1 通过 UART 编程串行数据闪存（引导加载程序服务）

TI 的 xWRLx432 器件可配置为自主运行的雷达传感器。在这种配置中，用户应用程序和 TI 固件补丁会通过 QSPI 端口存储在与 xWRLx432 连接的 SDF 中。

SDF 编程支持下载包含以下元件组合的元映像：

- M4F 的用户应用程序映像（应用程序子系统）
- 固件补丁

闪存编程器通过 UART-B 连接到器件。具体细节如下：

- xWRLx432 器件的 UART：
  - RX：焊球 F11/J11
  - TX：焊球 E10/L12
- 波特率：115200
- 数据长度：8 位
- 停止位：1 位
- 奇偶校验：无
- 最大数据块大小：240 字节

备注

“将文件写入 SFLASH”和“将文件写入 SRAM”命令支持的最大数据块大小仅为 240 字节。

该文件分为 N 条命令，其中：

N = (文件大小/240) + ((文件大小%240) ? 1 : 0)

2.2 二进制文件格式

目标二进制文件包含以下部分：

- 标头
- M4F 应用程序
- FECSS 补丁

TI xWRLx432 器件的 MMWAVE-L-SDK 软件包包括 *MulticoreImageGen* 实用程序。该程序使用前面列出的元件构建完整的映像。有关应用映像结构的更多信息，请参阅 TRM 中器件初始化下的引导映像格式

2.3 闪存编程序列

1. 将器件引导至器件管理模式。
2. 打开 *UniFlash* 工具或 SDK 可视化工具（如 xWRLx432 的 MMWAVE-L-SDK 中所列）。
3. 通过 RS232 (UART-B) 应用程序/UART 端口连接到器件（器件需要 UART 中断信号 - 有关更多详细信息，请参阅 xWRLx432 TRM）。
4. 刷写所需的映像 <META_IMAGE1/META_IMAGE2/META_IMAGE3/META_IMAGE4>
   a. 有关元映像的更多信息，请参阅辅助引导加载程序

2.4 支持的 UART 命令/响应及其格式

2.5 刷写序列

xWRLx432 TRM 包含有关使用 xWRLx432 器件通过 UART 刷写应用程序映像的详细信息。

2.6 ROM 辅助映像下载序列

通过将器件置于刷写模式，便会进入 ROM 辅助映像下载序列。请参阅通过 UART 编程串行数据闪存（引导加载程序服务），进一步了解与外部主机握手以接收映像的详细信息。图 2-2 显示了与串行数据闪存 (SDF) 的通信。

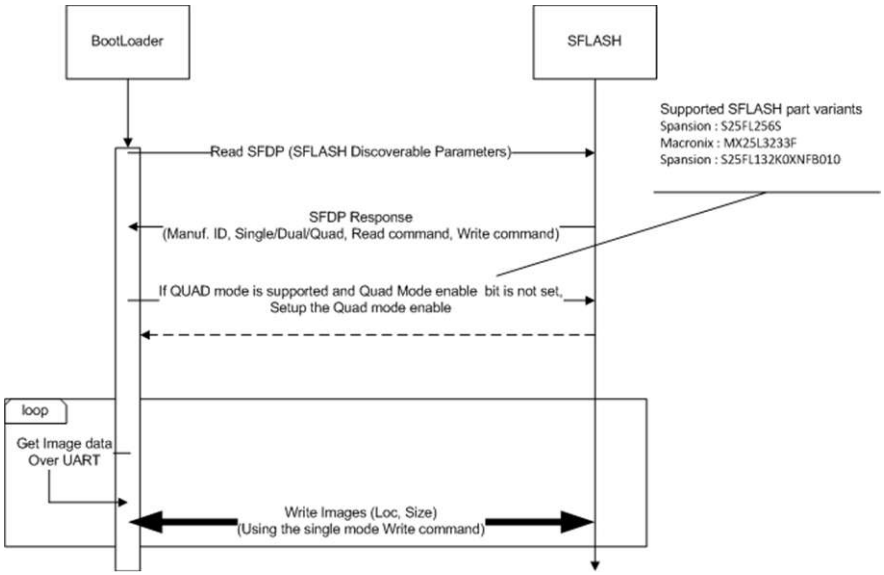

图 2-2. ROM 辅助映像下载序列

要点：

- ROM 辅助下载应支持所有闪存型号，这些型号支持使用 1 个虚拟字节和 24 位寻址的*内存映射模式*和*页面程序命令 (0x2)*。
- QE 位的设置因 SDF 供应商而异。ROM 引导加载程序支持在该流程中为 Spansion® 和 Macronix® 型号（仅限某些特定器件型号）设置 QE 位。
- 除了对通过 UART 接收的每个数据包进行基于校验和的完整性检查之外，还会对整个映像执行基于 CRC32 的完整性检查。当数据包被接收并写入到 SDF 时，会递增计算 CRC32。

2.7 引导应用程序映像

2.7.1 从串行闪存引导

在功能模式下，引导加载程序尝试的第一种引导模式是从 SDF 对映像进行引导加载（参阅图 2-3）。如果由于某种原因，四路或双路读取模式在此过程中失败，则 RBL 会尝试单路模式。

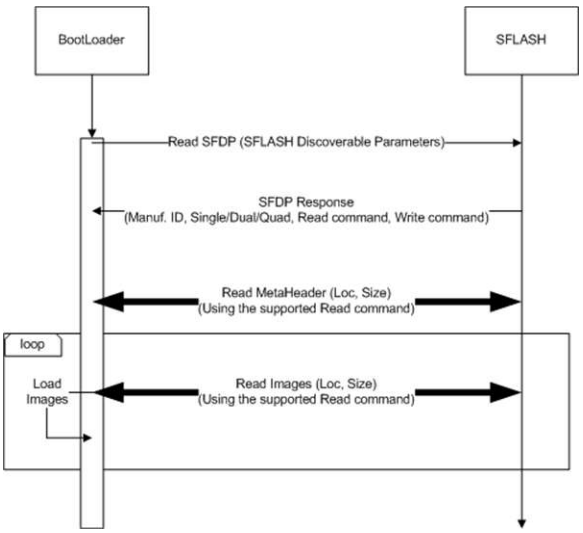

图 2-3. 映像加载序列

此引导模式涉及以下步骤（由 ROM 引导加载程序处理）：

1. 对 xWRLx432 器件的 QSPI 引脚进行引脚多路复用。
2. QSPI 设置为在（系统时钟/2）= (160/2) = 80MHz 下运行。
3. 发出 SFLASH 可发现参数 (SFDP) 命令以检索符合 JEDEC 标准的响应，其中包含有关 SFLASH 功能和命令集的信息。当接收到 SFDP 响应时，该信息用于与 SDF 进行通信并进一步解释内容和加载映像。

要点：

- ROM 引导加载程序根据 SDF 响应 SFDP 命令而发布的最高功能模式（四路、双路或单路），执行从 SDF 读取数据的操作。
- 对于支持四路模式的 SDF 型号，将发出四路模式命令；如果未设置四路使能 (QE) 位，则通信将失败。在此类情况下，加载流程假定 SDF 中的 QE 位已设置。
- 回退映像：如果 SDF 中的某个映像损坏，作为回退机制，引导加载程序支持从以下位置加载映像。映像的位置为：
  - META IMG1（SDF 偏移 – 0x0）
  - META IMG2（SDF 偏移 – 0x80000）
  - META IMG3（SDF 偏移 – 0x100000）
  - META IMG4（SDF 偏移 – 0x180000）

有关映像格式的详细信息，请参阅 *AWRL6432、IWRL6432、AWRL1432、IWRL1432 技术参考手册*。

2.7.2 引导模式 – SPI

当闪存器件不存在或无法通过 JEDEC 响应发现时，引导加载程序会查找要通过 SPI 或 UART 接口加载的映像。如果第一个 PING/命令是通过 SPI 接口发起的，则 RBL 会切换到 SPI 模式进行映像下载。可以从主机 PC 或外部处理器加载此映像。有关通过 SPI 引导的详细信息，请参阅 *AWRL6432、IWRL6432、AWRL1432、IWRL1432 技术参考手册*。

2.7.3 引导模式 - UART

当闪存器件不存在或无法通过 JEDEC 响应发现时，引导加载程序会查找要通过 SPI 或 UARTA 接口加载的映像。如果第一个 ping 命令是通过 UART 接口发起的，则 RBL 会切换到 UART 模式进行映像下载。

UART 通信协议涉及一个简单的命令响应流程。主机首先将命令数据包发送到 xWRLx432 器件，然后等待器件的响应。多字节字段以大端格式传输，先传输 MSB 字节，再传输 LSB 字节。引导加载程序立即处理每个命令数据包，不进行后台调度或处理，随后发送有效的响应数据包。响应可以是 ACK、NACK 或包含一些特定的响应信息。有关通过 UART 引导的详细信息，请参阅 *AWRL6432、IWRL6432、AWRL1432、IWRL1432 技术参考手册*。

3 辅助引导加载程序

从 SFLASH 引导应用程序通常由 ROM 引导加载程序 (RBL) 完成。RBL 将查看 SFLASH，在读取到有效的映像标头后，将该映像加载到 RAM 中并启动应用程序。但是，RBL 的局限性在于它只能引导一个应用程序，不能灵活更改该应用程序，因为如果不修改板上的 SOP 设置，它是无法配置的。如需对应用程序进行更改，需要将器件设置为刷写模式，将新的应用程序加载到 SFLASH，然后返回到功能模式，重新启动引导过程。本节讨论的辅助引导加载程序具有更高的灵活性，可以更为轻松地实现二进制更新，让用户能够更灵活地控制加载内容和加载方式。

辅助引导加载程序 (SBL) 是一种允许通过串行接口对 SFLASH 进行“无线”二进制更新并随后执行新应用程序的工具。该程序允许在 SFLASH 中修改二进制，而无需切换 SOP 模式。它还允许用户刷写多个映像：一个主应用程序映像和一个备份映像，以防无法加载主映像。主映像由 SBL 应用程序通过 UART 直接加载到 SFLASH 中，然后引导应用程序。但是，可以修改 SBL 以使用 SPI、CAN 和 LIN 等其他接口。

与 RBL 不同的是，SBL 只专注于使用应用程序代码引导器件，而不会同时初始化器件。它还可以在 SFLASH 的多个分区中加载映像，而 RBL 只在具有有效映像的第一个分区加载映像。SBL 也是可配置的，因为它是由 RBL 从闪存加载的应用程序，而不像 RBL 那样在器件上进行 ROM。SDK 中 SBL 示例的一些限制包括不支持映像验证，并为没有主机控制的系统增加了复杂性。不过，SDK 中提供了 SBL 源代码，用户可以对其进行修改以解决这些限制，同时还可添加加密等功能，从而满足用例要求。

3.1 SBL 执行流程

SBL 执行流程如下：

- SBL 应用程序由 TI RBL 从 SFLASH 的分区 1 加载

  在执行 SBL 期间，它为用户提供一个可配置的时间量来中断自动引导过程

  - 用户可以通过串行接口发送主应用程序，然后 SBL 将应用程序存储到闪存的正确部分（参阅用于 SBL 执行的闪存存储器分区）
    - SBL 示例可在 MMWAVE-L-SDK（用于 UART）和 Radar Toolbox（用于 CAN/LIN）中找到
  - 刷写后，器件会尝试从闪存引导应用程序代码并将其加载到 RAM 中，并在此过程中对映像执行有效性检查
- 如果可配置的计时器倒计时到期，SBL 会在分区中自动引导映像，如果无效、则会自动引导存储在 SFLASH 的分区 4 中的备份映像（该映像与 SBL 一同刷写）
  - 器件在过程中对映像执行有效性检查
- 如果此次引导成功，SBL 将更新程序计数器并且跳转到被载入到 RAM 中的应用程序

3.1.1 用于 SBL 执行的闪存存储器分区

辅助引导加载程序用例的典型闪存分区 - 来自 SDK SBL 示例

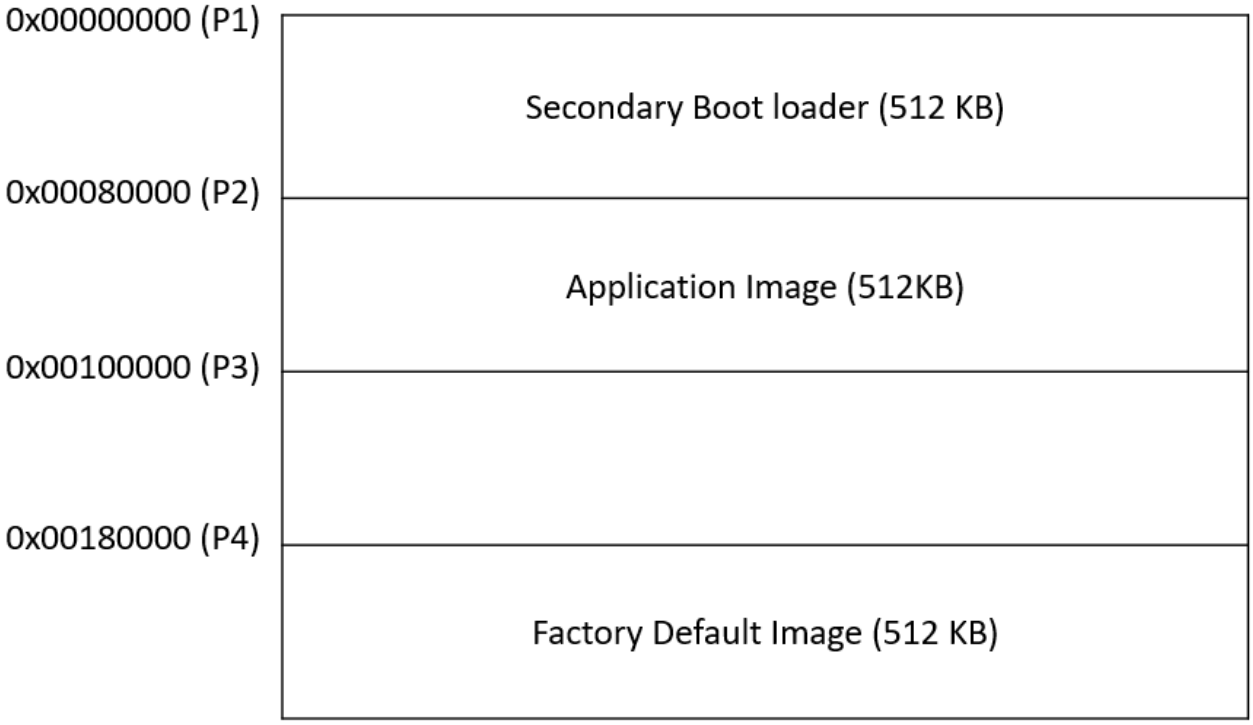

图 3-1. 用于辅助引导加载程序的闪存分区

- SBL 应用程序存储在 SFLASH 的分区 1 中，以便 RBL 可以将此应用程序加载到 RAM
- 出厂备份映像必须存储在分区 4 中。由于校准信息保存在分区的开始处，因此无法通过 UNIFLASH 或任何其他刷写工具将主映像直接加载到分区 2 中
  - SBL 应用程序会将主应用程序加载到分区 2 和分区 3 的正确区域，以避免覆盖存储在 SFLASH 中的校准信息

3.1.2 SBL 特性修改

映像选择：

- 可以修改 SBL 以提供用于将映像加载到 RAM 的选项
  - 目前的流程仅允许自动加载一个映像和一个备份映像，如果在加载主映像期间发生任何问题，备份映像会自动加载
  - 这个流程可以替换为通过 UART_read 等待用户输入，用户选择从 SFLASH 的哪个部分提取映像数据
    - 此选择最多可以加载 2-3 个映像，而不是仅加载一个映像
- 这种修改使器件能够更加灵活地加载内容，并为终端用户提供了多种选择

3.1.3 SBL 开发注意事项

共享存储器注意事项

- SBL 示例从共享存储器运行，以避免与 M4 RAM 中的任何应用程序代码发生冲突。
- 使用 SBL 加载应用程序时，应用程序无法将任何文本、数据或只读信息存储在共享存储器中。有关详细信息，请参阅勘误表中的 DIG#14。但是，在共享存储器中使用 L3 存储器的示例仍然可以使用共享存储器。

一般开发注意事项

- 由于 IWR 和 AWR 器件之间存在 EFUSE 差异，IWR 器件需要显式初始化 RAM1 存储器组

  - IWR 器件默认禁用 ECC，因此需要显式存储器初始化
  - 建议初始化所有存储器组，不过，这样会导致引导时间延迟
    - 也可以通过禁用 SYSTICK 计时器来解决该问题，该计时器会导致在执行 ISR 时对存储器组进行不必要的访问
- 如果对由 SBL 加载的应用程序执行热复位，请确保 SBL 执行热复位注意事项，并且 SBL 触发热复位
  - 特别是在应用程序中触发热复位的看门狗复位，需要特别注意，因为这是一个不可控的复位，可能无法执行确保程序退出后成功热复位所需的所有步骤

4 热复位

如图 2-1 中的 RBL 流程所示，由于 STC_WARM 或 WARM 复位原因，会执行相应的 STC 测试和映像验证，然后执行跳转到存储在引导矢量 (TOP_PRCM:PC_REGISTER2) 中的地址。该地址通常是用户应用程序的起始地址。在此流程中，需要提供必要的硬件基础架构，以确保存储器内容在热复位周期中保持不变，同时器件电源保持完好。在此流程中，RBL 不会默认从 SDF 重新加载映像。

热复位由器件内部生成，或由器件引脚 WARM_RESET 触发。对 TOP_PRCM:RST_SOFT_RESET 寄存器的写入会生成复位。WARM_RESET 可用于从外部复位器件，如果复位由看门狗计时器等内部源生成，则可用于将复位报告给外部。

4.1 完整性验证

RBL 将执行完整性验证步骤，以确保应用程序具有有效数据。在 STC_WARM 复位流程中，RBL 在执行完整性检查时做出以下假设：

- RBL 从 TOP_PRCM:PC_REGISTER2[24:0] 读取引导矢量，以确定要跳转到的引导矢量
- RBL 仅对 APPSS 映像执行完整性检查
- RBL 根据 TOP_PRCM:PC_REGISTER3[31:0] 中的映像长度 (从引导矢量开始) 字段计算 APPSS 映像的 CRC，并将其与 TOP_PRCM:PC_REGISTER4[31:0] 中的数据完整性检查字段进行比较。应用程序应填充该寄存器
- 应用程序应在 TOP_PRCM:PC_REGISTER1[23:20] 和 TOP_PRCM:PC_REGISTER2[19:16] 中正确填充 TOP_PRCM:PC_REGISTER3 和 TOP_PRCM:PC_REGISTER4 的奇偶校验位。有关更多信息，请参阅节 5.2。

4.2 LSTC/PBIST

可编程内置自检 (PBIST) 是一种用于在没有外部测试设备的情况下对 SoC 内存区域进行自检的特性。在嵌入式系统中，这些测试通常在系统引导时或关闭时执行，以检查 SoC 的运行状况。

逻辑内置自检控制器 (LSTC) IP 用于满足功能安全应用的逻辑自检要求，并符合 ISO-26262 ASIL 标准。如果启用了 LBIST，则在上电和/或热复位期间，且使用的是功能安全额定器件时，可以通过 RBL 触发该 LBIST。

更多详细信息，请参阅 *AWRL6432、IWRL6432、AWRL1432 和 IWRL1432 技术参考手册* 中的*安全模块*一节。

4.3 看门狗计时器

xWRLx432 器件提供看门狗计时器模块。看门狗计时器可以用作防止失控代码或无限循环代码的最后一种方法。该模块可防止器件进入未知状态，可能导致较大系统内发生未知功耗或通信中断。

当器件处于低功耗/深度睡眠模式时，WDT 不提供覆盖

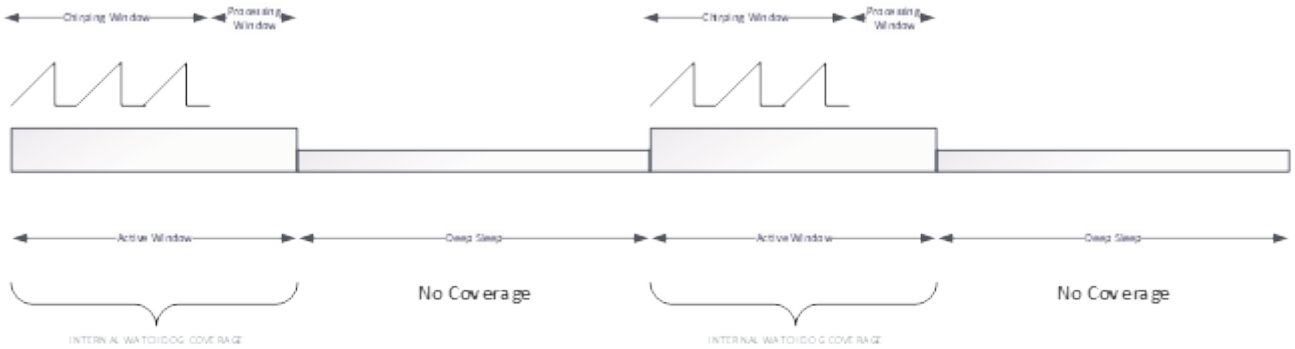

图 4-1. xWRLx432 WDT 覆盖范围

内部看门狗不涵盖器件上的所有故障模式，其中包括但不限于随机位翻转或瞬态故障。系统集成商必须进行适当的功能安全评估，以确定内部看门狗是否足以满足用例需求。

4.4 复位触发的应用程序闪存重新加载

在不同的应用程序用例中，器件热复位时可能需要完全重新加载用户的应用程序。这可确保应用程序映像的“干净”启动，并会使器件免受可能导致失控代码和看门狗热复位的存储器损坏问题的影响。

RBL 设计上不支持在热复位时从闪存加载。因此，如果需要这种行为，建议客户在设计硬件 PCB 时多加注意。

4.4.1 硬件解决方案

存在多种硬件解决方案，可在器件复位时从闪存创建重新加载。下面列出了其中两个

4.4.1.1 PMIC I2C 消息传递

根据系统中选择的 PMIC，可以向 PMIC 发送 I2C 消息，以通过电源触发 nRESET。这样，将实现器件干净重启，同时从闪存存储器中重新加载闪存内容。例如：可以使用 PMIC TPS650360 实现这一目的，前提是 PMIC 的复位输出引脚连接到毫米波雷达器件的复位引脚。毫米波雷达器件可通过 I2C 消息请求 PMIC 热复位，然后 PMIC 将复位毫米波雷达器件。

4.4.1.2 外部看门狗计时器

外部看门狗计时器可用于监控毫米波雷达器件，以检测软件代码（包括应用代码、驱动程序代码等）是否进入锁死状态或执行时间是否增加。外部看门狗计时器可以通过毫米波雷达器件的 GPIO 进行维护。如果外部看门狗计时器未按预期定期维护，则计时器可以向器件触发 nRESET 信号。通过外部看门狗计时器复位后，毫米波雷达器件将进行干净重启，同时从闪存存储器中重新加载闪存内容。外部看门狗计时器的一个示例是 TPS3430。它是一种具有可编程复位延迟功能的汽车窗口看门狗计时器。看门狗计时器还可用于检测硬件故障并从硬件故障中恢复。

4.4.1.3 外部电压监控或电压监控器

这些也称为复位 IC 和电压监控器。这些器件充当外部电压监视器，负责监控或监测电压轨并将信号置为有效，以便在检测到电压轨偏离其允许容差（欠压和过压阈值）时，对另一个器件执行复位、启用或禁用。对毫米波雷达器件进行复位可以实现器件的干净重启，并从闪存存储器重新加载闪存内容。其中一个电压监控器示例是 TPS3850，它可用于检测 1.2V、1.8V 和 3.3V 电源轨的欠压和过压。

4.4.2 软件解决方案

如果客户希望在热复位时从闪存重新加载映像，可以考虑软件端解决方法。下文将详细介绍这两种方法。

4.4.2.1 将引导向量设置为 0x0

通过修改热复位流程中使用的跳转地址，可以实现一种从闪存加载映像的软件端解决方法。另外，还必须特别注意确保前端控制器子系统 (FECSS) 处于正确的状态。用户必须对自己的系统进行必要的安全评估，以确定这个软件端解决方法是否满足他们的需求。解决方法顺序如下：

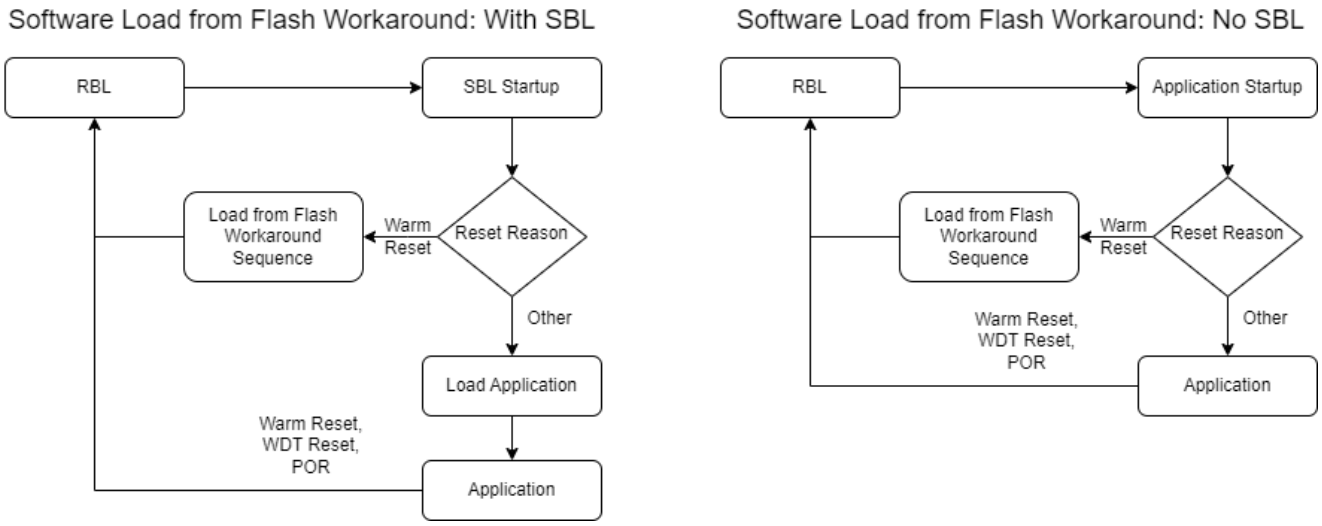

图 4-2. 热复位时从闪存加载软件

1. 缓慢关闭 FECSS 和 HWASS
2. 为 FECSS 上电
3. 将引导矢量写入 TOP_PRCM:PC_REGISTER2[24:0] = 0
4. 将引导矢量偶校验位写入 TOP_PRCM:PC_REGISTER1[15:12] = 0
5. 禁用映像完整性检查 - TOP_PRCM:PC_REGISTER1[5:3] = 0
6. 禁用 STC 测试 - TOP_PRCM:PC_REGISTER1[8:6] = 0
7. 触发热复位 - TOP_PRCM:RST_SOFT_RESET[0] = 1

在步骤 7 中的热复位发生后，执行重新进入 RBL，开始执行，如图 2-1 所示。由于复位寄存器已被清除，RBL 确定复位原因为无效。因此，它假定为 POR，并且再次从闪存加载应用程序映像。应用程序或 SBL 可以按照设计继续，直到发生另一次复位。

查看 MMWAVE-L-SDK 中的 watchdog_reset 示例，如下所示：`<SDK-Install-Directory>/examples/drivers/watchdog/watchdog_reset/`，执行从闪存重新加载的完整编码序列。

5 相关寄存器

初始化之后，RBL 清除所有复位寄存器，并将复位寄存器值存储在该寄存器中。

5.1 复位寄存器

复位寄存器可用于识别触发的复位类型是否与用户想要触发的预期复位相匹配。请参阅下表，了解这些寄存器的说明以及这些寄存器中的值代表什么。

表 5-1. SYS_RST_CAUSE 寄存器

wWRLx432:TOP_PRCM:SYS_RST_CAUSE	字段名称	说明
Bit#16	SYS_RST_CAUSE_SYS_RST_CAUSE_CLR	清除 sys_rst_cause 寄存器 0x0 -> sys_rst_cause 捕获启用 0x1 -> sys_rst_cause 寄存器清除和禁用
Bit#<2:0>	SYS_RST_CAUSE_SYS_RST_CAUSE	系统复位原因寄存器 3'b001 - POR 复位 3'b010 - 软寄存器引起的热复位 3'b100 - 看门狗引起的热复位

备注

在 STC_POR 复位时，SYS_RST_CAUSE 寄存器将为 0x0。只有在真正的 POR 复位时，此字段才会设置为 3'b001。

表 5-2. RADAR_WAKEUP_STATUS 寄存器字段说明

位	字段	类型	复位值	说明
31-21	RESERVED	R/W	X	
20	radar_state_is_deep_sleep	R	0h	RADAR 电源 FSM 处于 DEEP_SLEEP 状态
19	radar_state_is_go_to_deep_sleep	R	0h	RADAR 电源 FSM 处于 GO_TO_DEEP_SLEEP 状态
18	radar_state_is_sleep	R	0h	RADAR 电源 FSM 处于 SLEEP 状态
17	radar_state_is_idle	R	0h	RADAR 电源 FSM 处于 IDLE 状态
16	radar_state_is_wake_up	R	0h	RADAR 电源 FSM 处于 WAKEUP 状态
15-9	RESERVED	R/W	X	
8	wakeup_status_clear	R/W	0h	清除唤醒状态和源寄存器 0x 0 -> 启用唤醒状态和源捕获 0x 1 -> 清除并禁用唤醒状态和源寄存器
7-2	wakeup_source	R	0h	指示从 SLEEP/DEEP SLEEP 状态唤醒的源 Bit 0 -> 计时器作为唤醒源 Bit 1 -> UART 作为唤醒源 Bit 2 -> SPI 作为唤醒源 Bit 3 -> GPIO 作为唤醒源 Bit 4 -> RTC 计数器作为唤醒源 Bit 5 -> FRC 帧开始中断作为唤醒源
1-0	wakeup_status	R	0h	指示器件的唤醒状态，由 POR 唤醒还是从 SLEEP/DEEP SLEEP 状态中唤醒 0x 1 -> 从 SLEEP 状态唤醒 0x 2 -> 从 DEEP SLEEP 状态唤醒

表 5-3. RST_CAUSE 寄存器字段说明

位	字段	类型	复位值	说明
31-8	RESERVED	R	X	
7-0	COMMON	R	3h	APP CPU 的复位原因寄存器 0x00 - 全部清除 0x01 - 上电复位 (PoR) 0x02 - 子系统复位 (使用 xWRLx432:TOP_PRCM:RST_APP_PD_SOFT_RESET 和 POR RESET 从 PRCM 启动的热复位组合) 0x04 - STC 复位 0x08 - Reserved 0x10 - 仅 CPU 复位，通过写入 xWRLx432:APP_RCM:RST_FSM_TRIG<RST_FSM_TRIG_CPU> 触发 0x20 - 使用 xWRLx432:TOP_PRCM:RST_SOFT_APP_CORE_SYSRESET_REQ (无条件复位 - 通过调试程序) 或 xWRLx432:TOP_PRCM:APP_CORE_SYSRESET_PARAM_WAKEUP_OUT_STATE，从 PRCM 启动的内核复位 0x40 - Reserved

位	来源
[3:0]	**通过引导加载程序识别复位原因** M_BOOT_RESET_REASON_PORZ: (0x1U) M_BOOT_RESET_REASON_PORZ: (0x1U) M_BOOT_RESET_REASON_WARM: (0x2U) M_BOOT_RESET_REASON_DEEPSLEEP: (0x3U) M_BOOT_RESET_REASON_SOFT: (0x4U) M_BOOT_RESET_REASON_STC_WARM: (0x5U) M_BOOT_RESET_REASON_STC_PORZ: (0x6U)
[7:4]	SYS_RST_CAUSE[2:0]
[15:7]	RADAR_WAKEUP_STATUS[7:0]
[23:16]	RST_CAUSE[7:0]

5.2 PC 寄存器

图 5-1 所示为相关 PC 寄存器的组织结构。这些寄存器包含 RBL 使用的重要信息。

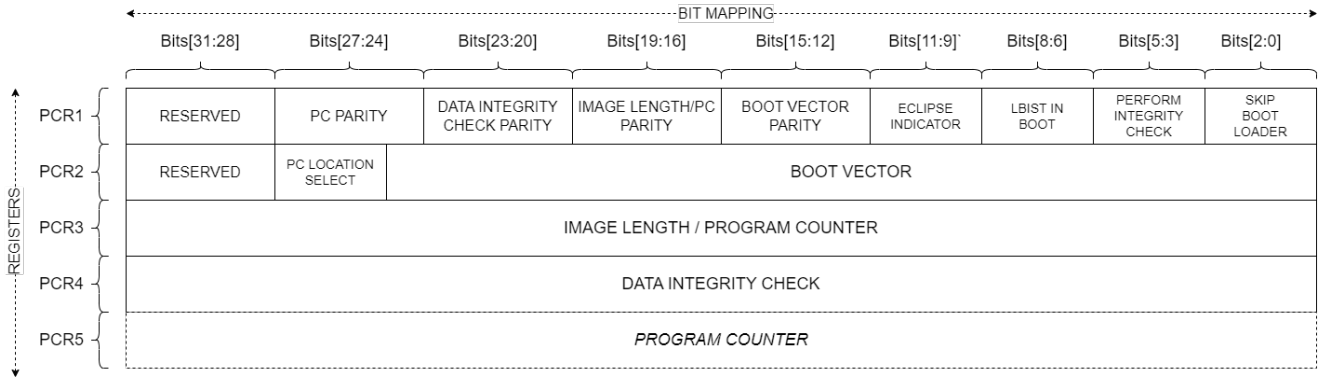

图 5-1. 相关的 PC 寄存器格式

5.2.1 地址

PC_REGISTER1 寄存器 (物理地址 = 5A040054h) [复位 = 00000000h]

PC_REGISTER2 寄存器 (物理地址 = 5A040058h) [复位 = 00000000h]

PC_REGISTER3 寄存器 (物理地址 = 5A04005Ch) [复位 = 00000000h]

PC_REGISTER4 寄存器 (物理地址 = 5A040060h) [复位 = 00000000h]

PC_REGISTER5 寄存器 (物理地址 = 5A040064h) [复位 = 00000000h]

6 修订历史记录

注：以前版本的页码可能与当前版本的页码不同

Changes from Revision * (August 2024) to Revision A (September 2024)	Page
• 更新了节 2.7.3	7

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265