User’s Guide

通过 JTAG 接口进行 MSP430™ 编程

摘要

本文档介绍了使用 JTAG 通信端口擦除、编程和验证基于闪存和基于 FRAM 的 MSP430™ 微控制器系列的存储器模块所需的功能。

内容

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

商标

MSP430™ and Code Composer Studio™ are trademarks of Texas Instruments.
IAR Embedded Workbench® is a registered trademark of IAR Systems.
所有商标均为其各自所有者的财产。

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

1 引言

1.1 关于本文档

本文档介绍了使用 JTAG 通信端口擦除、编程和验证基于闪存和基于 FRAM 的 MSP430 微控制器系列的存储器模块所需的功能。此外，该文档还介绍了如何编程所有 MSP430 器件上均具备的 JTAG 访问安全保险丝。此文档介绍了使用标准四线制 JTAG 接口和两线制 JTAG 接口（也称为 Spy-Bi-Wire (SBW)）的器件访问。

第 3 章描述了一个示例编程器系统，包括软件（提供了源代码）和相应的硬件。该示例旨在提供参考以帮助理解本报告中出现的概念，并帮助开发相似的 MSP430 编程器解决方案。因此，它并不是一个全功能编程工具，而是一本构建此类工具的指导手册。对于那些希望获得即用型工具的用户，应参考 TI 提供的名为 MSP-GANG 量产型编程器的完整编程工具解决方案。

在 Chapter 3 中描述了一个包括源代码和相应硬件的示例编程器系统。该示例旨在提供参考以帮助理解本报告中出现的概念，并帮助开发相似的 MSP430 编程器解决方案。因此，它并不是一个全功能编程工具，而是一本构建此类工具的基本指南。

寻找即用型工具的用户可以参考：

- MSP-GANG 量产型编程器：可同时对多达 8 个 MSP430 器件进行编程的器件编程器
- MSP 调试堆栈：在 MSP 调试器上运行的主机静态库和嵌入式固件
- MSP BSL：在原型设计阶段、最终量产和维护期间与 MSP430 引导加载程序进行通信的软件和硬件

1.2 本文档的结构

本文档介绍了四个主题：

节 2.2 *接口和指令*，介绍了对 MSP430 系列进行编程所需的 JTAG 信号和相关的引脚功能。此外，该部分还介绍了提供的软件宏例程，以及用于通过 JTAG 接口控制并与目标 MSP430 进行通信的 JTAG 指令。

节 2.3 *存储器编程控制序列*，介绍了如何以软件流格式使用所提供的宏和函数原型来控制目标 MSP430 器件以及对存储器进行编程或擦除操作。

节 2.4 *JTAG 访问保护*，介绍了一种可以禁用通过 JTAG 对目标器件的存储器进行访问的机制，这种机制可以出于安全目的禁止不必要的存储器访问。

Chapter 3 *JTAG 编程硬件和软件实现*，介绍了如何使用 MSP430F5437 作为主机控制器来开发一个示例 MSP430 闪存编程器。这一章包含原理图以及所需的软件和工程文件。此外，还包括关于如何使用给定实现方案的完整说明，从而为定制 MSP430 编程器解决方案提供一个可供参考的示例系统。

Copyright © 2021 Texas Instruments Incorporated

2 使用 JTAG 接口进行编程

2.1 简介

本文档概述了如何使用片上 JTAG 接口（4 线制或 2 线制 Spy-Bi-Wire (SBW) 接口）对基于闪存或基于 FRAM 的 MSP430 器件的存储器模块进行编程，重点介绍了用于访问和编程存储器以及所需时序的高级 JTAG 函数。

2.1.1 MSP430 JTAG 限制（与 IEEE 标准 1149.1 冲突）

- JTAG 引脚可与带 TEST 引脚的所有器件的端口功能共用。这些器件包括 5xx、6xx 和 FRxx 系列以及 2xx 和 4xx 系列中的某些器件组（请参阅表 2-14）。在这些器件上，必须发送一个特殊的进入序列才能启用 4 线制 JTAG 连接。有关该序列的说明，请参阅节 2.3.1.1。
- MSP430 器件必须是 JTAG 链中的第一个器件（因为要在 TDI 和 JTAG 保险丝检查序列上计时）。
- 只支持 BYPASS 指令。不支持 SAMPLE、PRELOAD 或 EXTEST 指令。
- 本文档中介绍的 REP430 软件实现方案不应与实时操作系统 (RTOS) 或中断系统一起使用，因为这可能会中断使用 JTAG 或 SBW 进行的编程，尤其是 IR/DR 移位和 TCLK 时钟生成过程。

2.1.2 TAP 控制器状态机

MSP430 JTAG 接口执行由 IEEE 标准 1149.1 规定的测试访问端口状态机（TAP 控制器）。本文档中引用了 IEEE 标准 1149.1 中确定的 TAP 控制器和特定 JTAG 状态。图 2-1 所示为 TAP 状态机。

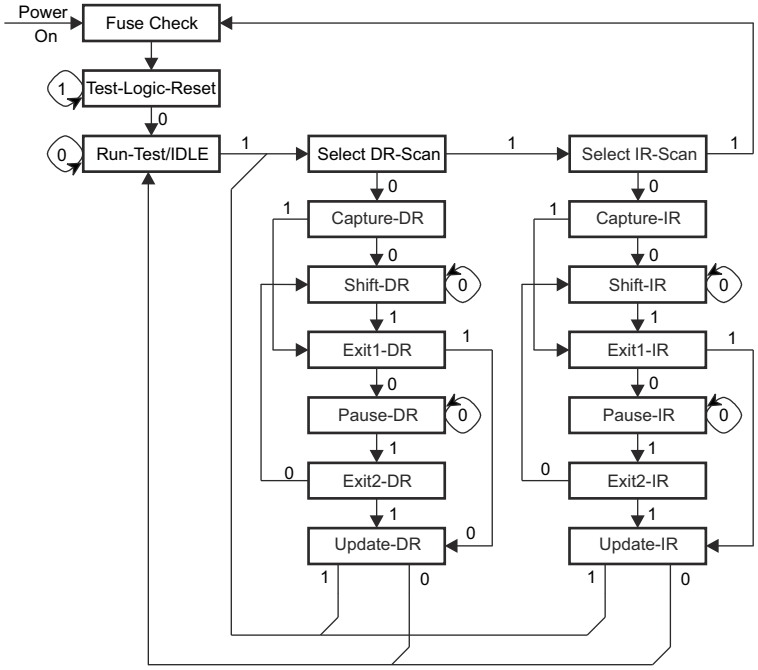

图 2-1. TAP 控制器状态机

2.2 接口和指令

本部分介绍了与 MSP430 器件 JTAG 接口的硬件连接，以及在编程期间使用的相关引脚函数。本部分还介绍了用于对 MSP430 目标进行编程的软件宏例程，以及用于通过 JTAG 接口与目标进行通信并对目标进行控制的 JTAG 指令。

Chapter 3 中详述的 Replicator 示例工程实现了以下高级流程，旨在阐明对 MSP430 目标存储器进行编程的最低要求。

- 启用 JTAG 访问（请参阅节 2.3.1.1）
- 擦除目标存储器（请参阅节 2.3.5 和节 2.3.8.2）
- 验证目标存储器（特殊情况：擦除检查）（请参阅节 2.3.7）
- 写入目标存储器（请参阅节 2.3.4 和节 2.3.8.1）
- 验证目标存储器（请参阅节 2.3.7）

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

- 从 JTAG 控制中释放目标（请参阅节 2.3.2.1.6）

2.2.1 JTAG 接口信号

MSP430 系列支持通过所有 MSP430 器件上提供的 JTAG 端口对闪存和 FRAM 存储器进行电路内编程。所有器件均支持 JTAG 4 线制接口。此外，一些器件还支持下一代优化的 2 线制 JTAG (Spy-Bi-Wire) 接口。通过使用这些工具，可建立一个使用 PC 或者其它控制器访问 MSP430 JTAG 端口的接口连接。请参阅《MSP430 硬件工具用户指南》中针对 *MSP-FET430PIF*、*MSP-FET430UIF*、*GANG430*、*PRGS430 系统内编程和调试的信号连接*部分。

2.2.1.1 2 线制 Spy-Bi-Wire 和 4 线制 JTAG 的优缺点

表 2-1. 2 线制 Spy-Bi-Wire 的优缺点

优点	缺点
只使用两个引脚（TEST（测试）和 RST（复位））	速度慢于 4 线制 JTAG
JTAG 和 IO 引脚间无交叉	

表 2-2. 4 线制 JTAG 的优缺点

优点	缺点
快于 2 线制 Spy-Bi-Wire	使用四个通用输入输出 (GPIO) 引脚
	当使用 JTAG 时，其它引脚功能不可用

2.2.1.2 4 线制 JTAG 接口

标准 JTAG 接口要求将四个信号用于发送和接收数据。在较大的 MSP430 器件上，这些引脚为 JTAG 专用。引脚总数较少的小型器件将这些 JTAG 线路与通用功能复用。在这些小型器件上，需要一个额外的信号来确定共用引脚的状态。这个信号被应用于 TEST 引脚。当由编程器供电时，其余所需的连接为接地和 V_{CC}。表 2-3 中对这些信号进行了说明。

表 2-3. 标准 4 线制 JTAG 信号

引脚	方向	说明
TMS	IN	用来控制 JTAG 状态机的信号
TCK	IN	JTAG 时钟输入
TDI	IN	JTAG 数据输入和 TCLK 输入
TDO	OUT	JTAG 数据输出
TEST	IN	启用 JTAG 引脚（只适用于共用 JTAG 器件）

TEST 输入只存在于具有共用 JTAG 函数的 MSP430 器件上，通常分配给端口 1。对于正常操作（非 JTAG 模式），这个引脚被内部下拉至接地，这将把共用引脚启用为标准 I/O 端口。若要为 JTAG 通信启用这些引脚，请参阅节 2.3.1.1。

TCLK 信号是一个输入时钟，这个时钟必须由一个外部时钟源提供给目标器件。这个时钟在内部用作目标器件的系统时钟 (MCLK) 以将数据加载到存储器位置并为 CPU 计时。没有针对 TCLK 的专用引脚；作为替代，TDI 引脚被用作 TCLK 输入。这种情况发生在 MSP430 TAP 控制器处于运行-测试/闲置状态时。

NOTE

MSP430 XOUT 引脚上支持 TCLK 输入，但是它已经被当前所有基于 MSP430 闪存和 FRAM 的器件上的 TDI 引脚所取代。现有的 FET 工具，以及本文档提供的软件，在 TDI 输入引脚上执行 TCLK。

2.2.1.3 2 线制 Spy-Bi-Wire (SBW) JTAG 接口

在支持 2 线制模式的器件中集成的内核 JTAG 逻辑与只支持 4 线制的器件的 JTAG 逻辑完全一样。根本差别是 2 线制器件在内部执行将 2 线制通信转换为标准 4 线制通信的额外逻辑电路。这样，可完全使用 MSP430 的现有 JTAG 仿真方法。

Copyright © 2021 Texas Instruments Incorporated

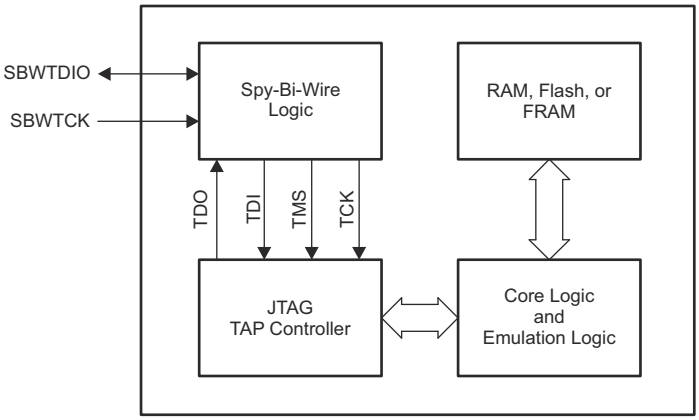

图 2-2. Spy-Bi-Wire 基本概念

2 线制接口由 SBWTCK（Spy-Bi-Wire 测试时钟）和 SBWTDIO（Spy-Bi-Wire 测试数据输入/输出）引脚组成。SBWTCK 信号是时钟信号和一个专用引脚。在正常操作中，这个引脚在内部被拉至接地。SBWTDIO 信号代表数据并且是一个双向连接。为了减少 2 线制接口的开销，SBWTDIO 线路与器件的 $\overline{\text{RST}}$/NMI 引脚共享。

节 2.6 介绍了 MSP430 器件及其各自的 JTAG 接口实现情况。

2.2.2 JTAG 访问宏

为了简化以下部分中对 JTAG 功能的说明，高级宏已被用于描述 JTAG 访问。本文档并未详述基本 JTAG 功能性；而是着重介绍了用于内存访问和编程的 MSP430 专用工具。就本文档而言，有必要显示那些必须加载到 JTAG 指令寄存器中的指令，以及何时需要这些指令。节 2.2.2.1 汇总了整个文档中使用的宏以及它们相关的功能性。更多信息请参阅随附软件。

表 2-4. JTAG 通信宏

宏名称	函数
IR_SHIFT（8 位指令）	将一个 8 位指令移入 JTAG 指令寄存器。同时，此 8 位值由 TDO 移出。
DR_SHIFT16（16 位数据）	将一个 16 位数据字移入一个 JTAG 数据寄存器。同时，此 16 位值由 TDO 移出。
DR_SHIFT20（20 位地址）	将一个 20 位地址字移入 JTAG 内存地址总线寄存器。同时，此 20 位值由 TDO 移出。只适用于 MSP430X 架构器件。
MsDelay（时间）	等待以毫秒为单位的额定时间
SetTCLK	将 TCLK 设定为 1
ClrTCLK	将 TCLK 设定为 0
TDOvalue	包含在 TDO 上移出的最后值的变量

2.2.2.1 针对 4 线制 JTAG 接口的宏

2.2.2.1.1 IR_SHIFT（8 位指令）

这个宏将一个 JTAG 指令载入到目标器件的 JTAG 指令寄存器中。在 MSP430 中，这个寄存器为 8 位宽且最低有效位 (LSB) 最先移入。在写入 JTAG 指令寄存器期间从 TDO 中输出的数据包含在目标器件上执行的 JTAG 接口（或者 JTAG ID）的版本标识符。JTAG ID 以 MSB 在前的方式移出。

不管 TDO 上发出何种 8 位指令，TDO 上的返回值一直为 JTAG ID。来自 TDI 的每个指令位由 TCK 上升边沿上的目标 MSP430 进行捕捉。执行此宏时，TCLK 不应改变状态（当 TAP 控制器处于运行-测试/闲置状态时，TCLK = TDI）。图 2-3 显示了如何将 ADDR_16BIT 指令载入到 JTAG IR 中。请参阅节 2.2.4，了解用于访问目标器件闪存模块的 JTAG 接口通信指令的完整列表。

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

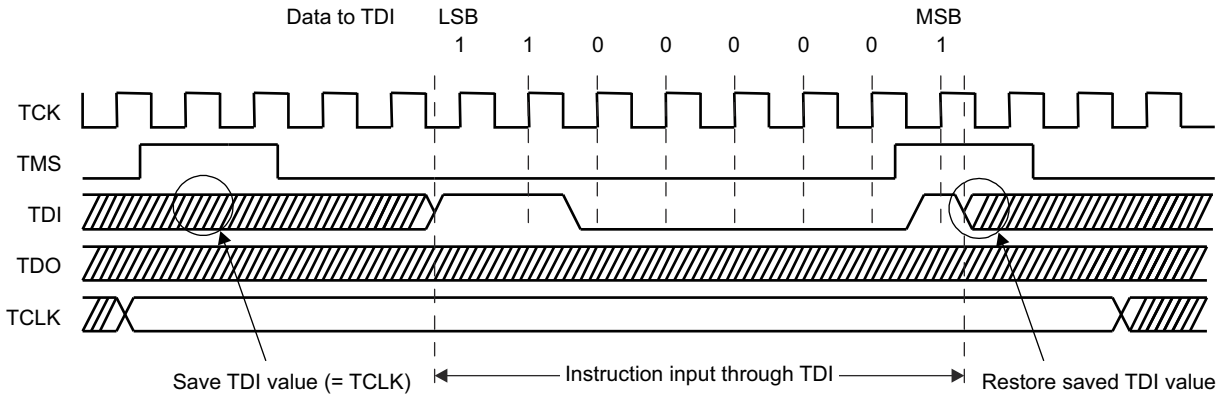

图 2-3. 针对 IR_SHIFT (0x83) 指令的时序示例

2.2.2.1.2 DR_SHIFT16（16 位数据）

这个宏将一个 16 位字载入到 JTAG 数据寄存器 (DR)中（在 MSP430 器件中，一个数据寄存器为 16 位宽）。数据字将移入目标 MSP430 器件的 TDI 输入端（最高有效位 (MSB) 在前）。来自 TDI 的每个位在 TCK 的上升沿上被捕捉。同时，TDO 移出最后一个被捕捉并被存储在地址数据寄存器中的值。在 TCK 的下降沿上，在 TDO 上出现一个新的位。在这个宏被执行时，TCLK 不应改变状态。图 2-4 显示了如何将一个 16 位字加载到 JTAG DR 中并通过 TDO 读出一个已存储的值。

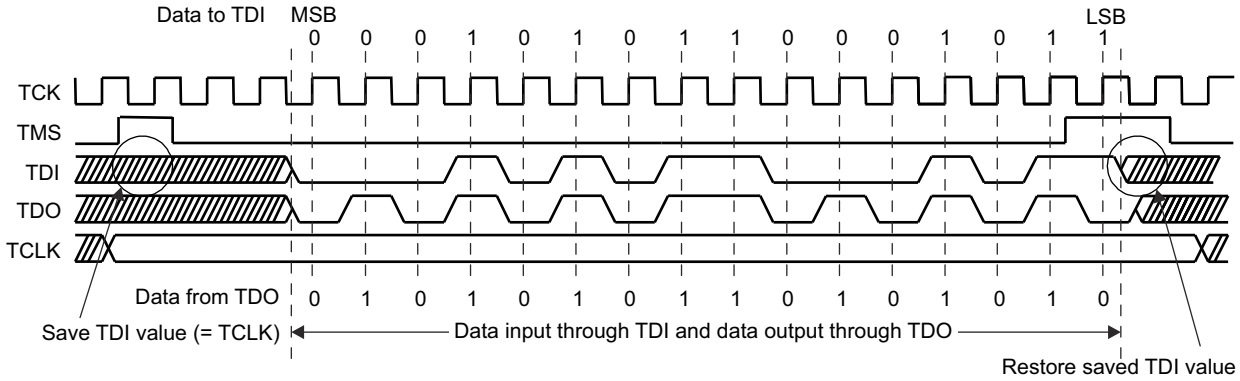

图 2-4. 数据寄存器 I/O：DR_SHIFT16 (0x158B)（TDO 输出为 0x55AA）

2.2.2.1.3 DR_SHIFT20（20 位地址）（只应用于 MSP430X 器件）

MSP430X 架构基于一个 20 位内存地址总线 (MAB)，可寻址高达 1MB 的连续内存。控制 20 位 MAB 无需新的 JTAG 指令（指令细节请参阅节 2.2.4.1），只有 JTAG 地址寄存器本身已经被扩展至 20 位。这个宏将一个 20 位地址字载入到 20 位宽的 JTAG MAB 寄存器中。地址字被移入目标 MSP430 器件的 TDI 输入端（MSB 在前）。在 TCK 的上升沿上捕捉来自 TDI 的每个位。同时，TDO 移出最后一个捕捉到的并且存储在 JTAGMAB 寄存器中的值。在 TCK 的下降沿上，在 TDO 上出现一个新的位。当这个宏在执行时，TCLK 不应改变状态。仅当通过 JTAG 操作 MAB 之前已将 IR_ADDR_16BIT 或 IR_ADDR_CAPTURE 加载到 JTAG 指令寄存器中时，才应使用此宏。在进行 20 位移入移出访问时，JTAG 地址寄存器的上四位 (19:16) 将最后移出。因此，当通过执行一个 16 位移入移出操作来访问 MAB 的低位部分时，MAB 的位 15 被首先读取。这个执行确保与最初 MSP430 架构和其 JTAG MAB 寄存器执行的兼容性。

NOTE

相关 C 语言代码软件示例应用中的 DR_SHIFT（20 位地址）宏自动将交换的 TDO (15:0)(19:16) 输出重组为一个连续的 20 位地址字 (19:0) 并且只返回一个 32 位长值。

图 2-5 显示了如何将一个 20 位地址字加载到 JTAG 地址寄存器中并通过 TDO 读出一个已存储的值。

Copyright © 2021 Texas Instruments Incorporated

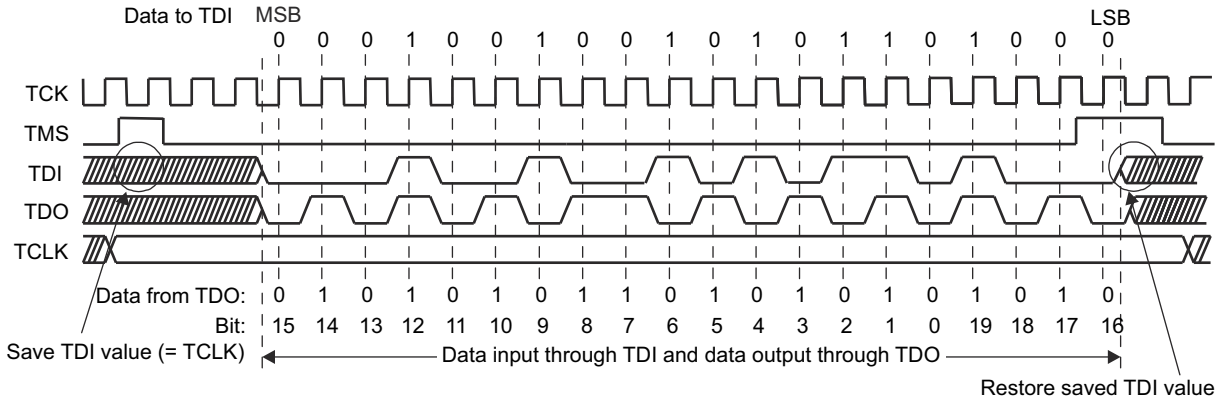

图 2-5. 地址寄存器 I/O：DR_SHIFT20 (0x12568)（TDO 输出为 0xA55AA）

2.2.2.1.4 MsDelay（时间）

这个宏使编程接口软件等待一个以毫秒 (ms) 为单位的指定时间长度。在这个宏执行时，进入和来自目标 MSP430 的所有信号必须保持它们之前的值。

2.2.2.1.5 SetTCLK

这个宏将 TCLK 输入时钟（在 TDI 信号输入上提供）置为高电平。在这个宏被执行时，TCK 和 TMS 必须保持它们之前最后的值（SBW 特定强制条件请参阅节 2.2.3.5.1 和图 2-11）。

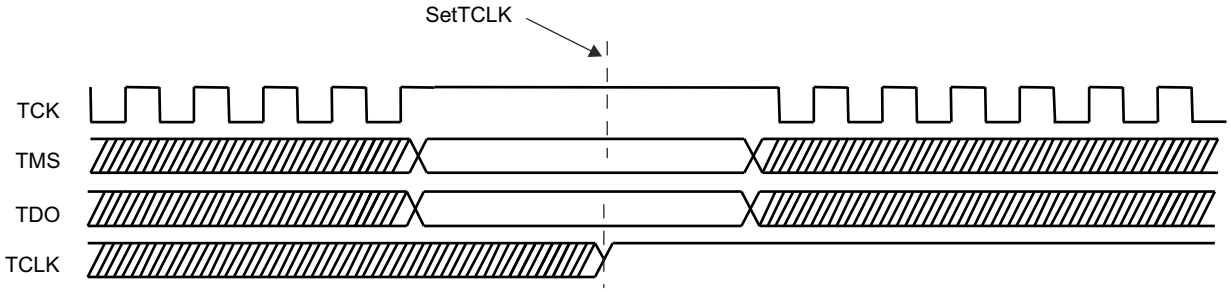

图 2-6. SetTCLK

2.2.2.1.6 ClrTCLK

这个宏将 TCLK 输入时钟复位为低电平。在这个宏被执行时，TCK 和 TMS 必须保持它们之前最后的值（SBW 特定强制条件请参阅节 2.2.3.5.1 和图 2-11）。

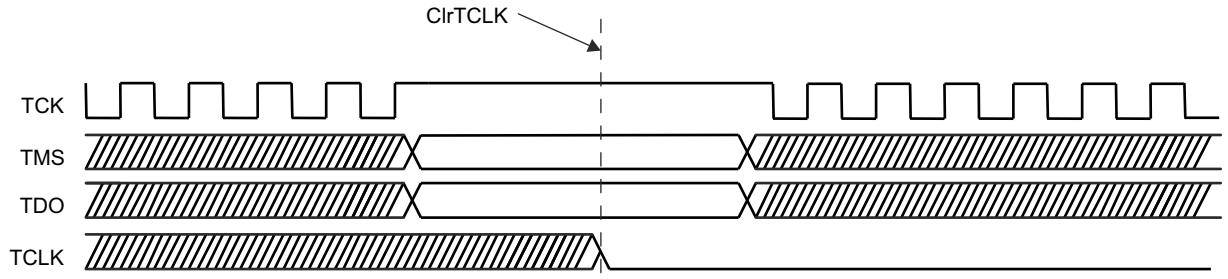

图 2-7. ClrTCLK

2.2.2.2 针对 Spy-Bi-Wire (SBW) 接口的宏

所有节 2.2.2.1 中描述的 JTAG 宏也适用于 2 线制接口并作为软件源代码与本文档一同提供。

2.2.3 Spy-Bi-Wire (SBW) 时序和控制

下面的部分对 SBW 执行的基本原则进行了说明，这是因为它与宏函数时序信号的生成相关。这是为了实现定制 MSP430 编程解决方案的开发，而不仅仅是依赖于提供的示例应用代码。

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

2.2.3.1 基本时序

SBW 接口串行通信使用时分多路复用功能，并分配三个时隙：TMS_SLOT、TDI_SLOT 和 TDO_SLOT。为了使用与 4 线制 JTAG 访问期间通过 TDI 为 TCLK 计时相似的方法通过 SBW 接口为 TCLK 计时，此处采用另一种 JTAG 时序方法。此实现方案利用了这样一个事实，即在 SBWTCK 的下降沿捕获 TDI 和 TMS 信号（分别在两种信号的时隙内），如图 2-8 所示。

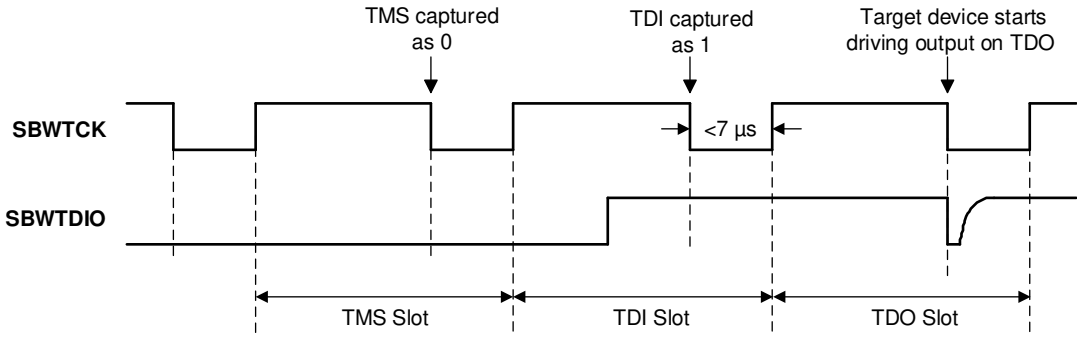

图 2-8. Spy-Bi-Wire 时序图

在 2 线制和 4 线制之间进行转换的逻辑如图 2-9 所示。

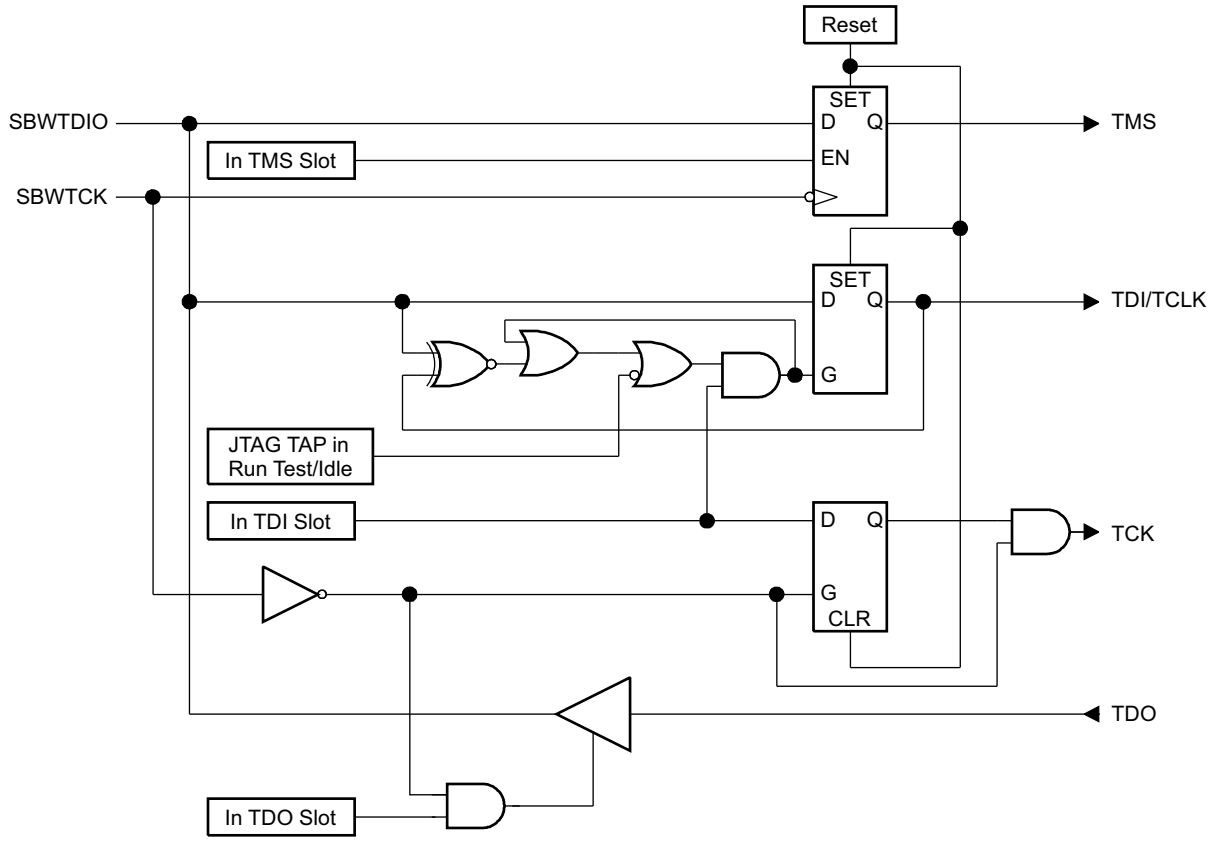

图 2-9. SBW 到 JTAG 接口图

这个实现方案的优势在于：

- TDI 和 TDO 上的数据对齐。
- 在 2 线制接口的 TDI_SLOT 期间，如果 JTAG TAP 控制器处于运行-测试/闲置状态，SBWTDIO 可被用作 TCLK 输入。为此，TDI 输出必须如图 2-11 显示的那样与其输入同步。同步逻辑只有在运行-测试/闲置状态时有效。

加电后，只要 SBW 接口还未被激活，TMS 和 TDI 就被内部设定为逻辑 1 电平。

Copyright © 2021 Texas Instruments Incorporated

2.2.3.2 TMS 时隙

TMS 时隙用于在目标器件 JTAG 模块的 TAP 控制器状态机中进行状态切换（请参阅节 2.1.2）。以下宏位于 Replicator 示例工程的 LowLevelFunc 头文件中。

2.2.3.2.1 TMSH 宏

将 TMS 时隙的 SBWTDIO 设置为高电平（无需进行特殊的 TDI 准备处理）

- 将 SBWTDIO 设置为高电平
- NOP 5 个周期（在 18MHz 下延迟）
- 将 SBWTCK 设置为低电平
- NOP 5 个周期（在 18MHz 下延迟）
- 将 SBWTCK 设置为高电平

2.2.3.2.2 TMSL 宏

将 TMS 时隙的 SBWTDIO 设置为低电平（无需进行特殊的 TDI 准备处理）

- 将 SBWTDIO 设置为低电平
- NOP 5 个周期（在 18MHz 下延迟）
- 将 SBWTCK 设置为低电平
- NOP 5 个周期（在 18MHz 下延迟）
- 将 SBWTCK 设置为高电平

2.2.3.2.3 TMSLDH 宏

将 TMS 时隙的 SBWTDIO 设置为低电平，但在 TMS 时隙结束之前的 SBWTCK 下降沿后将其恢复为高电平。用于 ClrTCLK 和 SetTCLK 的某些情况。

- 将 SBWTDIO 设置为低电平
- NOP 5 个周期（在 18MHz 下延迟）
- 将 SBWTCK 设置为低电平
- NOP 5 个周期（在 18MHz 下延迟）
- 将 SBWTDIO 设置为高电平
- 将 SBWTCK 设置为高电平

2.2.3.3 TDI 时隙

在大多数 TAP 控制器状态（例如 Shift-IR）下，TDI 用于将数据移入目标器件。在运行-测试/闲置状态下，TDI 时隙也可用于为目标 CPU 计时（请参阅节 2.2.3.5）。以下宏位于 Replicator 示例工程的 LowLevelFunc 头文件中。

2.2.3.3.1 TDIH 宏

将 TDI 时隙的 SBWTDIO 设置为高电平。

- 将 SBWTDIO 设置为高电平
- NOP 5 个周期（在 18MHz 下延迟）
- 将 SBWTCK 设置为低电平
- NOP 5 个周期（在 18MHz 下延迟）
- 将 SBWTCK 设置为高电平

2.2.3.3.2 TDIL 宏

将 TDI 时隙的 SBWTDIO 设置为低电平。

- 将 SBWTDIO 设置为低电平
- NOP 5 个周期（在 18MHz 下延迟）
- 将 SBWTCK 设置为低电平
- NOP 5 个周期（在 18MHz 下延迟）
- 将 SBWTCK 设置为高电平

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

2.2.3.4 TDO 时隙

如图 2-8 所示，为 TDO 操作分配了一个时隙（也请参阅图 2-10 中的详细时序）。主器件应根据 TDI 周期中 SBWTCK 的上升沿来释放对 SBWTDIO 线路的控制权。在主器件释放 SBWTDIO 线路后，一个内部总线保持器负责保持线路上的电压。SBWTCK 的下一个下降沿将触发从器件开始驱动总线。从器件仅在 SBWTCK 周期中的低电平时间内驱动 SBWTDIO 线路。在从器件释放 SBWTDIO 线路之前，主器件不应启用其驱动程序。因此，主器件可以将 SBWTCK 的上升沿作为启用其驱动程序的触发点。

NOTE

在 SBWTCK 上提供的时钟信号低电平的持续时间一定不能超过 7μs。如果低电平持续更长的时间，SBW 逻辑将被置为无效，并且必须按照节 2.3.1 将其重新激活。

当使用提供的源代码示例时，请确保中断在 SBWTCK 低电平时被禁用以保证时序准确。

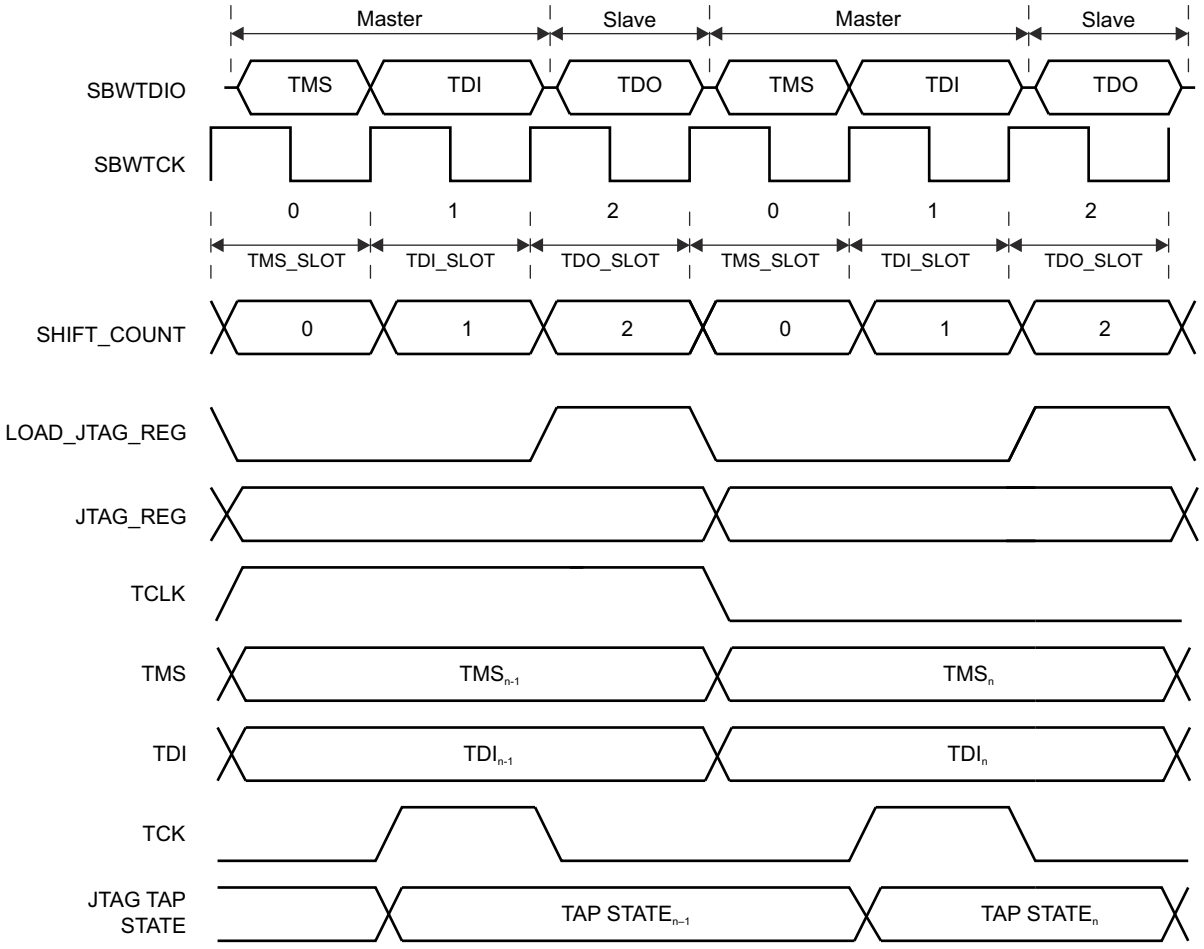

图 2-10. 详细 SBW 时序图

以下宏位于 Replicator 示例工程的 LowLevelFunc 头文件中。

2.2.3.4.1 TDO_RD 宏

`TRSLDIR |= TDOI_DIR` 将板上的 SN74LVC1T45 收发器设置为高阻态，而 `TRSLDR &= TDOI_DIR` 将其设置回被驱动。

用于在 TDO 时隙期间读取 TDO 值。

- 将 SBWTDIO 设置为高阻态
- NOP 5 个周期（在 18MHz 下延迟）
- 将 SBWTCK 设置为低电平

Copyright © 2021 Texas Instruments Incorporated

- NOP 5 个周期 (在 18MHz 下延迟)
- 读取 SBWTDIO 线路
- NOP 5 个周期 (在 18MHz 下延迟)
- 将 SBWTCK 设置为高电平
- 再次将 SBWTDIO 设置为被驱动

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

2.2.3.4.2 TDOsbw 宏（不读取）

当不需要读取时，用于通过 TDO 时隙计时。

- 将 SBWTDIO 设置为高阻态
- NOP 5 个周期（在 18MHz 下延迟）
- 将 SBWTCK 设置为低电平
- NOP 5 个周期（在 18MHz 下延迟）
- 将 SBWTCK 设置为高电平
- 再次将 SBWTDIO 设置为被驱动

2.2.3.5 Spy-Bi-Wire (SBW) 模式下的 TCLK 处理

2.2.3.5.1 SetTCLK 和 ClrTCLK

如果 JTAG TAP 控制器处于运行-测试/闲置状态，TDI 时隙可以提供 TCLK 信号（也就是说，可以为目标 CPU 计时）。在此实现之后，整个 TCLK 时钟周期的生成需要两个 TDI 时隙，其中一个时隙设置 TCLK 信号，而另外一个将其清除。在每种情况下，SBWTDIO 信号在 TMS 时隙内被设定为低电平，以将 TAP 控制器保持在运行-测试/闲置状态。为了只为 ClrTCLK 提供一个下降沿，SBWTDIO 信号必须在进入 TDI 时隙前被设定为高电平。相应的上升沿必须出现在 TMS 时隙中 SBWTCK 的低电平相位。否则，它将被认为是一个 TMS=1 的触发边沿，而且 TAP 控制器将退出运行-测试/闲置模式。

图 2-11 显示了处理 SBW 模式中的 TCLK。有关软件实现情况，请参阅 MSP430 Replicator 工程 (slau320.zip) 中的参考函数 SetTCLK_sbw 和 ClrTCLK_sbw。提供的针对 MSP430Xv2 架构的代码示例使用预处理器定义来实现更佳的分层式软件架构。上部软件层可以只参考 SetTCLK 和 ClrTCLK 符号，而针对 4 线制 JTAG 的实际执行符号为 SetTCLK_4wire 和 ClrTCLK_4wire，针对 Spy-Bi-Wire (SBW) 的实际执行符号为 SetTCLK_sbw 和 ClrTCLK_sbw。

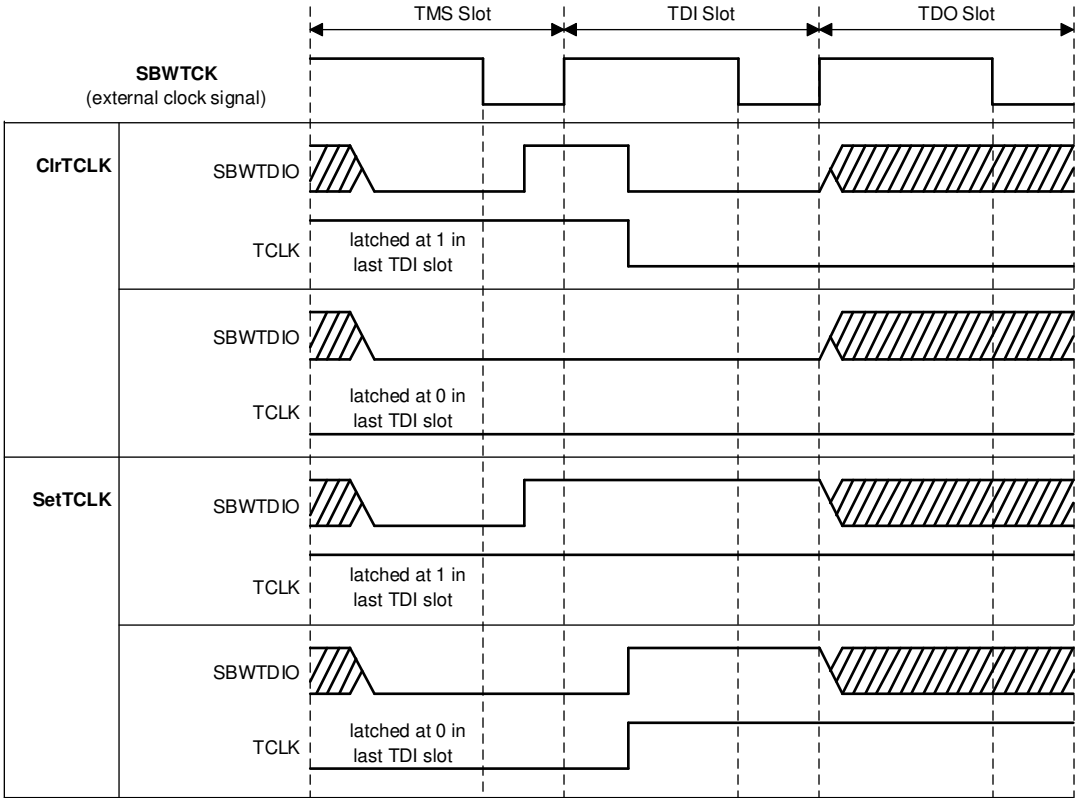

图 2-11. 运行-测试/闲置期间的 TDI 和 TCLK 同步

Copyright © 2021 Texas Instruments Incorporated

2.2.3.5.2 TCLK 选通信号

对于 F1xx、F2xx、G2xx 和 F4xx 系列的 MSP430 器件，可以在单个 TDI 时隙内提供自定义数量的 TCLK 时钟（请参阅示例工程 Replicator430 和 Replicator430X）。请参阅参考函数：TCLKstrobes()。此实现方案不适用于 F5xx 或 F6xx 器件。对于这些器件，闪存时序由内部 MODOSC 生成。

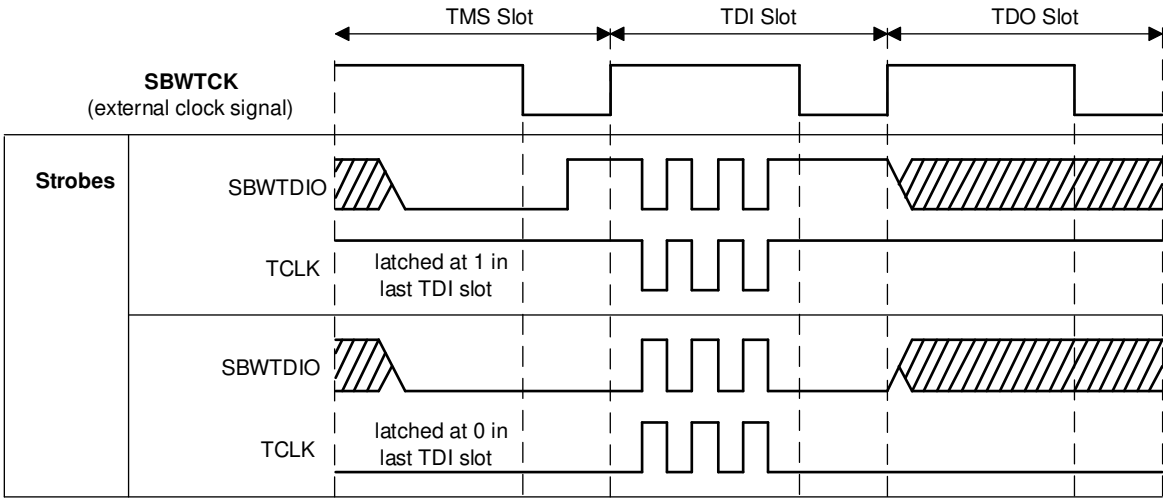

图 2-12. 使用 TCLK 选通信号进行 TCLK 计时

2.2.4 JTAG 通信指令

通过使用节 2.2.2.1.1 中描述的 IR_SHIFT 宏来移入一个 JTAG 指令，可实现对 JTAG 寄存器的选择和对 CPU 的控制。下面列出的可被写入 JTAG IR 的指令可用于对目标存储器进行编程。所有通过 JTAG 寄存器发送到目标 MSP430 的指令都按照 LSB 在前的方式传输。

表 2-5. 存储器访问指令

指令名称	8 位指令值
控制存储器地址总线 (MAB)	
IR_ADDR_16BIT	0x83
IR_ADDR_CAPTURE	0x84
控制存储器数据总线 (MDB)	
IR_DATA_TO_ADDR	0x85
IR_DATA_16BIT	0x41
IR_DATA_QUICK	0x43
IR_BYPASS	0xFF
控制 CPU	
IR_CNTRL_SIG_16BIT	0x13
IR_CNTRL_SIG_CAPTURE	0x14
IR_CNTRL_SIG_RELEASE	0x15
通过伪签名分析 (PSA) 进行存储器校验	
IR_DATA_PSA	0x44
IR_SHIFT_OUT_PSA	0x46
JTAG 访问安全保险丝编程	
IR_Prepare_Blow	0x22
IR_Ex_Blow	0x24
JTAG 邮箱系统	
IR_JMB_EXCHANGE	0x61

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

NOTE

不要将任何未列出的值写入到 JTAG 指令寄存器中。将上面未列出的指令值写入 MSP430 JTAG 寄存器会导致意外的器件运行状态。

NOTE

当一个新的 JTAG 指令被移入 JTAG 指令寄存器时，它在 TAP 控制器的 UPDATE-IR 状态时生效。当访问一个 JTAG 数据库时，最后一个值在 CAPTURE-DR 状态时被写入，而写入的新值在 UPDATE-DR 状态时变为有效。换而言之，无需遍历 JTAG TAP 控制器的运行-测试/闲置状态即可实现指令或者数据的移入。需要注意的一个事实是只有在运行-测试/闲置状态时才可为 TCLK 计时。这就是为什么所提供的软件示例应用专门使用节 2.2.2 中描述的 JTAG 宏，这个 JTAG 宏遍历运行-测试/闲置状态。

2.2.4.1 控制内存地址总线(MAB)

下面的指令用于控制目标 MSP430 的 MAB。为了完成这一操作，对一个被称为 JTAG MAB 寄存器的 16 位（或者在 MSP430X 架构中为 20 位）寄存器寻址。通过使用 TAP 控制器的 JTAG 数据路径，这个寄存器可被访问和修改。

2.2.4.1.1 IR_ADDR_16BIT

这个指令将 MAB 设置为一个特定的值，这个值随使用 DR_SHIFT16（16 位数据）宏的下一个 JTAG 16 位数据访问移入，或者随使用 DR_SHIFT（20 位地址）宏的下一个 JTAG20 位地址字访问移入。MSP430 CPU 的 MAB 被设定为写入 JTAG MAB 寄存器的值。在通过 TDI 移入新的 16 位或 20 位地址的同时，先前存储在 JTAG MAB 寄存器中的值在 TDO 上移出。

NOTE

在 MSP430X 器件中，用于升级 JTAG MAB 寄存器的 16 位移位并不会自动复位 JTAG MAB 寄存器的上四位 (19:16)。一直使用 20 位移位宏以确保上四位 (19:16) 被设定为一个已定义的值。

2.2.4.1.2 IR_ADDR_CAPTURE

这个指令能够使用下一个 16 或 20 位数据访问来读出 MAB 上的数据。在 16 或 20 位数据访问时，MAB 值并不发生变化；也就是说，用这个命令从 TDI 上发出的 16 或 20 位数据会被忽略（在提供的软件中，0 作为缺省值被发出）。

在整个示例代码中的几个地方，IR_ADDR_CAPTURE 指令还用于在使用 IR_DATA_16BIT 指令访问数据总线后将 CPU 设置为定义的状态。

2.2.4.2 控制存储器数据总线(MDB)

下面的指令控制 MSP430 CPU 的 MDB。为了完成这一操作，一个被称为 JTAG MDB 寄存器的 16 位寄存器被寻址。通过使用 TAP 控制器的 JTAG 数据路径，这个寄存器可被访问和修改。

2.2.4.2.1 IR_DATA_TO_ADDR

这个指令将 MSP430 MDB 设置为一个特定的值，这个值与使用 DR_SHIFT16（16 位数据）宏的下一个 JTAG 16 位数据访问一同移入。MSP430 CPU 的 MDB 设置为写入 JTAG MDB 寄存器的值。当新值被写入 MDB 寄存器中时，MSP430 MDB 中之前的值被捕捉并且移出 TDO。在 IR_DATA_TO_ADDR 指令执行期间，MSP430 MAB 由 JTAG MAB 寄存器中的值设定。这个指令用于写入 MSP430 的所有内存位置。

2.2.4.2.2 IR_DATA_16BIT

这个指令将 MSP430 MDB 设置为一个特定的 16 位值，这个值随下一个 16 位 JTAG 数据访问移入。完整的 MSP430 MDB 被设定为 JTAG MDB 寄存器的值。同时，MSP430 MDB 的最后一个值被捕捉并在 TDO 上被移出。在这个情况下，MAB 仍由 CPU 控制。目标 CPU 的程序计数器 (PC)设定 MAB 值。

Copyright © 2021 Texas Instruments Incorporated

2.2.4.2.3 IR_DATA_QUICK

这个指令将 MSP430 MDB 设置为一个特定的值，这个值随着下一个 16 位 JTAG 数据访问移入。16 位 MSP430 MDB 被设定为写入 JTAG MDB 寄存器的值。在 16 位数据传递期间，之前的 MDB 值被捕捉并在 TDO 上被移出。MAB 值由 CPU 的程序计数器 (PC) 设定。这个指令在 TCLK 的每个下降边沿上将程序计数器加 2 来自动指向下一个 16 位内存位置。在执行这条指令之前，目标 CPU 的程序计数器必须加载起始存储器地址，这样可以快速读取或写入存储器阵列（更多有关设置 PC 的信息，请参阅节 2.3.2.1.3）。

NOTE

不能使用 IR_DATA_QUICK 来写入闪存。

NOTE

不能使用 IR_DATA_QUICK 来读取或写入 USB RAM，因为这是双端口 RAM。需要对其进行逐字访问。

2.2.4.2.4 IR_BYPASS

这个指令将输入传递到 TDI，作为 TDO 上被一个 TCK 时钟时钟周期延迟的输出。当这个指令被载入时，IR_CNTRL_SIG_RELEASE 指令（请见节 2.2.4.3.3）同时执行。执行旁路指令后，在 TDI 上移出的 16 位数据不会影响目标 MSP430 器件 JTAG 控制模块的任何寄存器。

Copyright © 2021 Texas Instruments Incorporated

2.2.4.3 控制 CPU

以下指令通过一个由 JTAG 访问的 16 位寄存器来实现对 MSP430 CPU 的控制。此数据寄存器被称为 JTAG 控制信号寄存器。表 2-6 描述了组成 JTAG 控制信号寄存器的位功能，此寄存器用于内存访问。

表 2-6. 针对 1xx、2xx、4xx 系列的 JTAG 控制信号寄存器

位编号	名称	说明
0	R/W	控制 CPU 的读取/写入 (RW) 信号 1 = 读取 0 = 写入
1	(无)	一直写入 0
2	(无)	一直写入 0
3	HALT_JTAG	将 CPU 设定为受控的暂停状态 1 = CPU 停止 0 = CPU 正常运行
4	BYTE	控制 CPU 中用于存储器访问数据长度的 BYTE 信号 1 = 字节 (8 位) 访问 0 = 字 (16 位) 访问
5	(无)	一直写入 0
6	(无)	一直写入 0
7	INSTR_LOAD	只读：表示目标 CPU 指令状态 1 = 取指令状态 0 = 指令执行状态
8	(无)	一直写入 0
9	TCE	表示 CPU 同步 1 = 已同步 0 = 未同步
10	TCE1	建立 JTAG 对 CPU 的控制 1 = CPU 受 JTAG 控制 0 = CPU 自由运行
11	POR	控制上电复位 (POR) 信号 1 = 执行 POR 0 = 未复位
12	释放低字节	选择 RW 和 BYTE 位的控制源 1 = CPU 有控制权 0 = 控制信号寄存器有控制权
13	TAGFUNCSAT	将闪存模块设定为 JTAG 访问模式 1 = CPU 有控制权 (默认值) 0 = JTAG 有控制权
14	SWITCH	启用 TDO 输出为 TDI 输入 1 = JTAG 有控制权 0 = 正常运行
15	(无)	一直写入 0

Copyright © 2021 Texas Instruments Incorporated

表 2-7. 针对 5xx 和 6xx 系列产品的 JTAG 控制信号寄存器

位编号	名称	说明
0	R/W	控制 CPU 的读取/写入 (RW) 信号，与之前系列一样。 1 = 读取 0 = 写入
1	(无)	一直写入 0，与之前系列一样。
2	(无)	一直写入 0，与之前系列一样。
3	WAIT	到 CPU 的等待信号。只读。 1 = CPU 时钟停止 - 等待一个操作完成 0 = CPU 时钟未停止
4	BYTE	控制 CPU 中用于内存访问数据长度的 BYTE 信号，与之前系列一样。 1 = 字节 (8 位) 访问 0 = 字 (16 位) 访问
5	(无)	一直写入 0
6	(无)	一直写入 0
7	INSTR_LOAD	只读：表示目标 CPU 指令状态。真实状态与之前系列不一样。 1 = 取指令状态 0 = 指令执行状态
8	CPUSUSP	挂起 CPU。 通过声明 CPUSUSP 位为有效位并驱动完成最长指令所需的最少时钟来清空 CPU 流水线。当 CPUSUSP 为高电平时，不取或者执行指令。为了通过 JTAG 执行一个强制的外部指令序列 (例如，设定程序计数器) ，CPUSUSP 位必须为零。 0 = CPU 激活 1 = CPU 挂起 读取 CPUSUSP (位 8) 可以显示流水线是否为空： 0 = 流水线尚不为空 1 = 流水线为空
9	TCE0	表示 CPU 同步，与之前系列一样。 1 = 已同步 0 = 未同步
10	TCE1	建立 JTAG 对 CPU 的控制，与之前系列一样。 1 = CPU 受 JTAG 控制 0 = CPU 自由运行
11	POR	控制上电复位 (POR) 信号，与之前系列一样。 1 = 执行 POR 0 = 未复位
12	RELEASE_LBYTE0	从 JTAG 控制中释放低字节中的控制位。 00 = 如果 TCE1 为 1，所有位由 JTAG 控制 01 = RW (位 0) 和 BYTE (位 4) 从 JTAG 控制中释放 10 = RW (位 0)、HALT (位 1)、INTREQ (位 2) 和 BYTE (位 4) 从 JTAG 控制中释放 11 = 保留
13	RELEASE_LBYTE1	
14	INSTR_SEQ_NO0	指令序列编号。只读。 显示了当前使用 CPU 总线的流水线型 CPU 的指令序列号 (在流水线中最多有三个指令)。 00 = CPU 指令序列 0 01 = CPU 指令序列 1 10 = CPU 指令序列 2 11 = CPU 生成的“无操作”周期；总线上的数据未使用
15	INSTR_SEQ_NO1	

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

2.2.4.3.1 IR_CNTRL_SIG_16BIT

这个指令用下一个 16 位 JTAG 数据访问来启用完整 JTAG 控制信号寄存器的设置。同时，存储在寄存器中的最后一个值在 TDO 上被移出。当 TAP 控制器进入 UPDATE-DR 状态时，新值生效。

2.2.4.3.2 IR_CNTRL_SIG_CAPTURE

这个指令用下一个 JTAG 16 位数据访问指令来读出 JTAG 控制信号寄存器的值。

2.2.4.3.3 IR_CNTRL_SIG_RELEASE

这个指令将 CPU 从 JTAG 控制中完全释放出来。一旦被执行，JTAG 控制信号寄存器和其它 JTAG 数据寄存器不再对目标 MSP430 CPU 产生任何影响。这个指令通常用于将 CPU 从 JTAG 控制中释放出来。

2.2.4.4 通过伪签名分析 (PSA) 进行存储器校验

下面的指令借助一个 PSA 模式来支持对 MSP430 存储器内容的校验。

2.2.4.4.1 IR_DATA_PSA

IR_DATA_PSA 指令将 JTAG_DATA_REG 切换到 PSA 模式。在这个模式中，MSP430 的程序计数器按每两个 TCLK 上提供的系统时钟递增。在执行此指令之前，CPU 程序计数器必须加载起始地址。TCLK 时钟的数量确定了有多少内存地址被包含在 PSA 计算中。

2.2.4.4.2 IR_SHIFT_OUT_PSA

IR_SHIFT_OUT_PSA 指令应与 IR_DATA_PSA 指令一同使用。这个指令移出由 IR_DATA_PSA 命令生成的 PSA 模式。在 TAP 控制器的 SHIFT-DR 状态期间，JTAG_DATA_REG 的内容通过 TDO 引脚移出。在执行这个 JTAG 指令时，JTAG_DATA_REG 的捕捉和升级功能被禁用。

2.2.4.5 JTAG 访问安全保险丝编程

下列指令用于访问内置 JTAG 访问保护保险丝并对其进行编程，这些保险丝在 MSP430F1xx、2xx，和 4xx 闪存器件上提供。对保险丝进行编程（或熔断）时，未来通过 JTAG 接口对 MSP430 进行的访问会被永久禁用。这样可实现对已编入目标器件的最终 MSP430 固件的访问保护。这些指令在 MSP430F5xx 和 F6xx 器件上不可用。一个不同的基于软件的机制被用于这些系列以实现 JTAG 访问保护（详情请见节 2.4）。

2.2.4.5.1 IR_PREPARE_BLOW

这个指令将 MSP430 设定为程序保险丝模式。

2.2.4.5.2 IR_EX_BLOW

这个指令用于对访问保护熔丝进行编程（熔断）。为了正常执行，它必须在 IR_PREPARE_BLOW 指令被指定后载入。

Copyright © 2021 Texas Instruments Incorporated

2.3 内存编程控制序列

2.3.1 启动

在可以开始主存储器编程例程前，目标器件必须为编程进行初始化。这个部分描述了如何执行此初始化序列。

2.3.1.1 启用 JTAG 访问

只有与端口 I/O 共享 JTAG 引脚（这取决于是否存在 TEST 引脚）的器件才需要执行此步骤。所有支持 Spy-Bi-Wire 协议的器件都是这种情况。此外，一些较旧的器件组需要经过特殊处理才能启用 4 线制 JTAG（请参阅表 2-14 中的“TEST 引脚”列）。

参考函数：GetDevice、GetDevice_sbw、GetDevice_430X、GetDevice_430Xv2

- 只具有 TEST 引脚和 4 线制 JTAG 访问（无 SBW）的 MSP430 器件

  为了使用具有共用 JTAG 和一个 TEST 引脚的 MSP430 器件的 JTAG 特性，需要启用针对 JTAG 通信模式的共用 JTAG 引脚。具有专用 JTAG 输入/输出和没有 TEST 引脚的器件无需此步骤。 通过将 TEST 引脚连接到 V_{CC} 来启用用于 JTAG 通信的共用引脚。在正常运行（非 JTAG 模式）时，应释放此引脚，从而通过内部下拉电阻器将它拉至接地。表 2-8 显示了用于 JTAG 通信的端口 1 引脚。

表 2-8. 共用 JTAG 器件引脚功能

端口 1 功能 （TEST = 开）	JTAG 函数 (TEST = V_{CC})
P1.4	TCK
P1.5	TMS
P1.6	TDI/TCLK
P1.7	TDO

- 具有 Spy-Bi-Wire (SBW) 访问的 MSP430 器件

  TEST/SBWTCK 引脚被保持在低电平时，SBW 接口和到 JTAG 接口的任一访问被禁用。这通过一个内部下拉电阻器来完成。这个引脚也可从外部接至低电平。

  将 TEST/SBWTCK 引脚拉至高电平以启用 SBW 接口并且禁用 $\overline{\text{RST}}$/NMI/SBWTDIO 引脚的 $\overline{\text{RST}}$/NMI 功能。当 SBW 接口被激活时，内部复位信号被保持在高电平，并且内部 NMI 信号被保持在输入值（TEST/SBWTCK 变为高电平时 $\overline{\text{RST}}$/NMI 上的值）上。

  具有 SBW 的器件也支持标准 4 线制接口。通过将 SBWTDIO 线路下拉至低电平，然后将一个时钟应用到 SBWTCK 上来启用 4 线制 JTAG 接口访问。通过将 TEST/SBWTCK 低电平保持 100µs 来退出 4 线制 JTAG 模式。

  为了选择 2 线制 SBW 模式，SBWTDIO 线路被保持在高电平并且第一个时钟被应用到 SBWTCK 上。这个时钟之后，以 TMS 时隙开始的正常 SBW 时序被采用，并且可采用正常 JTAG 模式，通常以测试访问端口 (Tap) 复位和保险丝检查序列作为开始。通过保持 TEST/SBWTCK 低电平 100µs 来退出 SBW 模式。

  在采用引导加载程序 (BSL) 的器件中，TEST/SBWTCK 和 $\overline{\text{RST}}$/NMI/SBWTDIO 也用于调用 BSL。图 2-13 显示了用于进入 SBW/JTAG 或者 BSL 模式的不同情况。

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

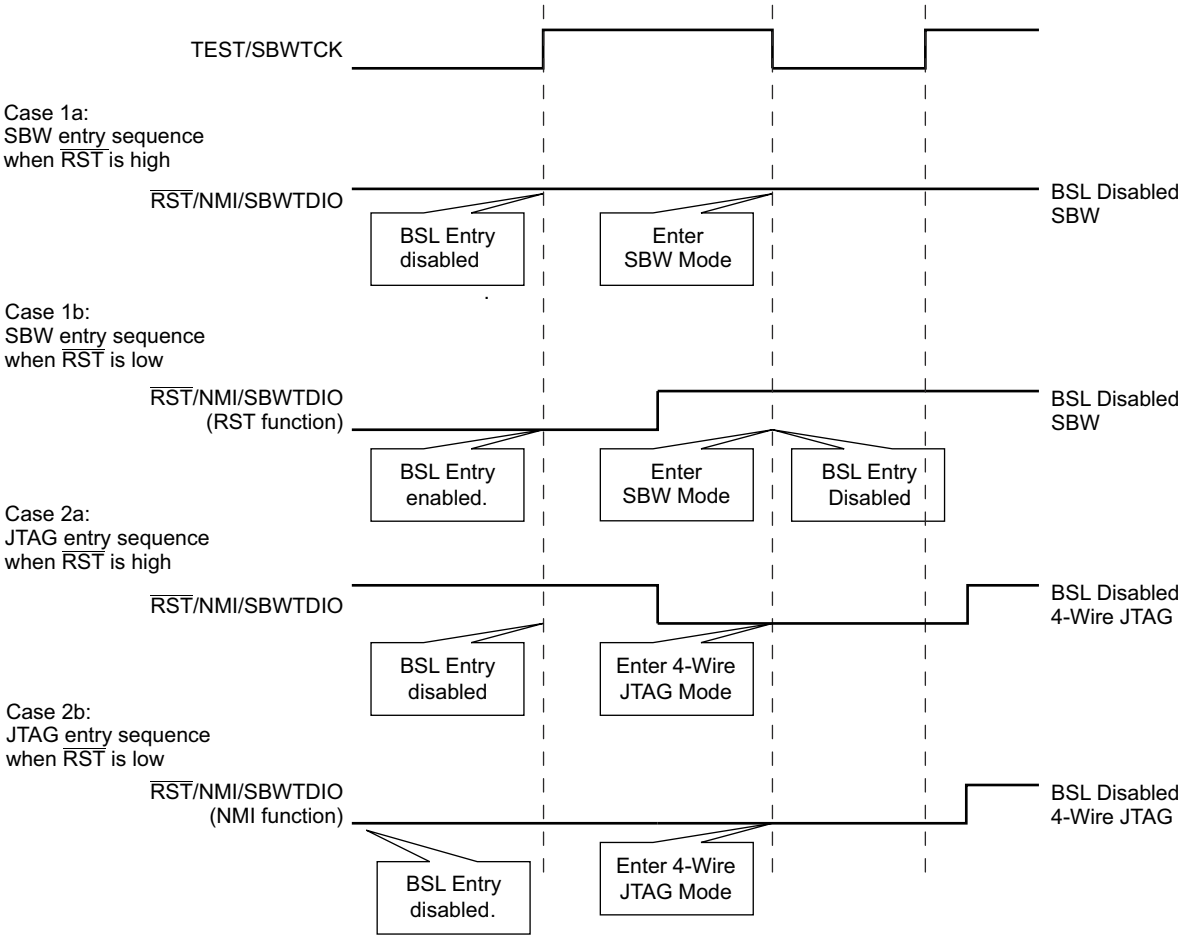

图 2-13. JTAG 访问进入序列（针对支持 SBW 的器件）

NOTE

在一些支持 Spy-Bi-Wire 的 MSP430 器件上，TEST/SBWTCK 对上升边沿十分敏感，这会导致测试逻辑进入一个状态，在这个状态中进入序列（2 线制或者 4 线制）不能被正确识别并且 JTAG 访问一直被禁用。当 JTAG 连接器被正确连接到目标器件时，SBWTCK 上的单向边沿会出现。有两个可能的权变措施来解决这个问题并且确保一个稳定的 JTAG 访问初始化。

- 在为器件加电或者插入连接器前，将 SBWTCK 驱动为低电平来避免单向上升信号边沿。
- 多次运行初始化序列（重复两次或者三次通常足够建立一个稳定的连接）。

Copyright © 2021 Texas Instruments Incorporated

2.3.1.2 保险丝检查和 JTAG 状态机（TAP 控制器）的复位

参考函数：ResetTAP、ResetTAP_sbw

每个 MSP430F1xx、2xx 和 4xx 闪存器件均包含物理保险丝，此保险丝用于永久禁用通过 JTAG 通信进行的存储器访问。此保险丝经过编程（或熔断）后，通过 JTAG 进行的存储器访问被永久禁用并且不能恢复。当在加电后初始化 JTAG 访问时，在允许 JTAG 访问前，必须完成保险丝检查。两次切换 TMS 信号来执行此检查。

在保险丝被检查时，高达 2mA 的电流流入 TDI 输入（或者进入不具有专用 JTAG 引脚的器件上的 TEST 引脚内）。为了实现电流稳定，两个 TMS 脉冲的低电平阶段应该最少持续 5µs。

在特定情况下（例如，插入一个电池），在 TDI 为逻辑低电平时，TMS 的一个切换有可能意外发生。在那个情况下，没有电流流过安全保险丝，但是内部逻辑认识到已经执行了一个保险丝检查。因此，此保险丝被错误地识别为已被编程（即，熔断）。为了避免这一事件的发生，更新一代的 MSP430 JTAG 工具（具有 CPUXv2 的器件-请见表 2-15）还在执行 TAP 控制器复位时将内部保险丝检查逻辑复位。因此，如图 2-14 所示，建议首先执行一个 TAP 的复位，然后检查 JTAG 保险丝状态。为了执行 TAP 控制器的复位，建议在 TMS 为高电平时向目标器件发送最少 6 个 TCK 时钟周期，之后在至少一个 TCK 时钟周期内将 TMS 设定为低电平。这将把 JTAG 状态机（TAP 控制器）设定为一个已定义的起始点：运行-测试/闲置状态。这个步骤也可在 JTAG 通信期间用来将 JTAG 端口复位。

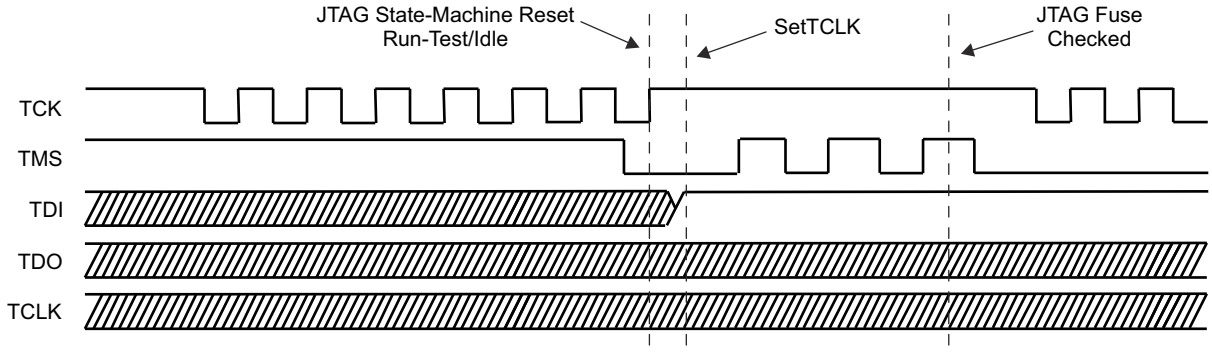

图 2-14. 保险丝检查和 TAP 控制器复位

执行保险丝检查时，在 SBW 模式下采用同样的步骤会产生改变 TAP 控制器状态的副作用。如节 2.2.3.1 中所描述的那样，在每个 TDI_SLOT 内自动生成内部信号 TCK。在 SBW 模式下执行一个保险丝检查，在 TAP 控制器复位后立即开始，在它的 Exit2-DR 状态中结束。必须再生成两个虚拟 TCK 周期才能恢复到运行-测试/闲置状态：一个 TCK 的 SBWTDIO 在 TMS_SLOT 期间处于高电平，接着一个 TCK 的 SBWTDIO 在 TMS_SLOT 期间处于低电平（参考函数：ResetTAP_sbw）。

NOTE

MSP430F5xx 和 F6xx 系列无需专用的保险丝检查序列（切换 TMS 两次）。这些系列执行一个软件机制而非一个硬件保险丝（此保险丝需要被检查或者燃烧）来启用 JTAG 安全保护。

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

2.3.2 通用器件 (CPU) 控制功能

这个部分描述的功能用于目标 MSP430 CPU 的普通控制，以及高级 JTAG 访问和总线控制。

2.3.2.1 针对 1xx，2xx，4xx 系列的函数参考

2.3.2.1.1 将 CPU 置于 JTAG 控制之下

参考函数：GetDevice、GetDevice_sbw、GetDevice_430X

在完成对保险丝的初始检查和复位后，目标器件的 CPU 必须置于 JTAG 控制之下。通过将 JTAG 控制信号寄存器的位 10 设定为 1 来完成。此后，CPU 需要一些时间来与 JTAG 控制同步。为了检查是否 CPU 已同步，会测试位 9 (TCE)（如果设定为 1 表示同步成功）。验证此位为高电平后，CPU 受 JTAG 接口控制。随后是将目标器件置于 JTAG 控制之下的流程。

IR_SHIFT("IR_CNTRL_SIG_16BIT")	
DR_SHIFT16(0x2401)	
IR_SHIFT("IR_CNTRL_SIG_CAPTURE")	
DR_SHIFT16(0x0000)	**否**
TDOword 的 位 9 = 1？	
支持	
CPU 受 JTAG 控制	

2.3.2.1.2 将 CPU 设定为取指令

参考函数：SetInstrFetch

对于目标器件，有时在 JTAG 端口上直接执行一个由主机提出的指令会很有用。为了完成这一操作，CPU 必须被设定为指令取状态。使用此设置，目标器件 CPU 会像正常情况下一样加载和执行指令，但该指令是通过 JTAG 传输的。JTAG 控制信号寄存器的位 7 表示 CPU 处于取指令状态。TCLK 应该在这个位为零时被切换。在最大 7 个 TCLK 时钟周期后，CPU 应该处于取指令状态。如果不是该状态（位 7 = 1），则表示发生了 JTAG 访问错误并建议进行 JTAG 复位。

IR_SHIFT("IR_CNTRL_SIG_CAPTURE")	
DR_SHIFT16(0x0000) = Readout data	
TDOvalue 的位 7 = 0？	
ClrTCLK	
SetTCLK	
CPU 处于取指令状态	

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

2.3.2.1.3 设置目标 CPU 程序计数器 (PC)

为了使用 MSP430 提供的 JTAG 接口特性，要求设置目标器件的 CPU PC。下面的流程用于完成这一操作。针对 MSP430 和 MSP430X 架构的执行显示如下。

- MSP430 架构：参考函数：SetPC

在以下序列之前，CPU 必须处于取指令状态。
IR_SHIFT("IR_CNTRL_SIG_16BIT")
DR_SHIFT16(0x3401)：释放低字节
IR_SHIFT("IR_DATA_16BIT")
DR_SHIFT16(0x4030)：PC 加载指令
ClrTCLK
SetTCLK
DR_SHIFT16("PC_Value")：为 PC 插入值
ClrTCLK
SetTCLK
IR_SHIFT("IR_ADDR_CAPTURE")：禁用 IR_DATA_16BIT
ClrTCLK：现在 PC 被设置为“PC_Value”
IR_SHIFT("IR_CNTRL_SIG_16BIT")
DR_SHIFT16(0x2401)：JTAG 控制的低字节
加载 PC 完成

- MSP430X 架构：参考函数：SetPC_430X

在以下序列之前，CPU 必须处于取指令状态。
IR_SHIFT("IR_CNTRL_SIG_16BIT")
DR_SHIFT16(0x3401)：释放低字节
IR_SHIFT("IR_DATA_16BIT")
DR_SHIFT16(0x0X80)：PC 加载指令，X = PC(19:16)
ClrTCLK
SetTCLK
DR_SHIFT16("PC(15:0)")：为 PC(15:0) 插入值
ClrTCLK
SetTCLK
IR_SHIFT("IR_ADDR_CAPTURE")：禁用 IR_DATA_16BIT
ClrTCLK：现在 PC 被设置为“PC_Value”
IR_SHIFT("IR_CNTRL_SIG_16BIT")
DR_SHIFT16(0x2401)：JTAG 控制的低字节
加载 PC 完成

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

2.3.2.1.4 目标 CPU 的受控停止或启动

参考函数：HaltCPU/ReleaseCPU

在 JTAG 接口访问一个存储器位置时，目标器件的 CPU 应该被置入一个已定义的暂停状态。CPU 的停止由 JTAG 控制信号寄存器的 HALT_JTAG 位（位 3）支持，在 HaltCPU 函数执行时该位被设定为 1。在访问所需的存储器位置之后，CPU 可恢复正常运行。此函数由 ReleaseCPU 原型实现，并简单地将 HALT_JTAG 位复位。

在执行以下序列之前，CPU 必须处于取指令状态	
HaltCPU	IR_SHIFT("IR_DATA_16BIT")
	DR_SHIFT16(0x3FFF)：“JMP $”指令用于防止 CPU 改变状态
	ClrTCLK
	IR_SHIFT("IR_CNTRL_SIG_16BIT")
	DR_SHIFT16(0x2409)：设置 HALT_JTAG 位
	SetTCLK
现在 CPU 处于受控状态并且在访问存储器期间不会被改变。 *注意：在访问目标存储器时不要将 HALT_JTAG 位复位 (=0)。*	
在此执行存储器访问	
使用 ReleaseCPU 将 CPU 切换回正常运行状态。	
ReleaseCPU	ClrTCLK
	IR_SHIFT("IR_CNTRL_SIG_16BIT")
	DR_SHIFT16(0x2401)：清除 HALT_JTAG 位
	IR_SHIFT("IR_ADDR_CAPTURE")
	SetTCLK
CPU 现在处于取指令状态并且已为接收新的 JTAG 指令做好准备。如果在访问存储器时更改了 PC，则必须将正确的地址载入到 PC 中。	

Copyright © 2021 Texas Instruments Incorporated

2.3.2.1.5 在受 JTAG 控制时复位 CPU

参考函数：ExecutePOR

有时需要在受 JTAG 控制时复位目标器件。建议在编辑或者擦除目标器件的闪存存储器前执行复位。当一个复位已经被执行，目标 CPU 的状态与一个实际器件加电后的状态等效。下面的流程用于强制执行一个上电复位。

IR_SHIFT("IR_CNTRL_SIG_16BIT")
DR_SHIFT16(0x2C01) : 应用复位
DR_SHIFT16(0x2401) : 删除复位
ClrTCLK
SetTCLK
ClrTCLK
SetTCLK
ClrTCLK
IR_SHIFT("IR_ADDR_CAPTURE")
SetTCLK
目标 CPU 现在会复位；PC 指向用户程序的起始地址，此地址是存储在复位矢量存储器位置 0xFFFEh 内的数据所指向的地址，并且所有寄存器均设置为它们各自的上电值。
为避免意外复位，现在必须通过将 0x5A80 写入目标器件的看门狗计时器控制寄存器来禁用器件的看门狗计时器。WriteMem() 可用于此寄存器访问。
除非使用了 WriteMem() 来停止看门狗计时器，否则现在需要调用 ReleaseCPU()。

2.3.2.1.6 将器件从 JTAG 控制中释放

参考函数：ReleaseDevice

在所需的 JTAG 通信完成后，CPU 从 JTAG 控制中释放出来。有两种方法来完成这一任务：

- 断开外部 JTAG 硬件并且执行一个真上电复位。然后 MSP430 开始执行开始地址为 0xFFFEh 的程序代码（复位矢量）。
- 将 MSP430 从 JTAG 控制中释放出来。通过使用 JTAG 控制信号寄存器执行一个复位来完成这一操作。然后，必须通过使用 IR_CNTRL_SIG_RELEASE 指令将 CPU 从 JTAG 控制中释放出来。然后目标 MSP430 开始执行存储在 0xFFFE 中的程序。

释放目标器件的流程：

IR_SHIFT("IR_CNTRL_SIG_16BIT")
DR_SHIFT16(0x2C01) : 应用复位
DR_SHIFT16(0x2401) : 删除复位
IR_SHIFT("IR_CNTRL_SIG_RELEASE")
目标 CPU 使用存储在位置 0x0FFFE（复位矢量）的地址开始执行程序。

NOTE

在擦除-编程-验证存储器访问周期期间，不建议将器件从 JTAG 控制中释放出来（或者执行一个加电循环）。 将器件从 JTAG 控制中释放出来将开始执行之前已编辑的用户代码，这有可能改变闪存内容。在这种情况下，用最初已编辑代码镜像对存储器内容的验证将返回失败结果。

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

2.3.2.2 5xx 和 6xx 系列的函数参考

2.3.2.2.1 将 CPU 置于 JTAG 控制之下

参考函数：GetDevice_430Xv2

对于 5xx 和 6xx 系列，将 JTAG 控制信号寄存器的位 10 (TCE1) 设置为 1，从而使 CPU 处于 JTAG 控制之下。虽然将目标器件置于 JTAG 控制之下与节 2.3.2.1.1 中描述的流程完全一样，必须采取额外的操作来完全控制目标 CPU；例如，不建议通过设置 JTAG 控制信号寄存器中的 POR 信号在不执行 CPU 复位的情况下获得对 CPU 的控制权。此外，必须注意，通过设置 CPUSUSP 信号并提供多个 TCLK 直到 CPU 预取管道被清除，CPU 会处于完全仿真状态（相当于 MSP430 和 MSP430X 架构的取指令状态）。图 2-15 显示了使 CPU 处于完全仿真状态所需的完整序列。

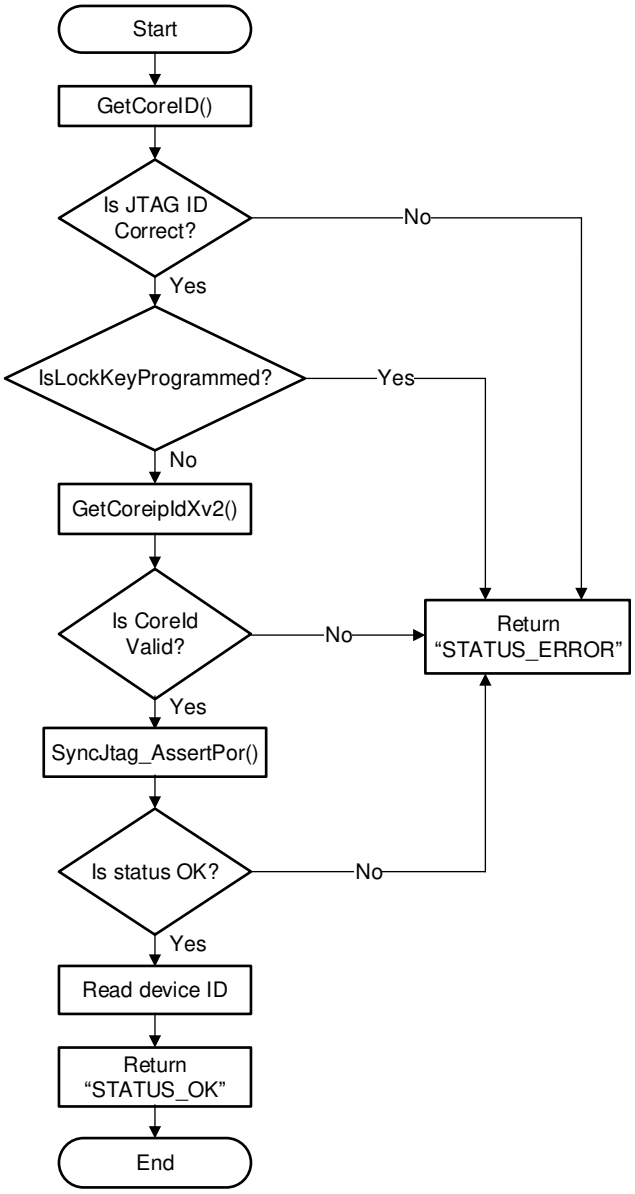

图 2-15. 将 CPU 置于 JTAG 控制之下

Copyright © 2021 Texas Instruments Incorporated

参考函数：GetCoreID

图 2-16 显示了针对 MSP430Xv2 器件的 JTAG 进入序列。如果第一个进入序列方法未返回有效的 JTAG-ID，将执行第二个使用“神奇模式”的序列方法。。这两个进入方法的区别为：第二个方法将器件保持在复位状态中并且通过使用 JTAG 邮箱来移入一个“神奇模式” (0xA55A)。由 BootCode 读取神奇模式，并且器件被传入 LPM4。如果器件处于 LPM4 中，无用户代码被执行。神奇模式强制器件复位。

在特殊情况下，器件处于 LPMx.5 中 (低功耗模式，在这个模式中，JTAG 未加电并且 JTAG 引脚被 ioLock 锁定) ，需要另外一个机制将器件置于 JTAG 控制之下。只有 TEST 和 REST 引脚不被 ioLock 拉为低电平。这意味着这些引脚必须被用于获得对器件的控制权。与在 Replicator 实现方案中使用 TDI/TDO 和 TMS 引脚的正常 SBW 通信相比，TEST 和 RST 引脚用于 SBW 通信以禁用 ioLock。

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

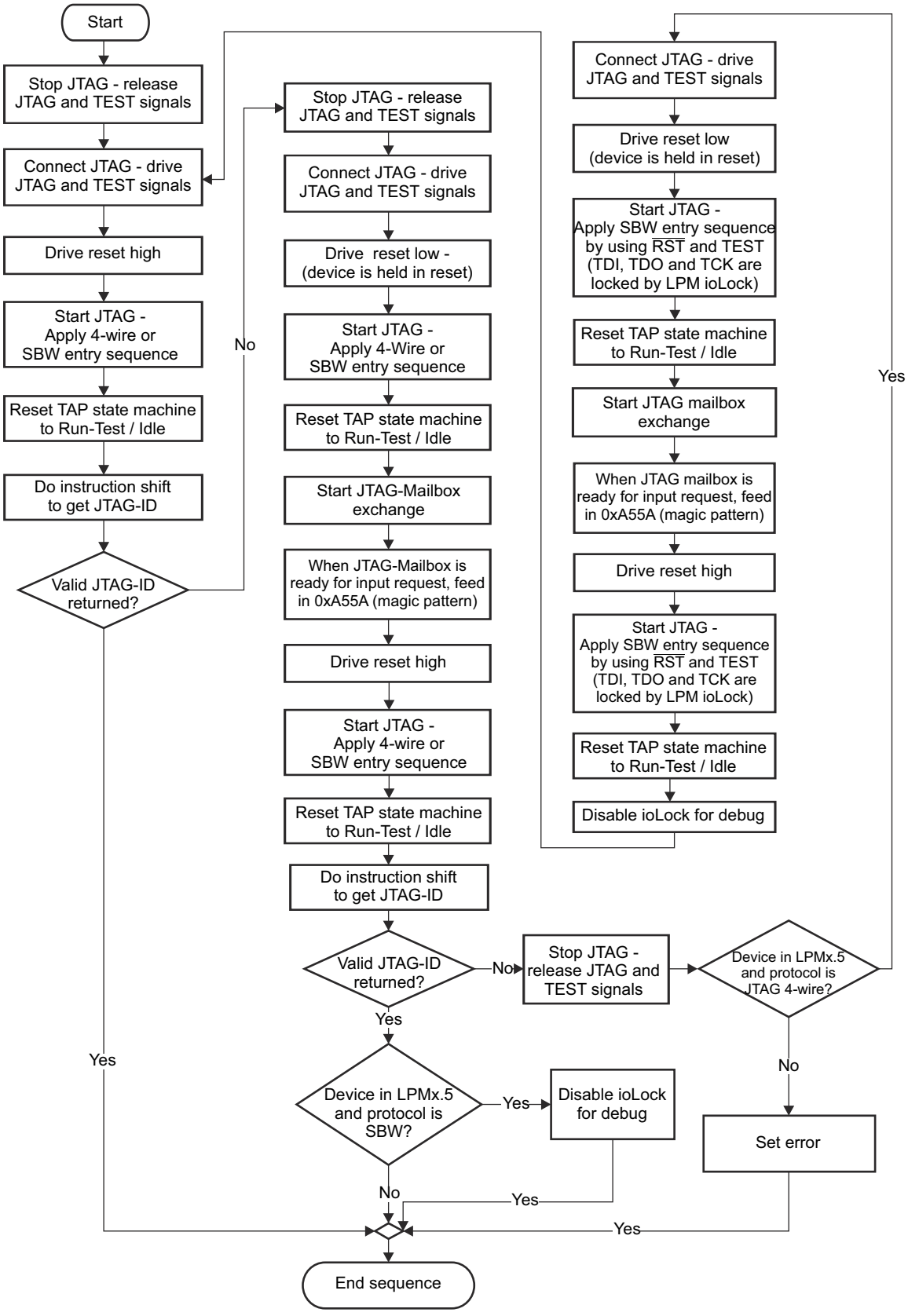

图 2-16. 针对 430Xv2 器件的 JTAG 进入序列

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

- 参考函数：GetCoreipIdXv2()

IR_Shift(IR_COREIP_ID)	
CoreId = DR_Shift16(0)	
CoreId = 0 ?	**有**
否	
IR_Shift(IR_DEVICE_ID)	
DeviceIdPointer = DR_Shift20(0)	

- 参考函数：SyncJtag_AssertPor()

IR_SHIFT("IR_CNTRL_SIG_16BIT")	
DR_SHIFT16(0x1501)	
IR_SHIFT("IR_CNTRL_SIG_CAPTURE")	
DR_SHIFT16(0x0000)	**否**
TDOword 的 位 9 = 1 ?	
支持	
CPU 受 JTAG 控制 - 之后始终应用上电复位 (POR)。	

2.3.2.2.2 设置目标 CPU 程序计数器 (PC)

为了使用 MSP430Xv2 架构提供的 JTAG 接口的一些特性，需要设置目标器件的 CPU PC。下面的流程用于完成这一操作。借助于 MSP430Xv2 架构，强烈建议在设置 PC 后，除了节 2.3.3.3 和节 2.3.7 下描述的快速访问方法外，不执行任何额外的内存访问。在设置 PC 后，目标器件可从 JTAG 控制中释放或者继续通过提供 TCLK 进行计时来执行用户程序代码，此程序代码之前被存储在 PC 现在指向的内存位置。在任何一种情况下，在内存可被再次访问之前，必须将 CPU 再次置于节 2.3.2.2.1 中描述的仿真状态中。

- MSP430Xv2 架构：参考函数：SetPC_430Xv2

在执行以下序列之前，CPU 必须处于完全仿真状态。
ClrTCLK
IR_SHIFT("IR_DATA_16BIT")
SetTCLK
DR_SHIFT16("MOVA opcode incl. upper nibble of 20 bit PC value")
clrTCLK
IR_SHIFT("IR_CNTRL_SIG_16BIT")
DR_SHIFT16(0x1400)：释放低字节
IR_SHIFT("IR_DATA_16BIT")
ClrTCLK
SetTCLK
DR_SHIFT16("PC_Value")：为 PC 插入低 16 位值
ClrTCLK
SetTCLK
DR_SHIFT16(0x4303)：插入要由 CPU 预取的 NOP 指令
ClrTCLK：现在已设置 PC
IR_SHIFT("IR_ADDR_CAPTURE")：禁用 IR_DATA_16BIT
DR_SHIFT20(0x00000)
加载 PC 完成

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

2.3.2.2.3 在受 JTAG 控制时复位 CPU

参考函数：ExecutePOR_430Xv2

NOTE

在具有低功耗加速器 (LEA) 的器件上，使用所述函数执行 POR 会使 LEA 模块复位。

- 具有闪存的 MSP430Xv2 架构

IR_SHIFT("IR_CNTRL_SIG_CAPTURE")：返回 JTAG ID
ClrTCLK：提供一个时钟周期来清空管道
SetTCLK
IR_SHIFT("IR_CNTRL_SIG_16BIT")：准备访问 JTAG CNTRL SIG 寄存器
DR_Shift16(0x0C01)：释放 CPUSUSP 信号并应用 POR 信号
DR_Shift16(0x0401)：再次释放 POR 信号
ClrTCLK
SetTCLK
ClrTCLK
SetTCLK
ClrTCLK
SetTCLK
ClrTCLK：另外两个时钟周期释放 CPU 内部 POR 延迟信号
SetTCLK
ClrTCLK
SetTCLK
IR_SHIFT("IR_CNTRL_SIG_16BIT")：再次设置 CPUSUSP 信号
DR_Shift16(0x0501)
ClrTCLK：...并多提供一个时钟周期
SetTCLK
CPU 现在处于完全仿真状态
现在通过设置 WDT_CNTRL 寄存器中的 HOLD 信号（即，使用 WriteMem_430Xv2）来禁用目标器件上的看门狗计时器
可以通过运行以下命令来检查“完全仿真状态”
IR_Shift("IR_CNTRL_SIG_CAPTURE")
DR_Shift16(0)：返回值和 0x0301 应为 true

- 具有 FRAM 存储器的 MSP430Xv2 架构

ClrTCLK：提供一个时钟周期来清空管道
SetTCLK
IR_SHIFT("IR_CNTRL_SIG_16BIT")：准备访问 JTAG CNTRL SIG 寄存器
DR_Shift16(0x0C01)：释放 CPUSUSP 信号并应用 POR 信号
DR_Shift16(0x0401)：再次释放 POR 信号
IR_Shift(IR_DATA_16BIT)：将 PC 设置为安全存储器位置
ClrTCLK
SetTCLK
ClrTCLK

Copyright © 2021 Texas Instruments Incorporated

SetTCLK
DR_Shift16(SAFE_FRAM_PC)：PC 设置为 0x4 - MAB 值可以是 0x6 或 0x8
ClrTCLK：将安全地址驱动到 PC 中
SetTCLK
IR_SHIFT("IR_DATA_CAPTURE")
ClrTCLK：另外两个时钟周期释放 CPU 内部 POR 延迟信号
SetTCLK
ClrTCLK
SetTCLK
IR_SHIFT("IR_CNTRL_SIG_16BIT")：再次设置 CPUSUSP 信号
DR_Shift16(0x0501)
ClrTCLK：...并多提供一个时钟周期
SetTCLK
CPU 现在处于完全仿真状态
现在通过设置 WDT_CNTRL 寄存器中的 HOLD 信号（即，使用 WriteMem_430Xv2，注意各个 FRAM 器件组的不同 WDT 地址）禁用目标器件上的看门狗计时器
使用默认值初始化测试存储器以确保 PC 值和 MAB 之间的一致性（同步后 MAB 会 +2）- 使用 WriteMem_430Xv2 将 0x3FFF 写入地址 0x06 和 0x08（仅适用于具有 JTAG ID 0x91 或 0x99 的器件）
可以通过运行以下命令来检查“完全仿真状态”
IR_Shift("IR_CNTRL_SIG_CAPTURE")
DR_Shift16(0)：返回值和 0x0301 应为 true

2.3.2.2.4 将器件从 JTAG 控制中释放

同样对于 5xx 和 6xx 系列，节 2.3.2.1.6 中描述的两种方法均可采用。一个使用 JTAG 控制信号寄存器的 POR（上电复位）不等同于真正的上电复位，真正的上电复位会在 POR 之前额外发出一个 BOR（掉电复位）。只有 BOR 导致目标器件的引导代码执行，此代码执行不同的校准和配置任务。因此，5xx 和 6xx JTAG 接口通过访问一个专用的 JTAG 数据寄存器来获得通过 JTAG 接口生成一个 BOR 的增强功能。只要 JTAG 数据寄存器中适当的 BOR 位被设定，器件从 JTAG 中释放出来并且执行一个完整的欠压复位启动序列。执行细节请参阅 ReleaseDevice_Xv2 参考函数。更多有关不同复位源和器件启动状态的信息，另请参阅《MSP430F5xx 和 MSP430F6xx 系列用户指南》中的*系统复位、中断和运行模式，系统控制模块 (SYS)* 一章。

参考函数：ReleaseDevice_430Xv2

2.3.2.2.5

NOTE

在擦除-编程-验证存储器访问周期期间，不建议将器件从 JTAG 控制中释放出来（或者执行一个加电循环）。将器件从 JTAG 控制中释放出来将开始执行之前已编辑的用户代码，这有可能改变闪存内容。在这种情况下，用最初已编辑代码镜像对存储器内容的验证将返回失败结果。

2.3.3 用 JTAG 访问非闪存存储器位置

2.3.3.1 读取访问

为了从任一存储器地址位置（外设，RAM，或者闪存 / FRAM）中读取数据，必须使用 JTAG 控制信号寄存器将 R/W 信号设定为 READ（位 0 设定为 1）。当 TCLK 为 0 时，必须使用 IR_ADDR_16BIT 指令将 MSP430 MAB 设置为要读取的特定地址。为了捕捉 MSP430 MDB 相应的值，IR_DATA_TO_ADDR 指令必须被执行。在 TCLK 的下一个上升沿之后，这个地址内的数据出现在 MDB 上。现在可以使用 16 位 JTAG 数据访问方式从 TDO 引脚捕获并读出 MDB。当 TCLK 再次被设定为低电平之后，从下一个存储器位置读取的地址可被应用于目标 MAB。

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

下面列出的是从目标器件的任一存储器地址读取数据所需的流程。针对 MSP430 和 MSP430X 架构的执行方式显示如下。

- MSP430 架构，参考函数：ReadMem

将 CPU 设置为已停止状态 (HaltCPU)		
ClrTCLK		
IR_SHIFT("IR_CNTRL_SIG_16BIT")		
DR_SHIFT16(0x2409)	：从存储器中读取一个字。若要读取一个字节，需要移位的值为 0x2419。	
IR_SHIFT("IR_ADDR_16BIT")		**是**
DR_SHIFT16("Address")	：设置所需地址	
IR_SHIFT("IR_DATA_TO_ADDR")		
SetTCLK		
ClrTCLK		
DR_SHIFT16(0x0000)	：在 TDO 上移出存储器值	
再次读取？		
否		
现在应该执行 ReleaseCPU，使 CPU 恢复正常运行。		

- MSP430X 架构，参考函数：ReadMem_430X

将 CPU 设置为已停止状态 (HaltCPU)		
ClrTCLK		
IR_SHIFT("IR_CNTRL_SIG_16BIT")		
DR_SHIFT16(0x2409)	：从存储器中读取一个字。若要读取一个字节，需要移位的值为 0x2419。	
IR_SHIFT("IR_ADDR_16BIT")		**是**
DR_SHIFT20("Address")	：设置所需地址	
IR_SHIFT("IR_DATA_TO_ADDR")		
SetTCLK		
ClrTCLK		
DR_SHIFT16(0x0000)	：在 TDO 上移出存储器值	
再次读取？		
否		
现在应该执行 ReleaseCPU，使 CPU 恢复正常运行。		

Copyright © 2021 Texas Instruments Incorporated

- MSP430Xv2 架构，参考函数：ReadMem_430Xv2

在执行以下序列之前，CPU 必须处于完全仿真状态。	
ClrTCLK	**是**
IR_SHIFT("IR_CNTRL_SIG_16BIT")	
DR_SHIFT16(0x0501) ：从存储器中读取一个字。若要读取一个字节，需要移位的值为 0x0511。	
IR_SHIFT("IR_ADDR_16BIT")	
DR_SHIFT20("Address") ：设置所需地址	
IR_SHIFT("IR_DATA_TO_ADDR")	
SetTCLK	
ClrTCLK	
DR_SHIFT16(0x0000) ：在 TDO 上移出存储器值	
SetTCLK	
ClrTCLK	
SetTCLK	
再次读取？	
否	
CPU 现在再次处于完全仿真状态。	

2.3.3.2 写入访问

为了向一个外设的内存位置或者 RAM 中写入数据（但不是向闪存或者 FRAM 中写入），必须使用 JTAG 控制信号寄存器将 R/W 信号设定为 WRITE（位 0 设定为 0）。TCLK 为低电平时，必须使用 IR_ADDR_16BIT 指令将 MAB 设定为特定的地址。必须使用 IR_DATA_TO_ADDR 指令和一个 16 位 JTAG 数据输入移位来将 MDB 设定为将被写入的数据值。在 TCLK 的下一个上升沿上，这个数据被写入到 MAB 上的值所设定的已选地址内。当 TCLK 被置为低电平时，下一个将被写入的地址和数据可被应用于 MAB 和 MDB。写入操作完成后，建议将 R/W 信号设定回 READ。下面是针对一个外设或者 RAM 内存地址写入的流程。针对 MSP430 和 MSP430X 架构的执行显示如下。

- MSP430 架构，参考函数：WriteMem

将 CPU 设置为已停止状态 (HaltCPU)	
ClrTCLK	
IR_SHIFT("IR_CNTRL_SIG_16BIT")	
DR_SHIFT16(0x2408) ：向存储器写入一个字。要写入一个字节，需要移位的值为 0x2418。	
IR_SHIFT("IR_ADDR_16BIT")	**是**
DR_SHIFT16("Address") ：设置所需地址	
IR_SHIFT("IR_DATA_TO_ADDR")	
DR_SHIFT16("Data") ：发送 16 位数据	
SetTCLK	
再次写入？	
否	
现在应该执行 ReleaseCPU，使 CPU 恢复正常运行。	

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

- MSP430X 架构，参考函数：WriteMem_430X

将 CPU 设置为已停止状态 (HaltCPU)		
ClrTCLK		
IR_SHIFT("IR_CNTRL_SIG_16BIT")		
DR_SHIFT16(0x2408)	：向存储器写入一个字。若要写入一个字节，需要移位的值为 0x2418。	
IR_SHIFT("IR_ADDR_16BIT")		是
DR_SHIFT20("Address")	：设置所需地址	
IR_SHIFT("IR_DATA_TO_ADDR")		
DR_SHIFT16("Data")	：发送 16 位数据	
SetTCLK		
再次写入？		
否		
现在应该执行 ReleaseCPU，使 CPU 恢复正常运行。		

- MSP430Xv2 架构，参考函数：WriteMem_430Xv2

在执行以下序列之前，CPU 必须处于完全仿真状态。		
ClrTCLK		是
IR_SHIFT("IR_CNTRL_SIG_16BIT")		
DR_SHIFT16(0x0500)	：向存储器写入一个字。若要写入一个字节，需要移位的值为 0x0510。	
IR_SHIFT("IR_ADDR_16BIT")		
DR_SHIFT20("Address")	：设置所需地址	
SetTCLK		
IR_SHIFT("IR_DATA_TO_ADDR")		
DR_SHIFT16("Data")	：发送 16 位数据	
ClrTCLK		
IR_SHIFT("IR_CNTRL_SIG_16BIT")		
DR_SHIFT16(0x0501)		
SetTCLK		
ClrTCLK		
SetTCLK		
再次写入？		
否		
CPU 现在再次处于完全仿真状态。		

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

2.3.3.3 存储器阵列的快速访问

MSP430 上执行的 JTAG 通信也支持用更有效的方式访问一个存储器阵列。指令 IR_DATA_QUICK 被用于完成这个操作。R/W 信号选择执行一个读取访问还是一个写入访问。在这个指令被载入到 JTAG IR 寄存器之前，MSP430CPU 的程序计数器 (PC) 必须被设定为所需的存储器起始地址。在 IR_DATA_QUICK 指令被移入 IR 寄存器后，PC 在 TCLK 的每个下降边沿上加 2，从而自动将 PC 指向下一个存储器位置。通过使用一个 DR_SHIFT16 操作，IR_DATA_QUICK 指令可将相应的 MDB 设定为一个所需的值（写入），或者捕捉（读取）MDB。当 TCLK 为低电平时，MDB 应该被设定。在下一个上升 TCLK 边沿上，MDB 上的值被写入 PC 寻址的位置内。为了读取一个内存位置，在 DR_SHIFT16 操作被执行前，TCLK 必须为高电平。

2.3.3.3.1 快速读取流程（所有存储器位置）

- MSP430 和 MSP430X 架构，参考函数：ReadMemQuick

将 PC 设置为起始地址 – 4（SetPC 或 SetPC_430X，视架构而定）	
将 CPU 切换为已停止状态 (HaltCPU)	
ClrTCLK	
IR_SHIFT("IR_CNTRL_SIG_16BIT")	
DR_SHIFT16(0x2409)　　：将 RW 设置为读取	
IR_SHIFT("IR_DATA_QUICK")	
SetTCLK	是
DR_SHIFT16(0x0000)　　：在 TDO 上移出存储器值	
ClrTCLK　　：自动递增 PC	
从下一个地址读取？	
否	
现在应该执行 ReleaseCPU，使 CPU 恢复正常运行。如果需要，使目标 CPU 的 PC 复位 (SetPC)。	

- MSP430Xv2 架构，参考函数：ReadMem_430Xv2

在执行以下序列之前，CPU 必须处于完全仿真状态。	
将 PC 设置为起始地址 (SetPC_430Xv2)	
SetTCLK	
IR_SHIFT("IR_CNTRL_SIG_16BIT")	
DR_SHIFT16(0x0501)　　：将 RW 设置为读取	
IR_SHIFT("IR_ADDR_CAPTURE")	
IR_SHIFT("IR_DATA_QUICK")	
SetTCLK	是
ClrTCLK　　：自动递增 PC	
DR_SHIFT16(0x0000)　　：在 TDO 上移出存储器值	
从下一个地址读取？	
否	
使 CPU 处于完全仿真状态。	

NOTE

对于 MSP430F5xx 和 MSP430F6xx 系列，在进行快速存储器访问时必须小心，这是因为 PC 已经指向要读取的实际地址之前的一个地址。这会导致安全访问违规，特别是在物理存储器块的末尾。

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

2.3.3.3.2 快速写入流程

- MSP430 和 MSP430X 架构，参考函数：WriteMemQuick

将 PC 设置为起始地址 – 4 (SetPC)	
将 CPU 切换为已停止状态 (HaltCPU)	
ClrTCLK	
IR_SHIFT("IR_CNTRL_SIG_16BIT")	
DR_SHIFT16(0x2408) ：将 RW 设置为写入	是
IR_SHIFT("IR_DATA_QUICK")	
DR_SHIFT16("Data") ：设置数据	
SetTCLK	
ClrTCLK ：自动递增 PC	
写入下一个地址？	
否	
现在应该执行 ReleaseCPU，使 CPU 恢复正常运行。如果需要，使目标 CPU 的 PC 复位 (SetPC)。	

- MSP430Xv2 架构

  对于 MSP430Xv2 架构，在闪存和 FRAM 器件中没有快速写入操作的具体实现方式。REP430 固件中的可用实现方式是调用 WriteMem_430Xv2() 函数以向指定存储器进行逐字写入。

2.3.4 编辑闪存存储器（使用板载闪存控制器）

2.3.4.1 针对 1xx，2xx，4xx 系列的函数参考

参考函数：WriteFLASH

这个部分描述了一个所提供的对 MSP430 器件中的闪存模块进行编程的方法。它采用的步骤与用户定义的应用软件将用的步骤相同，此步骤被设定在一个生产设备 MSP430 器件中。支持非连续闪存寻址。

此编程方法要求用户在擦除或编程周期执行期间提供一个 350kHz ±100kHz 的 TCLK 频率。在 Spy-Bi-Wire 模式下必须应用于 SBWTCK 的频率与在 4 线制模式下应用于 TCK 的频率相同。

更多有关闪存控制器时序的信息，请参阅相应的 MSP430 用户指南和特定于器件的数据表。表 2-9 显示了所需的 TCLK 周期的最小数量，具体取决于在闪存上执行的操作（如 MSP430 用户指南中定义的那样，FCTL2 寄存器位 0 至位 7 = 0x40）。

表 2-9. 擦除和编程最小 TCLK 时钟周期

闪存操作	最小 TCLK 计数
段擦除	4820（默认）或 9628（MSP430ixx 系列）
批量擦除	5300 至 10600(1)
程序字	35

(1) 取决于 MSP430 器件，请参阅器件专用数据表。更多详细信息，请参阅节 2.3.5。

下面的 JTAG 通信流程显示了使用板载闪存控制器进行 MSP430 闪存编程的操作流程。在这个执行中，16 位字被设定到主闪存区域中。为了对字节进行编程，在编程模式中时，JTAG CNTRL_SIG 寄存器中的 BYTE（字节）位必须被设定为高电平。StartAddr 是将被编程的闪存阵列的起始地址。

将 CPU 切换为已停止状态 (HaltCPU)
ClrTCLK
IR_SHIFT("IR_CNTRL_SIG_16BIT")
DR_SHIFT16(0x2408) ：将 RW 设置为写入
IR_SHIFT("IR_ADDR_16BIT")

Copyright © 2021 Texas Instruments Incorporated

DR_SHIFT16(0x0128) (1)	：指向 FCTL1 地址	
IR_SHIFT("IR_DATA_TO_ADDR")		
DR_SHIFT16(0xA540)	：启用闪存写入访问	
SetTCLK		
ClrTCLK		
IR_SHIFT("IR_ADDR_16BIT")		
DR_SHIFT16(0x012A) (1)	：指向 FCTL2 地址	
IR_SHIFT("IR_DATA_TO_ADDR")		
DR_SHIFT16(0xA540)	：源为 MCLK，除以 1	
SetTCLK		
ClrTCLK		
IR_SHIFT("IR_ADDR_16BIT")		
DR_SHIFT16(0x012C) (1)	：指向 FCTL3 地址	
IR_SHIFT("IR_DATA_TO_ADDR")		
DR_SHIFT16(0xA500) (2)	：清除 FCTL3 寄存器	
SetTCLK		
ClrTCLK		
IR_SHIFT("IR_CNTRL_SIG_16BIT")		
DR_SHIFT16(0x2408)	：将 RW 设置为写入	是
IR_SHIFT("IR_ADDR_16BIT")		
DR_SHIFT16("Address") (1)	：设置写入地址	
IR_SHIFT("IR_DATA_TO_ADDR")		
DR_SHIFT16("Data")	：设置写入数据	
SetTCLK		
ClrTCLK		
IR_SHIFT("IR_CNTRL_SIG_16BIT")		
DR_SHIFT16(0x2409)	：将 RW 设置为读取	
SetTCLK	*重复 35 次* (3)	
ClrTCLK		
写入另外一个闪存地址？		
否		
IR_SHIFT("IR_CNTRL_SIG_16BIT")		
DR_SHIFT16(0x2408)	：将 RW 设置为写入	
IR_SHIFT("IR_ADDR_16BIT")		
DR_SHIFT16(0x0128) (1)	：指向 FCTL1 地址	
IR_SHIFT("IR_DATA_TO_ADDR")		
DR_SHIFT16(0xA500)	：禁用闪存写入访问	
SetTCLK		
ClrTCLK		
IR_SHIFT("IR_ADDR_16BIT")		
DR_SHIFT16(0x012C) (1)	：指向 FCTL3 地址	
IR_SHIFT("IR_DATA_TO_ADDR")		
DR_SHIFT16(0xA510) (4)	：设置 FCTL3 中的 LOCK 位	
SetTCLK		
现在应该执行 ReleaseCPU，使 CPU 恢复正常运行。		

(1) 当对一个 MSP430X 架构器件进行编程时，用 DR_SHIFT20（“地址”）替代。
(2) 对于 2xx 器件的 Info 段 A 编程，用 0xA540 取代。
(3) 需要正确时序。必须满足 350kHz ±100kHz 的最小和最大 TCLK 频率要求。
(4) 对于 2xx 器件的 Info 段 A 编程，用 0xA550 取代。

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

2.3.4.2 5xx 和 6xx 系列的函数参考

由于 5xx 和 6xx 器件有一个专用的片上时序发生器，因此与其他 MSP430 系列相比，闪存访问要容易得多。用户无需确保 TCLK 信号具有特定的擦除或编程频率。存储器擦除和写访问所需的所有时序都是自动生成的。

以下说明的基本前提是可从 RAM 内部开始闪存访问操作，相关的《MSP430F5xx 和 MSP430F6xx 系列用户指南》章节对此进行了说明。此文档介绍了如何在目标器件 RAM 中加载相应的代码以及如何使用 JTAG 接口控制代码的正确执行。可以通过将器件从 JTAG 控制中释放来控制目标代码的执行。将器件从 JTAG 控制中释放可以使 CPU 在自由运行模式下执行程序代码。在所需的操作完成后，器件必须再次由 JTAG 控制。

这种方法有优点也有缺点。虽然自由运行的器件能够将闪存编程速度增加到上限，但需要一个借助 JTAG 的轮询机制来检索当前的目标器件状态。另一方面，这样一个轮询机制并不适合于那些并行访问多个目标器件的系统。所有目标不会以完全相同的频率运行，因此建议并行访问系统的方法是将目标置于 JTAG 控制之下。在 REP430 固件中没有实现此方法。

在目标器件的 CPU 和 JTAG 之间交换信息（例如，为了轮询器件状态）会使用 5xx JTAG 实现方案的一个新功能：JTAG 邮箱系统。JTAG 邮箱系统背后的理念是在调试、编程和测试期间拥有一个与 CPU 的直接接口，该接口对于该系列中的所有器件均相同，并使用很少或不使用用户应用程序资源（请参阅《MSP430F5xx 和 MSP430F6xx 系列用户指南》中的*系统复位、中断和运行模式，系统控制模块 (SYS)* 一章）。

图 2-17 显示了通过 JTAG 接口在 5xx 和 6xx 器件上执行闪存操作所需的一般流程。术语“闪存访问代码”表示可用于执行闪存访问操作的相应 MSP430 可执行代码。以下几个部分使用术语“闪存写入代码”表示用于对闪存进行编程的代码，而使用“闪存擦除代码”表示用于擦除闪存的代码。

Copyright © 2021 Texas Instruments Incorporated

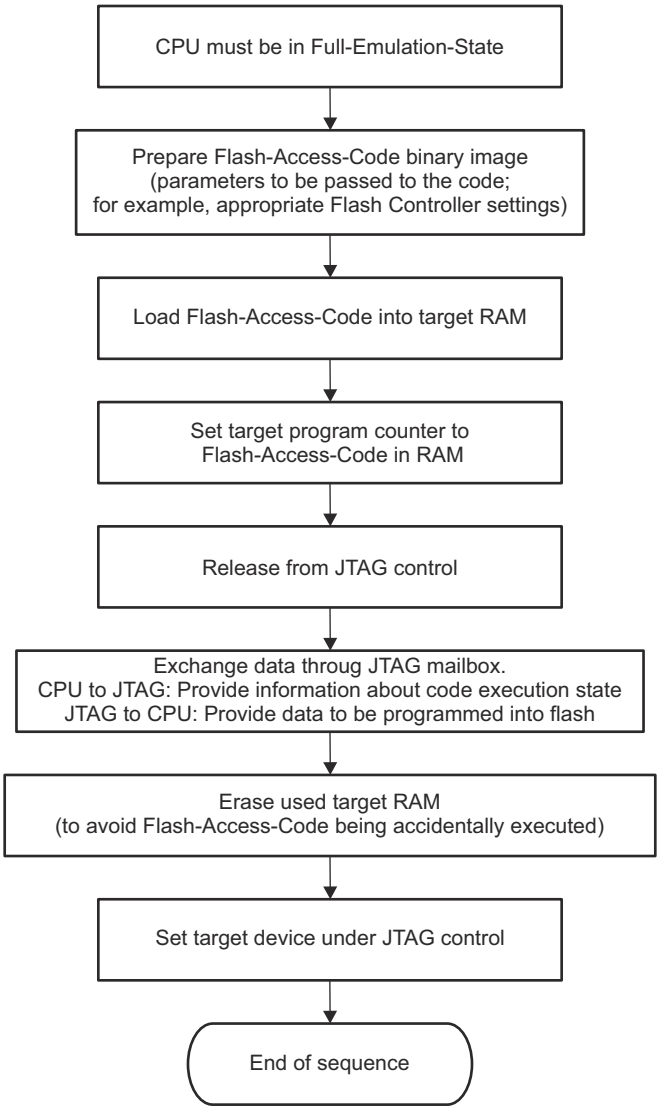

图 2-17. 访问闪存

参考函数：WriteFLASH_430Xv2

本部分介绍了一种随后使用 16 位字数据对闪存进行编程的方法：在 RAM 中执行相应“闪存写入代码”并通过 JTAG 邮箱系统将数据提供给 CPU。提供的源代码示例包括一个可表示为二进制参数的闪存写入代码示例。图 2-18 显示了随本文档提供的闪存访问代码的二进制映像的通用映射。

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

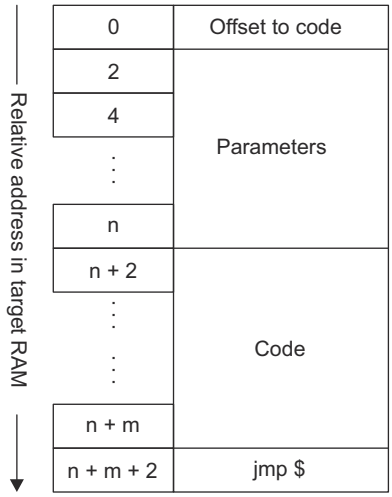

图 2-18. 闪存访问代码二进制镜像映射

- 此代码与位置无关。
- 第一个地址保存一个相对于实际程序代码起始地址的偏移值。当前的地址加上偏移值得到了在代码开始执行前必须分配给程序计数器的值。
- 针对代码专用参数的空间。
- 实际程序代码。
- 末端的无穷循环。

闪存-写入-代码特别采用以下参数：

- *StartAddr*：目标存储器中要写入的第一个地址
- *Length*：要写入的 16 位字的数量
- *FCTL3*：写入到闪存控制器模块 FCTL3 的值（主要确定是否应该设置 LOCKA）

在执行闪存写入代码时，要编程到目标闪存中的数据必须通过 JTAG 邮箱系统提供。以下序列说明了如何实现这一点。

- 从 JTAG 控制中释放

从 JTAG 控制中释放目标器件（自由运行）		
`IR_SHIFT("IR_JB_EXCHANGE")`		是
`DR_SHIFT16(0x0000)`	否	
TDOword 的位 0 = 1？		
是		
`DR_SHIFT16(0x0001)` ：向 JTAG 邮箱发送输入请求		
`DR_SHIFT16("Data")` ：将 16 位字移入 JTAG 邮箱		
写入另外一个闪存地址？		
否		
使目标器件处于完全仿真状态		

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

2.3.5 擦除闪存（使用板载闪存控制器）

2.3.5.1 针对 1xx，2xx，4xx 系列的函数参考

参考函数：EraseFLASH

本部分介绍了如何擦除闪存的一个段 (ERASE_SGMT)，如何擦除器件主存储器 (ERASE_MAIN)，以及如何擦除包括主闪存段和信息闪存段的完整闪存地址范围 (ERASE_MASS)。此方法要求用户在擦除周期执行期间以及闪存编程期间提供一个频率为 350kHz ±100kHz 的 TCLK 信号。下面的表分别显示了段擦除和批量擦除流程，以及闪存控制器执行每个操作（FCTL2 寄存器位 0 至 7 = 0x40）时所需的最少 TCLK 周期数量。

2.3.5.1.1 擦除一个闪存段的流程

将 CPU 切换为已停止状态 (HaltCPU)	
ClrTCLK	
IR_SHIFT("IR_CNTRL_SIG_16BIT")	
DR_SHIFT16(0x2408)	：将 RW 设置为写入
IR_SHIFT("IR_ADDR_16BIT")	
DR_SHIFT16(0x0128) (1)	：指向 FCTL1 地址
IR_SHIFT("IR_DATA_TO_ADDR")	
DR_SHIFT16(0xA502)	：启用闪存段擦除
SetTCLK	
ClrTCLK	
IR_SHIFT("IR_ADDR_16BIT")	
DR_SHIFT16(0x012A) (1)	：指向 FCTL2 地址
IR_SHIFT("IR_DATA_TO_ADDR")	
DR_SHIFT16(0xA540)	：源为 MCLK，除以 1
SetTCLK	
ClrTCLK	
IR_SHIFT("IR_ADDR_16BIT")	
DR_SHIFT16(0x012C) (1)	：指向 FCTL3 地址
IR_SHIFT("IR_DATA_TO_ADDR")	
DR_SHIFT16(0xA500) (4)	：清除 FCTL3 寄存器
SetTCLK	
ClrTCLK	
IR_SHIFT("IR_ADDR_16BIT")	
DR_SHIFT16("EraseAddr") (1)	：设置擦除地址 (2)
IR_SHIFT("IR_DATA_TO_ADDR")	
DR_SHIFT16(0x55AA)	：写入虚拟数据以开始擦除
SetTCLK	
ClrTCLK	
IR_SHIFT("IR_CNTRL_SIG_16BIT")	
DR_SHIFT16(0x2409)	：将 RW 设置为读取
SetTCLK	*重复 4819 次* (3)
ClrTCLK	
IR_SHIFT("IR_CNTRL_SIG_16BIT")	
DR_SHIFT16(0x2408)	：将 RW 设置为写入
IR_SHIFT("IR_ADDR_16BIT")	
DR_SHIFT16(0x0128) (1)	：指向 FCTL1 地址
IR_SHIFT("IR_DATA_TO_ADDR")	
DR_SHIFT16(0xA500)	：禁用闪存擦除
SetTCLK	

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

ClrTCLK	
IR_SHIFT("IR_ADDR_16BIT")	
DR_SHIFT16(0x012C) (1)	：指向 FCTL3 地址
IR_SHIFT("IR_DATA_TO_ADDR")	
DR_SHIFT16(0xA510) (4)	：设置 FCTL3 中的 LOCK 位
SetTCLK	
现在应该执行 ReleaseCPU，使 CPU 恢复正常运行。	

(1) 当编辑一个 MSP430X 架构器件时，用 DR_SHIFT20（“地址”）替代。
(2) EraseAddr 参数是指向将被擦除的闪存段的地址。
(3) 需要正确时序。必须满足 350kHz ±100kHz 的最小和最大 TCLK 频率要求。
(4) 对于 2xx 器件的 Info 段 A 编程，用 0xA540 取代。

2.3.5.1.2 擦除整个闪存地址空间的流程（批量擦除）

当使用所描述的方法执行批量或主存储器擦除时，除了频率为 350kHz ±100kHz 的 TCLK 信号（用于闪存时序发生器，请参阅数据表参数 f_{FTG}），必须考虑另外两个数据表参数。第一个是 $t_{CMErase}$（累积批量擦除时间），而第二个是 $t_{批量擦除}$（批量擦除时间）。当前已在专用的 MSP430 器件中实现这些参数的两个不同规格组合。表 2-10 显示了这些参数的概述（假定一个 450KHz 的最大 TCLK 频率）。

表 2-10. 闪存参数(f_{FTG}=540kHz)

执行	$t_{CMErase}$	$t_{批量擦除}$	闪存时序生成器产生的批量擦除持续时间
1	200ms	5300×t_{FTG}	11.1ms
2	20ms	10600×t_{FTG}	20ms

对于执行 1，为了确保在建议的 200ms 擦除时间内可以安全地擦除闪存空间，5300 个 TCLK 周期被传送至目标 MSP430 器件并且重复 19 次。在使用执行 2 时，以下序列只需执行一次。

NOTE

MSP430F2xx 器件有四个信息存储器段，每个存储器段的大小为 64 字节。段 INFOA（更多信息请参阅《MSP430F2xx 系列用户指南》）是一个可锁定的闪存信息段并且包含针对 MSP430F2xx 时钟系统 (DCO) 的重要校准数据，此数据对于生产测试时被编程的指定器件唯一。剩余的三个信息存储器段（INFOB，INFOC 和 INFOD）在 INFOA 被锁定时不能被批量擦除操作擦除。INFOB、INFOC 和 INFOD 可被逐段擦除，此擦除操作与 INFOA 的锁定设置无关。将 INFOA 解锁可实现批量擦除操作。

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

将 CPU 切换为已停止状态 (HaltCPU)		执行一次或者重复 19 次 (3)
ClrTCLK		
IR_SHIFT("IR_CNTRL_SIG_16BIT")		
DR_SHIFT16(0x2408)	: 将 RW 设置为写入	
IR_SHIFT("IR_ADDR_16BIT")		
DR_SHIFT16(0x0128) (1)	: FCTL1 地址	
IR_SHIFT("IR_DATA_TO_ADDR")		
DR_SHIFT16(0xA506)	: 启用闪存批量擦除	
SetTCLK		
ClrTCLK		
IR_SHIFT("IR_ADDR_16BIT")		
DR_SHIFT16(0x012A) (1)	: FCTL2 地址	
IR_SHIFT("IR_DATA_TO_ADDR")		
DR_SHIFT16(0xA540)	: 源为 MCLK，除数为 0	
SetTCLK		
ClrTCLK		
IR_SHIFT("IR_ADDR_16BIT")		
DR_SHIFT16(0x012C) (1)	: FCTL3 地址	
IR_SHIFT("IR_DATA_TO_ADDR")		
DR_SHIFT16(0xA500) (4)	: 清除 FCTL3 寄存器	执行一次或者重复 19 次 (3)
SetTCLK		
ClrTCLK		
IR_SHIFT("IR_ADDR_16BIT")		
DR_SHIFT16("EraseAddr") (1)	: 设置擦除地址 (2)	
IR_SHIFT("IR_DATA_TO_ADDR")		
DR_SHIFT16(0x55AA)	: 写入虚拟数据以开始擦除	
SetTCLK		
ClrTCLK		
IR_SHIFT("IR_CNTRL_SIG_16BIT")		
DR_SHIFT16(0x2409)	: 将 RW 设置为读取	
SetTCLK	执行 10600 或者 5300 次(3)	
ClrTCLK		
IR_SHIFT("IR_CNTRL_SIG_16BIT")		
DR_SHIFT16(0x2408)	: 将 RW 设置为写入	
IR_SHIFT("IR_ADDR_16BIT")		
DR_SHIFT16(0x0128) (1)	: FCTL1 地址	
IR_SHIFT("IR_DATA_TO_ADDR")		
DR_SHIFT16(0xA500)	: 禁用闪存擦除	
SetTCLK		
ClrTCLK		
IR_SHIFT("IR_ADDR_16BIT")		
DR_SHIFT16(0x012C) (1)	: 指向 FCTL3 地址	
IR_SHIFT("IR_DATA_TO_ADDR")		
DR_SHIFT16(0xA510) (4)	: 设置 FCTL3 中的 LOCK 位	
SetTCLK		
现在应该执行 ReleaseCPU，使 CPU 恢复正常运行。		

(1) 当对一个 MSP430X 架构器件进行编程时，用 DR_SHIFT20（“地址”）替代。
(2) EraseAddr 参数是指向将被擦除的闪存段的地址。对于批量擦除，应该使用信息存储器地址范围中的偶数地址。对于主存储器的擦除，应该使用主存储器地址范围内的偶数地址。

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

(3) 需要正确时序。必须满足 350kHz ±100kHz 的最小和最大 TCLK 频率要求。
(4) 对于 2xx 器件的 INFO 段 A 编程，用 0xA540 取代。

2.3.5.2 5xx 和 6xx 系列的函数参考

参考函数：EraseFLASH_430Xv2 和 EraseFLASH_430Xv2_wo_release

与闪存编程相似，也通过一个载入到目标器件 RAM 中的可执行代码来处理擦除操作。与本文档一同提供的闪存-擦除-代码采用下列参数。

- *EraseAddr*：用于触发闪存操作的虚拟写入的有效闪存地址
- *EraseMode*：写入到闪存控制器模块 FCTL1 的值（主要通过 MERAS 和 ERASE 位来定义擦除模式）
- *FCTL3*：写入到闪存控制器模块 FCTL3 的值（主要确定是否应该设置 LOCKA）

可在 JTAG 控制或者自由运行模式下执行闪存-擦除-代码。与节 2.3.4.2 中描述的相似，采用 JTAG 邮箱系统来检索目标器件中的闪存擦除代码的当前状态。

NOTE

不支持单步执行闪存段擦除操作。任何此类尝试都可能导致程序计数器损坏。

2.3.6 从闪存中读取数据

参考函数：ReadMem 或 ReadMemQuick

使用之前为非闪存存储器地址指定的正常内存读取流程可读取闪存。快速访问方法也可用于读取闪存。

2.3.7 验证目标存储器

参考函数：VerifyMem

使用一个伪签名分析 (PSA) 算法来执行验证，此算法内置于 MSP430 JTAG 逻辑内部并且执行速度约为 23ms/4KB。

- MSP430 和 MSP430X 架构，参考函数：VerifyPSA、VerifyPSA_430X

ExecutePOR	
PSA_CRC = StartAddr-2	
器件具有 EnhancedVerify（请参阅表 2-14）	**器件不具有 EnhancedVerify**（请参阅表 2-14）
`SetPC(StartAddr-4)`	`SetPC(StartAddr-2)`
`HaltCPU`	`SetTCLK`
`ClrTCLK`	`ClrTCLK`
`IR_SHIFT("IR_DATA_16BIT")`	
`DR_SHIFT16(StartAddr-2)`	
`IR_SHIFT("IR_DATA_PSA")`	

Copyright © 2021 Texas Instruments Incorporated

计算 PSA 值：		否
PSA_CRC & 0x8000 == 0x8000	**PSA_CRC & 0x8000 != 0x8000**	
XOR PSA_CRC，具有 0x0805	PSA_CRC <<= 0x1	
PSA_CRC <<= 0x1		
PSA_CRC |= 0x0001		
实际验证还是 EraseCheck？		
验证	**EraseCheck**	
XOR PSA_CRC，具有来自参考数据的下一个字	XOR PSA_CRC，具有 0xFFFF	
通过 PSA 计时：		
SetTCLK		
ClrTCK		
SetTMS		
SetTCK: 选择 DR-Scan		
ClrTCK		
ClrTMS		
SetTCK: Capture-DR		
ClrTCK		
SetTCK: Shift-DR		
ClrTCK		
SetTMS		
SetTCK: Exit-DR		
ClrTCK		
SetTCK		
ClrTMS		
ClrTCK		
SetTCK		
ClrTCLK		
已到达要验证的存储器区域末尾？		
是		
IR_SHIFT("IR_SHIFT_OUT_PSA")		
DR_SHIFT16(0x0000): 读出 PSA 值		
SetTCLK		
如果器件具有增强验证功能，则在此处调用 ReleaseCPU()（请参阅表 2-14）		
将移出的 PSA 值与 PSA_CRC 进行比较		

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

- MSP430Xv2 架构（包括 FRxx 器件），参考函数：VerifyPSA_430Xv2

<table>
<tr><td colspan="3">ExecutePOR</td></tr>
<tr><td colspan="3">PSA_CRC = StartAddr-2</td></tr>
<tr><td colspan="3">SetPC(StartAddr)</td></tr>
<tr><td colspan="3">SetTCLK</td></tr>
<tr><td colspan="3">IR_SHIFT("IR_CNTRL_SIG_16BIT")</td></tr>
<tr><td colspan="3">DR_SHIFT16(0x0501)</td></tr>
<tr><td colspan="3">IR_SHIFT("IR_DATA_16BIT")</td></tr>
<tr><td colspan="3">DR_SHIFT16(StartAddr-2)</td></tr>
<tr><td colspan="3">IR_SHIFT("IR_DATA_PSA")</td></tr>
<tr><td colspan="2">计算 PSA 值：</td><td rowspan="19">否</td></tr>
<tr><td>PSA_CRC & 0x8000 == 0x8000</td><td>PSA_CRC & 0x8000 != 0x8000</td></tr>
<tr><td>XOR PSA_CRC，具有 0x0805</td><td rowspan="3">PSA_CRC <<= 0x1</td></tr>
<tr><td>PSA_CRC <<= 0x1</td></tr>
<tr><td>PSA_CRC |= 0x0001</td></tr>
<tr><td colspan="2">实际验证还是 EraseCheck？</td></tr>
<tr><td>验证</td><td>EraseCheck</td></tr>
<tr><td>XOR PSA_CRC，具有来自参考数据的下一个字</td><td>XOR PSA_CRC，具有 0xFFFF</td></tr>
<tr><td colspan="2">通过 PSA 计时：</td></tr>
<tr><td colspan="2">ClrTCLK</td></tr>
<tr><td colspan="2">ClrTCK</td></tr>
<tr><td colspan="2">SetTMS</td></tr>
<tr><td colspan="2">SetTCK: 选择 DR-Scan</td></tr>
<tr><td colspan="2">ClrTCK</td></tr>
<tr><td colspan="2">ClrTMS</td></tr>
<tr><td colspan="2">SetTCK: Capture-DR</td></tr>
<tr><td colspan="2">ClrTCK</td></tr>
<tr><td colspan="2">SetTCK: Shift-DR</td></tr>
<tr><td colspan="2">ClrTCK</td></tr>
<tr><td colspan="2">SetTMS</td></tr>
<tr><td colspan="2">SetTCK: Exit-DR</td></tr>
<tr><td colspan="2">ClrTCK</td></tr>
<tr><td colspan="2">SetTCK</td></tr>
<tr><td colspan="2">ClrTMS</td></tr>
<tr><td colspan="2">ClrTCK</td></tr>
<tr><td colspan="2">SetTCK</td></tr>
<tr><td colspan="2">SetTCLK</td></tr>
<tr><td colspan="2">已到达要验证的存储器区域末尾？</td></tr>
<tr><td colspan="3">是</td></tr>
<tr><td colspan="3">IR_SHIFT("IR_SHIFT_OUT_PSA")</td></tr>
<tr><td colspan="3">DR_SHIFT16(0x0000)：读出 PSA 值</td></tr>
</table>

Copyright © 2021 Texas Instruments Incorporated

如果器件具有增强验证功能，则在此处调用 ExecutePOR_430Xv2()（请参阅表 2-14）
将移出的 PSA 值与 PSA_CRC 进行比较

2.3.8 FRAM 存储器技术

FRAM 存储器是一款非易失性存储器，其运行方式与标准 SRAM 类似。FRAM 的读取方式与 SRAM 读取方式相类似，并且没有特殊要求。相似地，任何对非保护段的写入方式与 SRAM 中的写入方式类似。

因此，FRAM 写入是破坏性的并且总是要求用读取的信息写回到同一个内存位置。这个写回操作是 FRAM 存储控制器本身的一部分并且无需用户干预。这些写回操作与应用代码或者编程工具执行的正常写入访问不同。

2.3.8.1 写入和读取 FRAM

依照上述说明，WriteMem_430Xv2 和 ReadMem_430Xv2 函数可用于读取和写入 FRAM。WriteMemQuick_430Xv2 和 ReadMemQuick_430Xv2 也适用。

2.3.8.2 擦除 FRAM

FRAM 的一个优势就是在写入内存前无需擦除。如果需要执行“闪存类型”擦除，通过使用写入函数来将 0xFF 写入到存储器的所有位置可完成虚拟擦除。

Replicator430FR 示例工程提供了两种不同的参考设计。对于 FRAM 中的段擦除，请参阅函数 EraseFRAM_430Xv2。FRAM 批量擦除需通过 EraseFRAMViaBootCode_430Xv2 中描述的流程来完成。

2.4 JTAG 访问保护

可以通过多种方法来保护对 MSP 器件的存储器访问。表 2-11 概述了所有可用方法和适用器件系列。与 JTAG 接口直接相关的所有机制都在参考部分中进行了说明。

为完整起见，此列表还包括 BSL 接口的锁定机制。有关这些特性的详细描述和使用说明，请参阅*参考资料*列中列出的文档。

表 2-11. 存储器保护机制概述

保护模式	适用器件	永久锁定	说明	解锁方法	参考资料
JTAG 保险丝	闪存 1xx、2xx 和 4xx 系列（FRxx 和 i20xx 器件除外）	是	向 TEST 引脚（器件没有 TEST 引脚的，则向 TDI 引脚）施加高电压，这会熔断实际的聚合物保险丝并使 JTAG 接口无法使用	无	节 2.4.1
合并 JTAG“软”保险丝（电子保险丝）与禁用 BSL	5xx、6xx 和 FRxx 系列	是	为防止通过 JTAG 或 BSL 接口解锁目标存储器，两者都必须禁用。	无	节 2.4.2.1、节 2.4.2.2 和《MSP430™ 闪存器件引导加载程序 (BSL) 用户指南》

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

表 2-11. 存储器保护机制概述 (continued)

保护模式	适用器件	永久锁定	说明	解锁方法	参考资料
JTAG 无密码锁定	闪存 5xx 和 6xx 系列	否	在地址 0x17FC 写入除 0x00000000 或 0xFFFFFFFF 以外的模式会锁定 JTAG 接口	使用 BSL 接口重置 BSL 存储器中的锁定密钥	节 2.4.2.1
	FR2xx 和 FR4xx 器件	否	在地址 0xFF80 处写入 0x55555555 模式会锁定 JTAG 接口（在 REP430FR 固件中实现的解决方案）。 **NOTE** 注意：对于 FR2xx 和 FR4xx 器件，还可以通过在地址处 0xFF80 写入除 0x00000000 和 0xFFFFFFFF 以外的模式来锁定 JTAG 访问	1.使用 BSL 擦除主存储器 2.使用通过 JTAG 邮箱应用的特殊擦除命令 (User_Code_Erase 01A1A)。在 REP430FR 固件中使用的函数是 EraseFRAMViaBootCode_430Xv2()。	节 2.4.2.2
	FR5xx 和 FR6xx 器件	否		使用 BSL 擦除主存储器	
JTAG 密码锁定	FR5xx 和 FR6xx 器件	否	将 JTAG 锁定签名 (0xAAAA)、密码长度和密码本身写入 0xFF80 和以下存储器段（密码可以扩展到 0xFFFE 处的复位向量）可以锁定 JTAG 接口	将 JTAG 邮箱与器件 BootCode 结合使用，给定的密码将与应用的密码进行比较。	请参阅器件系列用户指南以了解有关如何设置密码的详细信息。 解锁：节 2.4.4.1
存储器保护单元 (MPU)	FRxx 器件	否	用户最多可以指定三个具有不同访问权限（读取、写入、执行）的存储器段	掉电复位 (BOR) 将重置所有 MPU 设置	请参阅器件系列用户指南以了解有关如何设置 MPU 的详细信息。 解锁：节 2.4.5
IP 封装	FR5xx 和 FR6xx 系列（FR57xx 除外）器件	否	用户可以指定一个存储器段的起始地址和结束地址来防止对该段进行任何读取、写入或执行访问	执行“完全擦除”（CCS 中的“erase main, information and IP-protected area on connect”（擦除连接的主段、信息和 IP 保护区域）选项）将擦除整个存储器并重置 IPE 设置	请参阅器件系列用户指南以了解有关如何设置 IP 封装的详细信息。 解锁：节 2.4.6
BSL 密码保护	所有包含 BSL 的器件	否	默认情况下，BSL 会受保护以防止读取和写入	BSL 密码等于中断矢量表内容 – 可以由锁定模式下的简化 BSL 命令集提供	《MSP430™ 闪存器件引导加载程序 (BSL) 用户指南》
禁用 BSL	闪存 1xx、2xx 和 4xx 系列	否	通过向某个地址（取决于 BSL 实现方式）写入锁定签名来禁用 BSL 接口	可以使用 JTAG 接口进行解锁（请参阅器件系列用户指南了解详细信息）	《MSP430™ 闪存器件引导加载程序 (BSL) 用户指南》
	闪存 5xx 和 6xx 系列	否	通过清除 BSL 存储器末尾的某些签名（0x17F6 和 0x17F4）来锁定 BSL 接口	可以通过使用 JTAG 接口擦除和重新编程包含签名的闪存段来恢复签名	《MSP430F5xx 和 MSP430F6xx 系列用户指南》
	FRxx 器件	否	通过在主存储器中的 0xFF84 和 0xFF86 地址处写入给定的 BSL 锁定签名来锁定 BSL 接口	可以通过使用 JTAG 接口来覆盖锁定签名，以重新启用对 BSL 接口的访问	《MSP430FR58xx、MSP430FR59xx 和 MSP430FR6xx 系列用户指南》

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

表 2-11. 存储器保护机制概述 (continued)

保护模式	适用器件	永久锁定	说明	解锁方法	参考资料
通过客户编写的启动代码 (SUC) 进行存储器保护	MSP430i20xx 系列 (MSP430i2040)	取决于 SUC 实现方式	用户可以通过写入启动代码来设置 JTAG 访问保护	取决于 SUC 实现方式	请参阅《MSP430i2xx 系列用户指南》了解详细信息。

2.4.1 正在燃烧 JTAG 保险丝 - 针对 1xx，2xx，4xx 系列的函数参考

根据目标 MSP430 器件系列，描述和执行了两个相似的方法。

所有具有一个 TEST 引脚的器件使用这个输入来应用编程电压，V_{PP}。如前所述，这些器件有函数共用的 JTAG 接口引脚。具有专用 JTAG 接口引脚且引脚数量更多的 MSP430 器件使用 TDI 引脚进行保险丝编程。

带有一个 TEST 引脚的器件：

表 2-12. MSP430 器件 JTAG 接口（共用引脚）

引脚	方向	使用
P1.5/TMS	IN	控制 JTAG 状态机的信号
P1.4/TCK	IN	JTAG 时钟输入
P1.6/TDI	IN	JTAG 数据引脚/TCLK 输入
P1.7/TDO	OUT	JTAG 数据输出
TEST	IN	逻辑高电平启用 JTAG 通信；在对 JTAG 进行编程时的 V_{PP} 输入

无 TEST 引脚的器件（专用 JTAG 引脚）：

表 2-13. MSP430 器件专用 JTAG 接口

引脚	方向	使用
TMS	IN	控制 JTAG 状态机的信号
TCK	IN	JTAG 时钟输入
TDI	IN	JTAG 数据输入/TCLK 输入；在对 JTAG 保险丝进行编程时的 V_{PP} 输入
TDO	输入/输出	JTAG 数据输出；在对 JTAG 保险丝进行编程时的 TDI 输入

NOTE

保险丝编程所需的 V_{PP} 的值可在相应的目标器件数据表中找到。对于现有的闪存器件，V_{PP} 所需的电压为 6.5V±0.5V。

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

2.4.1.1 标准 4 线制 JTAG

参考函数：BlowFuse

2.4.1.1.1 TDI 引脚上的保险丝编程电压 (仅适用于专用 JTAG 引脚器件)

对保险丝进行编程时，通过 TDI 输入施加 V_{PP}。在此模式下，通常在 TDI 上发送的通信数据将通过 TDO 发送。(表 2-13 介绍了 TDI 和 TDO 引脚的双重功能。) 当生成合适的时序来熔断保险丝时，必须将 V_{PP} 源的稳定时间考虑在内。下面的流程详述了 BlowFuse 函数内置的保险丝编程序列。

IR_SHIFT("IR_CNTRL_SIG_16BIT")
DR_SHIFT_IN(0x7201)：将 TDO 配置为 TDI
TDI 信号释放到目标，现在会在 TDO 上提供 TDI。
IR_SHIFT("IR_PREPARE_BLOW")（通过 TDO 引脚）
MsDelay(1)：延迟 1ms
将 V_{PP} 连接至 TDI 引脚
等到 V_{PP} 输入已经稳定 (取决于 V_{PP} 源)
IR_SHIFT("IR_EX_BLOW")：发送到 TDO 上的目标
MsDelay(1)：延迟 1ms
从 TDI 引脚上移除 V_{PP}
将 TDI 引脚切换回 TDI 功能并将 JTAG 状态机复位 (ResetTAP)

2.4.1.1.2 TEST 引脚上的保险丝编程电压

同样的方法用来对用于 TEST 引脚 MSP430 器件的保险丝进行编程，除了熔断电源，V_{PP}，现在被应用于 TEST 输入引脚。

IR_SHIFT("IR_PREPARE_BLOW")
MsDelay(1)：延迟 1ms
将 V_{PP} 连接至 TEST 引脚
等到 VPP 输入已经稳定 (取决于 VPP 源)
IR_SHIFT("IR_EX_BLOW")
MsDelay(1)：延迟 1ms
从 TEST 引脚上移除 V_{PP}
将 JTAG 状态机复位 (ResetTAP)

2.4.1.2 使用 SBW 时的保险丝编程电压

参考函数：BlowFuse_sbw

在 SBW 模式中，TEST/SBWTCK 被用来施加熔断电压 V_{PP}。所需的时序序列显示在图 2-19 中。实际的保险丝编程发生在 TAP 控制器的运行-测试/闲置状态中。通过 SBW 移入 IR_EX_BLOW 指令后，必须再执行一个 TMS_SLOT。然后，必须将一个稳定的 V_{PP} 应用于 SBWTCK。只要 V_{PP} 已稳定到将保险丝熔断，将 SBWTDIO 置为高电平。要求在退出 IR_EX_BLOW 指令移位时 SBWTDIO 为低电平。

Copyright © 2021 Texas Instruments Incorporated

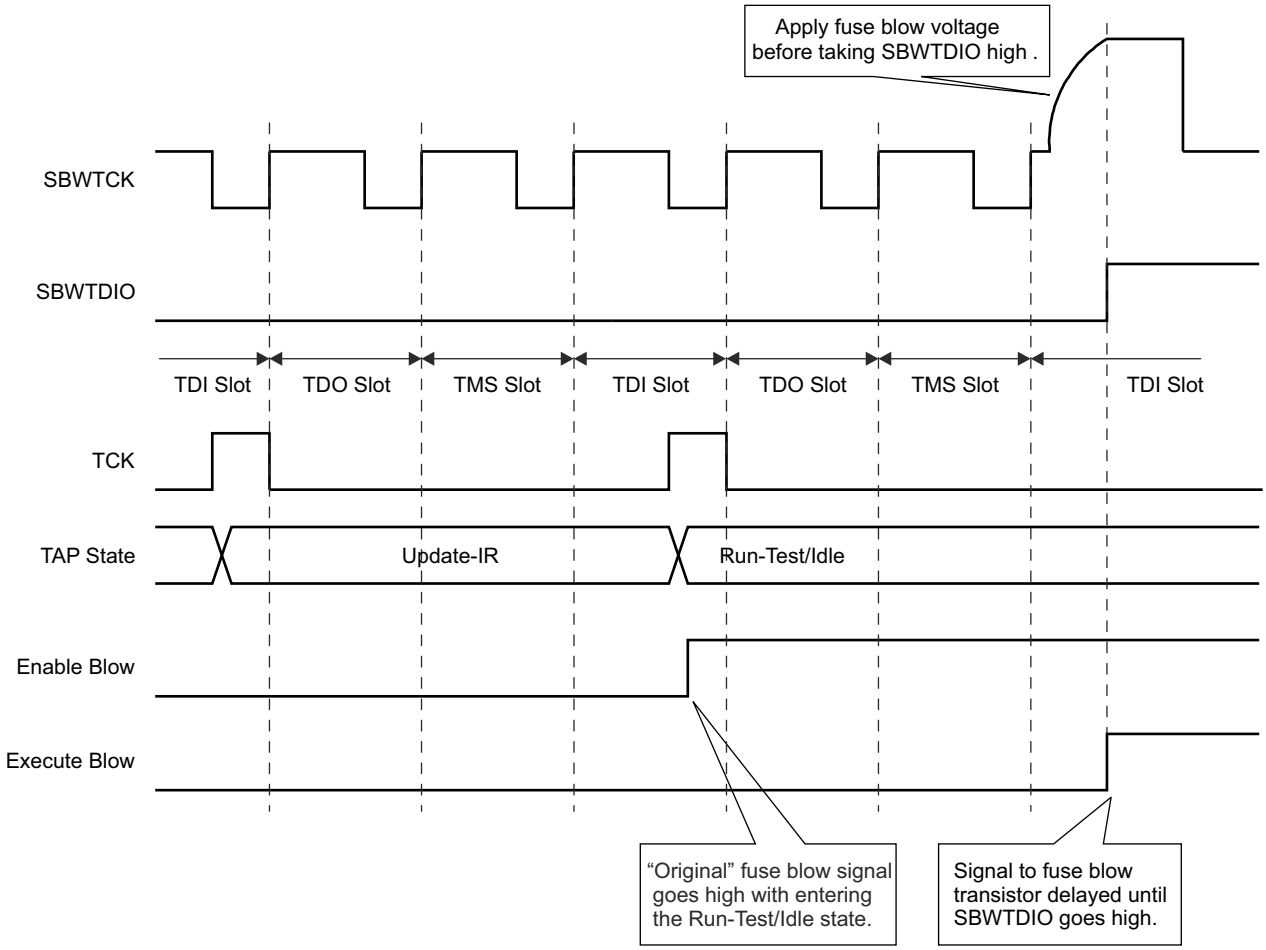

图 2-19. 保险丝熔断时序

2.4.2 对 JTAG 锁定密钥进行编程 - 5xx、6xx 和 FRxx 系列的函数参考

2.4.2.1 闪存器件

参考函数：ProgramLockKey

NOTE

对于 MSP430F5xx 和 MSP430F6xx 系列，无需将特殊高电压施加到器件的 TEST 引脚上。

与 1xx、2xx、4xx 系列需要特殊处理才能熔断 JTAG 安全保险丝不同，5xx 和 6xx 系列的 JTAG 是通过将特定签名编程到器件闪存中的专用地址来锁定的。JTAG 安全锁定密钥位于引导加载程序 (BSL) 存储器的末尾，地址为 0x17FC 至 0x17FF。编辑进这些地址的 0 或者 0xFFFFFFFF 之外的其它值会将 JTAG 接口锁定，而且此锁定不可撤销。所有 5xx 和 6xx MSP430 器件均随附了预编程的 BSL (TI-BSL) 代码，默认情况下，该代码可以保护自身不会遭到意外擦除和写入访问。此功能是通过设置 SYS 模块 SYSBSLC 寄存器中的 SYSBSLPE 位来实现的 (有关详细信息，请参阅《MSP430F5xx 和 MSP430F6xx 系列用户指南》的 *SYS 模块* 一章) 。由于 JTAG 安全锁定密钥位于 BSL 存储器地址范围内，在编辑保护密钥前，必须采取适当的操作来解除对存储器区域的保护。为此，可以直接写入 SYSBSLC 寄存器地址并且将 SYSBSLPE 设置为 0 来进行节 2.3.3.2 中所述的常规存储器写入访问。之后，BSL 存储器像常规闪存一样运行并且可如节 2.3.4.2 中描述的那样在地址 0x17FC 至 0x17FF 上编辑一个 JTAG 锁定密钥。需要使用掉电复位 (BOR) 在启动期间激活 JTAG 安全保护功能。BOR 可如节 2.3.2.2.4 中描述的那样发布。如果硬件设置不允许执行一个开闭循环 (例如，电池已经焊接到印刷电路板 (PCB) 上) ，也可通过写入一个专用 JTAG 测试数据寄存器来生成 BOR。BOR 也会复位 JTAG 接口，这将使器件从 JTAG 控制中释放出来。

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

2.4.2.2 FRAM 存储器器件

参考函数：ProgramLockKey

NOTE

对于 MSP430FR5xx 系列，无需在器件的 TEST 引脚上施加一个特别高电压。

基于 FRAM 的器件使用一个被写入一个特别位置的 LockKey 来保护器件。这些器件支持两个不同的保护级别：仅适用于 FR5xx 和 FR6xx 器件的“受保护模式”以及适用于所有 FRAM 器件的“安全模式”。

在受保护模式中，应用能够定义一个密码并且用这个密码来保护器件。通过采用正确的密码，UnlockDevice 函数可被用于连接至器件（详细信息请见节 2.4.4）。如需了解密码的一般信息，请参阅《MSP430FR57xx 系列用户指南》。

在安全模式下，无法通过 JTAG 访问器件。为了启用安全模式，将 0x55555555 写入到存储器位置 0xFF80。写入密码后，需要一个 BOR 来启用安全保险丝。

2.4.3 针对一个成功受保护器件的测试

参考函数：IsFuseBlown、IsLockKeyProgrammed

在 JTAG 保险丝熔断（对于 1xx、2xx 或 4xx 器件）或已编程 JTAG 锁定密钥（对于 5xx 或 6xx 器件）并且已发出 RESET（通过 JTAG ExecutePOR 命令或硬件中的 $\overline{\text{RST}}$/NMI 引脚来完成）之后，目标 MSP430 上唯一可用的 JTAG 功能是 BYPASS。当目标处于 BYPASS 状态时，从主机发送到目标的数据会延迟一个 TCK 脉冲，然后在 TDO 上输出，而目标 MSP430 下游的其他器件可以在此处接收数据。

为了测试一个受保护的器件，可以尝试访问任一 JTAG 数据寄存器。在下列通信序列中，JTAG CNTRL_SIG 寄存器被访问。

初始化 JTAG 访问 (ResetTAP)	
IR_SHIFT("IR_CNTRL_SIG_CAPTURE")	
DR_SHIFT16(0xAAAA)	
TDO 的输出值 = 0x5555 吗？	
是： *器件受保护*	*否：* *器件不受保护*

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

2.4.4 在受保护模式和安全模式下解锁 FRAM 器件

2.4.4.1 FR5xx 和 FR6xx 器件

支持用户密码保护的器件可通过提供正确的密码来解锁。为了解锁器件，JTAG 邮箱与器件 BootCode 一同使用。为了激活密码解锁机制，必须在器件上应用密码交换请求 (0x1E1E)（请参阅下方的详细序列图）。

NOTE

在执行密码交换请求后，有一个长度仅为 1.2s 的时隙来应用正确的密码。如果未在这个期限内应用密码或者如果密码错误的话，器件将执行一个 BOR 事件。

初始化 JTAG 访问 (ResetTAP) 并将器件保持在 RESET 状态		
IR_Shift(IR_JMB_EXCHANGE)	: 发出 JTAG 邮箱交换请求	
DR_Shift16(0x0011)	: 将 JTAG 邮箱配置为 32 位模式	
DR_Shift16(0xA55A)	: 在执行 BootCode 后使器件进入 LPM4	
DR_Shift16(0x1E1E)	: 执行密码交换请求	
停止 JTAG（释放并运行器件）		
等到 BootCode 被执行并且器件处于 LPM4		
初始化 JTAG 访问 (ResetTAP)		
IR_Shift(IR_JMB_EXCHANGE)	: 发出 JTAG 邮箱交换请求	
DR_Shift16(0x0001)	: 将 JTAG 邮箱配置为 16 位模式	
DR_Shift16(password[i])	: 将 JTAG 邮箱配置为 16 位模式	重复密码长度

2.4.4.2 FR4xx 和 FR2xx 器件

FR4xx 和 FR2xx 器件系列不支持通过用户定义的密码进行 JTAG 保护。这些器件系列只有一个简单的 JTAG 锁定机制。如果将 JTAG 锁定签名写入存储器地址 0xFF80，则会锁定 JTAG 访问。若要解锁这些器件系列，必须通过 JTAG 邮箱应用特殊的擦除命令 (User_Code_Erase 0x1A1A)（请参阅以下详细序列图）。

在器件复位时重新启动器件、初始化 JTAG 并输入“擦除”命令	
ClrTST()	重新启动器件
ClrRST()	重新启动器件
StartJtag(RSTLOW)	重新启动 JTAG – 保持 RST 为低电平，器件不启动
ResetTAP()	使 JTAG TAP 状态机复位
IR_Shift(IR_JMB_EXCHANGE)	发出 JTAG 邮箱交换请求
DR_Shift(0xA55A)	在执行 BootCode 后使器件进入 LPM4
DR_Shift(0x1A1A)	发送器件 UserCode 擦除命令
停止 JTAG（释放器件以便运行）	
等到 BootCode 被执行并且器件处于 LPM4 中	
初始化 JTAG 访问 (Rest Tap)	
调用 SyncJtag_AssertPor() 函数取回 JTAG 控制权	

2.4.5 存储器保护单元处理

如果一个器件有一个存储器保护单元 (MPU)，在擦除或者写入存储器前，此单元必须被禁用。MPU 可以将存储器分成几个不同的部分。每个部分可以有不同的访问权限，例如 READ（读取）、WRITE（写入）和 EXECUTE（执行）。未被禁用的 MPU 会导致不完全的写入或擦除。此外，存储在一个寄存器中的 MPU 设置可能被锁定。为了禁用这个寄存器锁定，必须执行一个 BOR。下面的序列图显示了如何禁用 MPU。可以在函数 DisableMpu430Xv2() 的相关 zip 文件中找到示例代码。

是否已启用 MPU（读取寄存器地址 0x05A0）？	
是	*否*

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

是否已锁定 MPU 设置？				返回
是		否		
IR_Shift(IR_JMB_EXCHANGE)	：发出 JTAG 邮箱交换请求	在地址 0x05A0 上写入 0xA500	：禁用 MPU	
DR_Shift16(0x0001)	：将邮箱配置为 16 位模式	返回	：MPU 被禁用	
DR_Shift16(0xA55A)	：在执行 BootCode 后使器件进入 LPM4			
执行 BOR				
初始化 JTAG 访问 (ResetTAP)				
同步 JTAG 并将 POR 置为有效				
在地址 0x05A0 上写入 0xA500	：禁用 MPU			
返回	：MPU 被禁用			

2.4.6 知识产权封装 (IPE)

除了存储器保护单元 (MPU)，FR59xx、FR58xx、FR69xx 和 FR68xx 器件还具有 IP 封装 (IPE) 模块。IPE 模块使用户能够指定要防止外部访问的单个存储器段。请参阅《MSP430FR58xx、MSP430FR59xx 和 MSP430FR6xx 系列用户指南》，了解有关如何调用 IP 封装功能的信息。

若要删除 IPE 设置，必须完全擦除目标存储器。常规批量擦除不会影响受 IPE 保护的区域，因此必须执行下图详述的特殊擦除序列来重置 IPE 设置。

在器件复位时重新启动器件、初始化 JTAG 并输入“擦除”命令	
ClrTST()	重新启动器件
ClrRST()	重新启动器件
StartJtag(RSTLOW)	重新启动 JTAG – 保持 RST 为低电平，器件不启动
ResetTAP()	使 JTAG TAP 状态机复位
IR_Shift(IR_JMB_EXCHANGE)	发出 JTAG 邮箱交换请求
DR_Shift(0xA55A)	在执行 BootCode 后使器件进入 LPM4
DR_Shift(0x1B1B)	发送“完全擦除”命令
停止 JTAG（释放器件以便运行）	
等到 BootCode 被执行并且器件处于 LPM4 中	
初始化 JTAG 访问 (Rest Tap)	
调用 SyncJtag_AssertPor() 函数取回 JTAG 控制权	

2.4.7 FRAM 写保护

在 MSP430FR2xx 和 MSP430FR4xx 器件上，FRAM 受到保护以防止不必要的访问。若要启用 FRAM 访问，请清除 SYSCFG0 寄存器中的 DFWP 和 PFWP 位。在这些系列的某些器件上，SYSCFG0 寄存器受密码保护。更多详细信息，请参阅器件系列用户指南。参考函数：DisableMPU_430Xv2。

Copyright © 2021 Texas Instruments Incorporated

2.5 JTAG 函数原型

2.5.1 低电平 JTAG 函数

static word IR_Shift (byte Instruction)

通过 TDI，将一个新的指令移入 JTAG 指令寄存器。（这个指令被首先移入 MSB； JTAG 指令寄存器将 MSB 解释为 LSB。）

参数： 字节指令（8 位 JTAG 指令）

结果： 字 TDOword（值从 TDO=JTAG_ID 上移出）

static word DR_Shift16 (word Data)

通过 TDI 将一个指定的 16 位字移入 JTAG 数据寄存器（数据移位 MSB 在前）

参数： 字数据（16 位数据值）

结果： 字（同时在 TDO 上移出的值）

static void ResetTAP (void)

执行保险丝熔断检查，复位 JTAG 接口，并且将 JTAG 状态机（TAP 控制器）置于运行-测试/闲置状态

参数： 无

结果： 无

static word ExecutePOR (void)

通过 JTAG 控制信号寄存器执行一个上电清除命令。这个函数也禁用目标寄存器的看门狗定时器以避免达到自动复位条件。

参数： 无

结果： 字（如果被查询的 JTAG ID 有效，为 STATUS_OK，否则为 STATUS_ERROR）

static word SetInstrFetch (void)

将目标器件的 CPU 置于取指令状态

参数： 无

结果： 字（如果取指令状态被设定，为 STATUS_OK，否则为 STATUS_ERROR）

static void SetPC (word Addr)

将所需的 16 位地址载入到目标器件 CPU 的程序计数器 (PC)

参数： 字 Addr（所需的 16 位 PC 值）

结果： 无

static void HaltCPU (void)

将目标 CPU 置于一个受控的、被停止状态

参数： 无

结果： 无

static void ReleaseCPU (void)

将目标器件的 CPU 从受控的、被停止状态中释放（不要将目标器件从 JTAG 控制中释放。请见 ReleaseDevice。）

参数： 无

结果： 无

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

word VerifyPSA (word StartAddr, word Length, word *DataArray)

将计算得出的伪签名分析 (PSA) 值与从目标器件中移出的 PSA 值相比较。它可被用于最后一个数据块或者擦除验证（由之前讨论的 EraseCheck 和 VerifyMem 原型调用）。

参数： 字 StartAddr（内存数据块的起始地址将被检查）

字长度（数据块内字的数量）

字 *DataArray（指向一个包含数据的阵列的指针，0 为擦除检查）

结果： 字（如果比较成功，为 STATUS_OK，否则为 STATUS_ERROR）

对于 MSP430X 和 MSP430Xv2 架构器件，定义了以下函数：

static unsigned long DR_Shift20(unsigned long address)

通过 TDI 将给定的 20 位地址字移入 JTAG 数据寄存器（数据移位 MSB 在前）

参数： 长地址（20 位地址字）

结果： 长 TDO 值

对于 MSP430Xv2 架构器件，定义了以下函数：

word GetCoreipIdXv2()

确定和比较内核标识信息 (Xv2)

参数： 无

结果： 字（如果返回正确的 JTAG ID，为 STATUS_OK，否则为 STATUS_ERROR）

void jResetJtagTap(void)

使目标 JTAG 接口复位并执行保险丝硬件检查

参数： 无

结果： 无

void StartJtagJSbw(byte states)

在 JSBW 模式下开始 JTAG 通信

参数： 字节状态（复位状态）

结果： 无

void jRelease(void)

释放 JSBW 逻辑

参数： 无

结果： 无

long jsbw_Shift(word Format, long Data)

将值移入 TDI（MSB 在前）并同时从 TDO 移出值（MSB 在前）

参数： 字格式（移位的位数，即 8 (F_BYTE)、16 (F_WORD)、20 (F_ADDR) 或 32 (F_LONG)）

长数据（要移入 TDI 的数据）

结果： 无符号长整型（扫描的 TDO 值）

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

long jsbw_IR_Shift(byte instruction)

通过 JSBW 将新指令移入 JTAG 指令寄存器（MSB 在前，但 MSB - LSB 互换，便于使用相同的移位函数 Shift()）。

参数： 指令（8 位 JTAG 指令）

结果： 字（TDOword - 从 TDO 移出的值：JTAG 标识）

long jsbw_DR_Shift(long data)

通过 JSBW 将数据移入 JTAG 数据寄存器（MSB 在前，但 MSB - LSB 互换，便于使用相同的移位函数 Shift()）。

参数： 长数据

结果： 字（TDOword - 从 TDO 移出的值：JTAG 标识）

void JsbwMagicPattern(void)

通过 JSBW 应用魔法模式

参数： 无

结果： 无

void jsbwJtagUnlock(void)

通过 JSBW 使 JTAG 锁定复位

参数： 无

结果： 无

2.5.2 高级 JTAG 例程

字 GetDevice（空）

将目标 MSP430 器件置于 JTAG 控制之下将目标器件的 CPU 看门狗设定为一个保持状态；设定全局 DEVICE 变量。

自变量： 无

结果： word（如果保险丝被熔断，为 STATUS_ERROR，JTAG_ID 不正确（不等于 0x89 或者发生同步超时；否则为 STATUS_OK）

空 ReleaseDevice（字地址）

将目标器件从 JTAG 控制中释放；CPU 开始在指定的 PC 地址上执行

自变量： 字 Addr（0xFFFE：执行复位；被载入到 PC 中的复位矢量上的地址；否则地址由载入到 PC 中的 Addr 指定）

结果： 无

空 WriteMem（字格式、字地址、字数据）

将一个单字节或者字写入到指定的地址（只适用于 RAM / 外设）

自变量： 字格式（F_BYTE 或 F_WORD）

字 Addr（针对将被写入的目标地址）

字数据（将被写入的数据值）

结果： 无

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

空 WriteMemQuick (字 StartAddr , 字长度 , 字 *DataArray)

将一个字阵列写入目标器件存储器 (只适用于 RAM / 外设)

自变量： 字 StartAddr (目标存储器的起始地址)

字长度 (被编程的字的数量)

字 *DataArray (指向包含数据的阵列的指针)

结果： 无

空 WriteFLASH (字 StartAddr , 字长度 , 字 *DataArray)

使用目标器件的闪存控制器来编程/验证写入到闪存中的字阵列

自变量： 字 StartAddr (目标闪存的起始地址)

字长度 (被编程的字的数量)

字 *DataArray (指向包含数据的阵列的指针)

结果： 无

word WriteFLASHallSections(const unsigned int *data, const unsigned long *address, const unsigned long *length_of_sections, const unsigned long sections)

使用 WriteFLASH() 函数来编程/验证向闪存写入的一组字数据数组。符合文件 Target_Code_(IDE).s43 或 Target_Code.h 的 CodeArray 结构约定。

参数： const unsigned int *DataArray (指向包含数据的数组的指针)

const unsigned long *address (指向包含起始地址的数组的指针)

const unsigned long *length_of_sections (指向包含字数 (从起始地址开始计数) 的数组的指针)

const unsigned long sections (代码文件中的分段数)

结果： word (如果验证成功，为 STATUS_OK，否则为 STATUS_ERROR)

字 ReadMem (字格式、字地址)

从一个指定目标存储器地址读取一个字节或字

自变量： 字格式 (F_BYTE 或 F_WORD)

字 Addr (将读取针对数据的目标地址)

结果： 字 (存储在目标地址存储器位置的数据值)

空 ReadMemQuick (字 StartAddr , 字长度 , 字 *DataArray)

从目标存储器读取一个字阵列

自变量： 字 StartAddr (将被读取的目标存储器的起始地址)

字长度 (被读取的字的数量)

字 *DataArray (指向数据存储阵列的指针)

结果： 无

Copyright © 2021 Texas Instruments Incorporated

空 EraseFLASH (字 EraseMode , 字 EraseAddr)

执行一个批量擦除 (具有或没有信息存储器) 或者一个由给定模式和地址指定的闪存模块的段擦除

自变量： 字 EraseMode (ERASE_MASK , ERASE_MAIN 或者 ERASE_SGMT)

字 EraseAddr (任何所选段中将被擦除的地址)

结果： 无

字 EraseCheck (字 StartAddr , 字长度)

使用 VerifyPSA 函数在给定的存储器范围内执行擦除检查

自变量： 字 StartAddr (将被检查的存储器的起始地址)

字长度 (将被检查的字的数量)

结果： word (如果检查成功，为 STATUS_OK，否则为 STATUS_ERROR)

字 VerifyMem (字 StartAddr , 字长度 , 字 *DataArray)

在指定的存储器范围内执行一个程序验证

自变量： 字 StartAddr (将被验证的存储器的起始地址)

字长度 (将被验证的字的数量)

字 *DataArray (指向包含数据的阵列的指针)

结果： word (如果验证成功，为 STATUS_OK，否则为 STATUS_ERROR)

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

对于 MSP430 和 MSP430X 架构器件，定义了以下函数：

字 BlowFuse（空）

编程（或者熔断）JTAG 接口访问安全保险丝。这个函数也使用 IsFuseBlown() 原型来检查一个已成功编程的保险丝。

自变量： 无

结果： word（如果保险丝熔断成功，为 STATUS_OK，否则为 STATUS_ERROR）

字 IsFuseBlown（空）

确定目标器件上的安全保险丝是否已经被编程。

自变量： 无

结果： word（如果保险丝被熔断，为 STATUS_OK，否则为 STATUS_ERROR）

void UnlockInfoA(void)

解锁 InfoMemory（闪存）的 A 段

自变量： 无

结果： 无

对于 MSP430Xv2 架构（闪存和 FRAM 器件），定义了以下函数：

word ProgramLockKey(void)

禁用对目标器件的 JTAG 访问。

自变量： 无

结果： word（如果保险丝熔断成功，为 TRUE，否则为 FALSE）

word IsLockKeyProgrammed(void)

检查是否已编程 JTAG 锁定密钥。

自变量： 无

结果： word（如果保险丝被熔断，为 STATUS_OK，否则为 STATUS_ERROR）

对于 MSP430Xv2 架构（仅限闪存器件），定义了以下函数：

void UnlockInfoA_430Xv2(void)

解锁 InfoMemory（闪存）的 A 段。

自变量： 无

结果： 无

void UnlockBsl_430Xv2Flash(void)

解锁 BSL 存储器保护。

自变量： 无

结果： 无

Copyright © 2021 Texas Instruments Incorporated

对于 MSP430Xv2 架构（仅限 FRAM 器件），定义了以下函数：

word UnlockDevice_430Xv2(unsigned short* password, unsigned long passwordLength)

在设置了 JTAG 密码时解锁 FRAM 存储器

参数： unsigned short* password（指向包含 JTAG 密码的数组的指针）

unsigned long passwordLength（密码的长度，以字为单位）

结果： word（如果存储器解锁成功，为 STATUS_OK，否则为 STATUS_ERROR）

void EraseFRAM_430Xv2(unsigned long StartAddr, unsigned long Length)

对用户定义的 FRAM 存储器区段执行擦除。对于 FRAM 器件，擦除操作相当于对相应存储器区段执行 0xFFFF 写操作。（可以通过 PSA 进行“擦除检查”扩展）此函数应仅用于“段擦除”。对于“批量擦除”，请考虑改用 EraseFRAMViaBootCode_430Xv2。

参数： word StartAddr（擦除的起始地址）

word Length（存储器区段的长度，以字为单位）

结果： 无

word DisableMpu_430Xv2(void)

禁用存储器保护单元（仅限 FRAM 器件）

自变量： 无

结果： word（如果成功禁用 MPU，为 STATUS_OK，否则为 STATUS_ERROR）

word DisableFramWprod_430Xv2(void)

禁用存储器写保护（仅限 FRAM 器件 FR6047、FR5994）

自变量： 无

结果： word（如果成功禁用存储器写保护，为 STATUS_OK，否则为 STATUS_ERROR）

void UnlockBsl_430Xv2FRAM(void)

解锁 BSL 存储器保护。

自变量： 无

结果： 无

word EraseFRAMViaBootCode_430Xv2(word mailBoxMode, word data1, word data2)

使用 JTAG 邮箱执行 FRxx 器件的擦除。

参数： word mailBoxMode 32 位 16 位模式

word data1 邮箱数据 - 第一个 16 位

word data2 邮箱数据 - 第二个 16 位

结果： word（如果擦除成功，为 STATUS_OK，否则为 STATUS_ERROR）

short DownloadProgram(struct_Program* program)

此函数下载转换后的 MSP430.txt 文件

参数： struct_Program* 程序结构，包含可执行代码和存储器数据

结果： word（如果验证成功，为 STATUS_OK，否则为 STATUS_ERROR）

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

short DownloadMsp430Code()

此函数可配置将程序下载到目标存储器时所需的所有信息。配置完成后会调用 DownloadProgram() 函数。

自变量： 无

结果： 无

2.6 器件系列的 JTAG 特性

表 2-14. MSP430F1xx、MSP430F2xx、MSP430F4xx、MSP430Gxx JTAG 特性

器件(1) (2)	0x0FF0 上的器件 ID	0x0FF1 上的器件 ID	0x0FFD 上的器件 ID	测试引脚	CPUX	对 SRAM 进行 Data Quick 访问(3)	对闪存进行 Data Quick 访问(4)	Fast Flash(5)	Enh Verify(6)	Spy-Bi-Wire
AFE2xx	0x02	0x53	-	真	假	读取/写入	读取	真	假	真
F11x1(A)	0xF1	0x12	-	真	假	读取/写入	读取	假	假	假
F11x2	0x11	0x32	-	真	假	读取/写入	读取	假	假	假
F12x	0xF1	0x23	-	真	假	假	读取	假	假	假
F12x2	0x12	0x32	-	真	假	读取/写入	读取	假	假	假
F13x，F14x，F14x1	0xF1	0x49	-	假	假	读取/写入	读取	假	假	假
F15x，F16x	0xF1	0x69	-	假	假	读取/写入	读取	假	假	假
F161x	0xF1	0x6C	-	假	假	读取/写入	读取	假	假	假
F20x1	0xF2	0x01	0x01	真	假	读取/写入	读取	真	假	真
F20x2	0xF2	0x01	0x02	真	假	读取/写入	读取	真	假	真
F20x3	0xF2	0x01	0x03	真	假	读取/写入	读取	真	假	真
F21x1	0xF2	0x13	0x01	真	假	读取/写入	读取	假	真	假
F21x2	0xF2	0x13	0x02	真	假	读取/写入	读取	真	真	真
F22x2、F22x4、G2x44	0xF2	0x27	-	真	假	读取/写入	读取	真	真	真
F23x，F24x，F24x1，F2410	0xF2	0x49	-	假	假	读取/写入	读取	真	真	假
F23x0	0xF2	0x37	-	真	假	读取/写入	读取	真	真	假
F241x，F261x	0xF2	0x6F	-	假	真	读取/写入	读取	真	真	假
F412、F413	0xF4	0x13	-	假	假	假	读取	假	假	假
F415、F417	0xF4	0x27	'W'	假	假	读取/写入	读取	假	假	假
F41x2	0x41	0x52	-	真	假	读取/写入	读取	真	真	真
F(E)42x	0xF4	0x27	'E'	假	假	读取/写入	读取	假	假	假
F(E)42xA	0x42	0x7A	'E'	假	假	读取/写入	读取	真	假	假
F(G)42x0	0xF4	0x27	'G'	假	假	读取/写入	读取	假	假	假
F43x（80 引脚）	0xF4	0x37	-	假	假	读取/写入	读取	假	假	假
F43x，F44x（100 引脚）	0xF4	0x49	-	假	假	读取/写入	读取	假	假	假
F471xx	0xF4	0x7F	-	假	真	读取/写入	读取	真	真	假
F47xx	0xF4	0x49	0x02	假	假	读取/写入	读取	真	真	假
FE42x2	0x42	0x52	'E'	假	假	读取/写入	读取	假	假	假
FG43x	0xF4	0x39	'G'	假	假	读取/写入	读取	假	假	假

Copyright © 2021 Texas Instruments Incorporated

表 2-14. MSP430F1xx、MSP430F2xx、MSP430F4xx、MSP430Gxx JTAG 特性 (continued)

器件(1) (2)	0x0FF0 上的器件 ID	0x0FF1 上的器件 ID	0x0FFD 上的器件 ID	测试引脚	CPUX	对 SRAM 进行 Data Quick 访问(3)	对闪存进行 Data Quick 访问(4)	Fast Flash(5)	Enh Verify(6)	Spy-Bi-Wire
F(G)461x，F461x1	0xF4	0x6F	'G'	假	真	读取/写入	读取	真	真	假
F(G)47x	0xF4	0x79	'G'	假	假	读取/写入	读取	真	真	假
FW428	0xF4	0x29	'W'	假	假	读取/写入	读取	假	假	假
FW429	0xF4	0x29	'W'	假	假	读取/写入	读取	假	假	假
FW42x	0xF4	0x27	'W'	假	假	读取/写入	读取	假	假	假
G2x01、G2x11	0xF2	0x01	0x01	真	假	读取/写入	读取	真	假	真
G2x21、G2x31	0xF2	0x01	0x02	真	假	读取/写入	读取	真	假	真
G2xx2	0x24	0x52	-	真	假	读取/写入	读取	真	假	真
G2xx3	0x25	0x53	-	真	假	读取/写入	读取	真	假	真
G2xx5	0x29	0x55	-	真	假	读取/写入	读取	真	假	真
TCH5E	0x25	0x5C	-	真	假	读取/写入	读取	真	假	真

(1) 此表中列出的所有器件都具有 JTAG ID 0x89。
(2) 此表中列出的所有器件都支持 4 线制 JTAG。
(3) 对 SRAM 进行 DataQuick 访问：如果是读取/写入，则器件支持使用 IR_DATA_QUICK 指令在快速模式下读写 SRAM（请参阅节 2.3.3.3）。某些器件可能仅支持此指令用于读取操作或仅支持用于写入操作。
(4) 对闪存进行 DataQuick 访问：如果是读取/写入，则器件支持使用 IR_DATA_QUICK 指令在快速模式下读写闪存（请参阅节 2.3.3.3）。某些器件可能仅支持此指令用于读取操作或仅支持用于写入操作。
(5) FastFlash：如果为 TRUE，器件有一个 20ms 的累积擦除时间 ($t_{CMErase}$)；如果为 FALSE，$t_{CMErase}$ 为 200ms（请参阅节 2.3.5.1.2）。
(6) EnhVerify：如果为 TRUE，器件支持一种更高级的存储器内容验证机制（PSA 校验和计算）。如果为 FALSE，器件的 CPU 在 PSA 校验和算法执行期间仍然在后台工作。因此，必须在计算校验和之后执行 POR。通过增强的 PSA 硬件实现，CPU 在计算校验和的过程中完全停止，且计算后不需要 POR。

表 2-15. MSP430F5xx、MSP430F6xx、CC430 JTAG 特性

器件	0x1A04 上的器件 ID	0x1A05 上的器件 ID	对 SRAM 进行 Data Quick 访问(3)	对闪存进行 Data Quick 访问(4)
CC430F5123	0x3C	0x81	读取/写入	读取
CC430F5125	0x3B	0x81	读取/写入	读取
CC430F5133	0x51	0x33	读取/写入	读取
CC430F5135	0x51	0x35	读取/写入	读取
CC430F5137	0x51	0x37	读取/写入	读取
CC430F5143	0x3A	0x81	读取/写入	读取
CC430F5145	0x39	0x81	读取/写入	读取
CC430F5147	0x38	0x81	读取/写入	读取
CC430F6125	0x61	0x25	读取/写入	读取
CC430F6126	0x61	0x26	读取/写入	读取
CC430F6127	0x61	0x27	读取/写入	读取
CC430F6135	0x61	0x35	读取/写入	读取
CC430F6137	0x61	0x37	读取/写入	读取
CC430F6143	0x37	0x81	读取/写入	读取
CC430F6145	0x36	0x81	读取/写入	读取
CC430F6147	0x35	0x81	读取/写入	读取
MSP430F5131	0x26	0x80	读取/写入	读取
MSP430F5132	0x28	0x80	读取/写入	读取

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

表 2-15. MSP430F5xx、MSP430F6xx、CC430 JTAG 特性 (continued)

器件	0x1A04 上的器件 ID	0x1A05 上的器件 ID	对 SRAM 进行 Data Quick 访问(3)	对闪存进行 Data Quick 访问(4)
MSP430F5151	0x2A	0x80	读取/写入	读取
MSP430F5152	0x2C	0x80	读取/写入	读取
MSP430F5171	0x2E	0x80	读取/写入	读取
MSP430F5172	0x30	0x80	读取/写入	读取
MSP430F5212	0x40	0x81	读取/写入	读取
MSP430F5213	0x41	0x81	读取/写入	读取
MSP430F5214	0x42	0x81	读取/写入	读取
MSP430F5217	0x45	0x81	读取/写入	读取
MSP430F5218	0x46	0x81	读取/写入	读取
MSP430F5219	0x47	0x81	读取/写入	读取
MSP430F5222	0x4A	0x81	读取/写入	读取
MSP430F5223	0x4B	0x81	读取/写入	读取
MSP430F5224	0x4C	0x81	读取/写入	读取
MSP430F5227	0x4F	0x81	读取/写入	读取
MSP430F5228	0x50	0x81	读取/写入	读取
MSP430F5229	0x51	0x81	读取/写入	读取
MSP430F5232	0xFA	0x81	读取/写入	读取
MSP430F5234	0xF9	0x81	读取/写入	读取
MSP430F5237	0xF8	0x81	读取/写入	读取
MSP430F5239	0xF7	0x81	读取/写入	读取
MSP430F5242	0xF6	0x81	读取/写入	读取
MSP430F5244	0xF5	0x81	读取/写入	读取
MSP430F5247	0xF4	0x81	读取/写入	读取
MSP430F5249	0xF3	0x81	读取/写入	读取
MSP430F5252	0x06	0x82	读取/写入	读取
MSP430F5253	0x05	0x82	读取/写入	读取
MSP430F5254	0x04	0x82	读取/写入	读取
MSP430F5255	0x03	0x82	读取/写入	读取
MSP430F5256	0x02	0x82	读取/写入	读取
MSP430F5257	0x01	0x82	读取/写入	读取
MSP430F5258	0x00	0x82	读取/写入	读取
MSP430F5259	0xFF	0x81	读取/写入	读取
MSP430F5304	0x12	0x81	读取/写入	读取
MSP430F5308	0x13	0x81	读取/写入	读取
MSP430F5309	0x14	0x81	读取/写入	读取
MSP430F5310	0x15	0x81	读取/写入	读取
MSP430F5324	0x16	0x81	读取/写入	读取
MSP430F5325	0x17	0x81	读取/写入	读取
MSP430F5326	0x18	0x81	读取/写入	读取
MSP430F5327	0x19	0x81	读取/写入	读取
MSP430F5328	0x1A	0x81	读取/写入	读取
MSP430F5329	0x1B	0x81	读取/写入	读取
MSP430F5333	0x25	0x81	读取/写入	读取

Copyright © 2021 Texas Instruments Incorporated

表 2-15. MSP430F5xx、MSP430F6xx、CC430 JTAG 特性 (continued)

器件	0x1A04 上的器件 ID	0x1A05 上的器件 ID	对 SRAM 进行 Data Quick 访问(3)	对闪存进行 Data Quick 访问(4)
MSP430F5335	0x27	0x81	读取/写入	读取
MSP430F5336	0x28	0x81	读取/写入	读取
MSP430F5338	0x2A	0x81	读取/写入	读取
MSP430F5340	0x1C	0x81	读取/写入	读取
MSP430F5341	0x1D	0x81	读取/写入	读取
MSP430F5342	0x1E	0x81	读取/写入	读取
MSP430F5357	0x34	0x81	读取/写入	读取
MSP430F5358	0x33	0x81	读取/写入	读取
MSP430F5359	0x32	0x81	读取/写入	读取
MSP430F5418	0x54	0x18	读取/写入	读取
MSP430F5418A	0x00	0x80	读取/写入	读取
MSP430F5419	0x54	0x19	读取/写入	读取
MSP430F5419A	0x01	0x80	读取/写入	读取
MSP430F5435	0x54	0x35	读取/写入	读取
MSP430F5435A	0x02	0x80	读取/写入	读取
MSP430F5436	0x54	0x36	读取/写入	读取
MSP430F5436A	0x03	0x80	读取/写入	读取
MSP430F5437	0x54	0x37	读取/写入	读取
MSP430F5437A	0x04	0x80	读取/写入	读取
MSP430F5438	0x54	0x38	读取/写入	读取
MSP430F5438A	0x05	0x80	读取/写入	读取
MSP430F5500	0x3B	0x80	读取/写入	读取
MSP430F5501	0x32	0x80	读取/写入	读取
MSP430F5502	0x33	0x80	读取/写入	读取
MSP430F5503	0x34	0x80	读取/写入	读取
MSP430F5504	0x35	0x80	读取/写入	读取
MSP430F5505	0x36	0x80	读取/写入	读取
MSP430F5506	0x37	0x80	读取/写入	读取
MSP430F5507	0x38	0x80	读取/写入	读取
MSP430F5508	0x39	0x80	读取/写入	读取
MSP430F5509	0x3A	0x80	读取/写入	读取
MSP430F5510	0x31	0x80	读取/写入	读取
MSP430F5513	0x55	0x13	读取/写入	读取
MSP430F5514	0x55	0x14	读取/写入	读取
MSP430F5515	0x55	0x15	读取/写入	读取
MSP430F5517	0x55	0x17	读取/写入	读取
MSP430F5519	0x55	0x19	读取/写入	读取
MSP430F5521	0x55	0x21	读取/写入	读取
MSP430F5522	0x55	0x22	读取/写入	读取
MSP430F5524	0x55	0x24	读取/写入	读取
MSP430F5525	0x55	0x25	读取/写入	读取
MSP430F5526	0x55	0x26	读取/写入	读取
MSP430F5527	0x55	0x27	读取/写入	读取

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

表 2-15. MSP430F5xx、MSP430F6xx、CC430 JTAG 特性 (continued)

器件	0x1A04 上的器件 ID	0x1A05 上的器件 ID	对 SRAM 进行 Data Quick 访问(3)	对闪存进行 Data Quick 访问(4)
MSP430F5528	0x55	0x28	读取/写入	读取
MSP430F5529	0x55	0x29	读取/写入	读取
MSP430F5630	0x3C	0x80	读取/写入	读取
MSP430F5631	0x3E	0x80	读取/写入	读取
MSP430F5632	0x40	0x80	读取/写入	读取
MSP430F5633	0x42	0x80	读取/写入	读取
MSP430F5634	0x44	0x80	读取/写入	读取
MSP430F5635	0x0E	0x80	读取/写入	读取
MSP430F5636	0x10	0x80	读取/写入	读取
MSP430F5637	0x12	0x80	读取/写入	读取
MSP430F5638	0x14	0x80	读取/写入	读取
MSP430F5658	0x31	0x81	读取/写入	读取
MSP430F5659	0x30	0x81	读取/写入	读取
MSP430F6433	0x1F	0x81	读取/写入	读取
MSP430F6435	0x21	0x81	读取/写入	读取
MSP430F6436	0x22	0x81	读取/写入	读取
MSP430F6438	0x24	0x81	读取/写入	读取
MSP430F6458	0x2E	0x81	读取/写入	读取
MSP430F6459	0x2D	0x81	读取/写入	读取
MSP430F6630	0x46	0x80	读取/写入	读取
MSP430F6631	0x48	0x80	读取/写入	读取
MSP430F6632	0x4A	0x80	读取/写入	读取
MSP430F6633	0x4C	0x80	读取/写入	读取
MSP430F6634	0x4E	0x80	读取/写入	读取
MSP430F6635	0x16	0x80	读取/写入	读取
MSP430F6636	0x18	0x80	读取/写入	读取
MSP430F6637	0x1A	0x80	读取/写入	读取
MSP430F6638	0x1C	0x80	读取/写入	读取
MSP430F6658	0x2C	0x81	读取/写入	读取
MSP430F6659	0x2B	0x81	读取/写入	读取
MSP430F6700	0x54	0x80	读取/写入	读取
MSP430F6701	0x55	0x80	读取/写入	读取
MSP430F6702	0x56	0x80	读取/写入	读取
MSP430F6703	0x57	0x80	读取/写入	读取
MSP430F6720	0x58	0x80	读取/写入	读取
MSP430F6720A	0x76	0x82	读取/写入	读取
MSP430F6721	0x59	0x80	读取/写入	读取
MSP430F6721A	0x77	0x82	读取/写入	读取
MSP430F6722	0x60	0x80	读取/写入	读取
MSP430F6723	0x61	0x80	读取/写入	读取
MSP430F6723A	0x79	0x82	读取/写入	读取
MSP430F6724	0x6D	0x81	读取/写入	读取
MSP430F6724A	0x7A	0x82	读取/写入	读取

Copyright © 2021 Texas Instruments Incorporated

表 2-15. MSP430F5xx、MSP430F6xx、CC430 JTAG 特性 (continued)

器件	0x1A04 上的器件 ID	0x1A05 上的器件 ID	对 SRAM 进行 Data Quick 访问[3]	对闪存进行 Data Quick 访问[4]
MSP430F6725	0x6E	0x81	读取/写入	读取
MSP430F6725A	0x7B	0x82	读取/写入	读取
MSP430F6726	0x6F	0x81	读取/写入	读取
MSP430F6726A	0x7C	0x82	读取/写入	读取
MSP430F6730	0x62	0x80	读取/写入	读取
MSP430F6730A	0x80	0x82	读取/写入	读取
MSP430F6731	0x63	0x80	读取/写入	读取
MSP430F6731A	0x81	0x82	读取/写入	读取
MSP430F6732	0x64	0x80	读取/写入	读取
MSP430F6733	0x65	0x80	读取/写入	读取
MSP430F6733A	0x83	0x82	读取/写入	读取
MSP430F6734	0x6A	0x81	读取/写入	读取
MSP430F6734A	0x84	0x82	读取/写入	读取
MSP430F6735	0x6B	0x81	读取/写入	读取
MSP430F6735A	0x85	0x82	读取/写入	读取
MSP430F6736	0x6C	0x81	读取/写入	读取
MSP430F6736A	0x86	0x82	读取/写入	读取
MSP430F6745	0x88	0x81	读取/写入	读取
MSP430F67451	0x97	0x81	读取/写入	读取
MSP430F67451A	0x25	0x82	读取/写入	读取
MSP430F6745A	0x16	0x82	读取/写入	读取
MSP430F6746	0x89	0x81	读取/写入	读取
MSP430F67461	0x98	0x81	读取/写入	读取
MSP430F67461A	0x26	0x82	读取/写入	读取
MSP430F6746A	0x17	0x82	读取/写入	读取
MSP430F6747	0x8A	0x81	读取/写入	读取
MSP430F67471	0x99	0x81	读取/写入	读取
MSP430F67471A	0x27	0x82	读取/写入	读取
MSP430F6747A	0x18	0x82	读取/写入	读取
MSP430F6748	0x8B	0x81	读取/写入	读取
MSP430F67481	0x9A	0x81	读取/写入	读取
MSP430F67481A	0x28	0x82	读取/写入	读取
MSP430F6748A	0x19	0x82	读取/写入	读取
MSP430F6749	0x8C	0x81	读取/写入	读取
MSP430F67491	0x9B	0x81	读取/写入	读取
MSP430F67491A	0x29	0x82	读取/写入	读取
MSP430F6749A	0x1A	0x82	读取/写入	读取
MSP430F67621	0x38	0x82	读取/写入	读取
MSP430F67621A	0x87	0x82	读取/写入	读取
MSP430F67641	0x39	0x82	读取/写入	读取
MSP430F67641A	0x88	0x82	读取/写入	读取
MSP430F6765	0x8D	0x81	读取/写入	读取
MSP430F67651	0x9C	0x81	读取/写入	读取

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

表 2-15. MSP430F5xx、MSP430F6xx、CC430 JTAG 特性 (continued)

器件	0x1A04 上的器件 ID	0x1A05 上的器件 ID	对 SRAM 进行 Data Quick 访问(3)	对闪存进行 Data Quick 访问(4)
MSP430F67651A	0x2A	0x82	读取/写入	读取
MSP430F6765A	0x1B	0x82	读取/写入	读取
MSP430F6766	0x8E	0x81	读取/写入	读取
MSP430F67661	0x9D	0x81	读取/写入	读取
MSP430F67661A	0x2B	0x82	读取/写入	读取
MSP430F6766A	0x1C	0x82	读取/写入	读取
MSP430F6767	0x8F	0x81	读取/写入	读取
MSP430F67671	0x9E	0x81	读取/写入	读取
MSP430F67671A	0x2C	0x82	读取/写入	读取
MSP430F6767A	0x1D	0x82	读取/写入	读取
MSP430F6768	0x90	0x81	读取/写入	读取
MSP430F67681	0x9F	0x81	读取/写入	读取
MSP430F67681A	0x2D	0x82	读取/写入	读取
MSP430F6768A	0x1E	0x82	读取/写入	读取
MSP430F6769	0x91	0x81	读取/写入	读取
MSP430F67691	0xA0	0x81	读取/写入	读取
MSP430F67691A	0x2E	0x82	读取/写入	读取
MSP430F6769A	0x1F	0x82	读取/写入	读取
MSP430F6775	0x92	0x81	读取/写入	读取
MSP430F67751	0xA1	0x81	读取/写入	读取
MSP430F67751A	0x2F	0x82	读取/写入	读取
MSP430F6775A	0x20	0x82	读取/写入	读取
MSP430F6776	0x93	0x81	读取/写入	读取
MSP430F67761	0xA2	0x81	读取/写入	读取
MSP430F67761A	0x30	0x82	读取/写入	读取
MSP430F6776A	0x21	0x82	读取/写入	读取
MSP430F6777	0x94	0x81	读取/写入	读取
MSP430F67771	0xA3	0x81	读取/写入	读取
MSP430F67771A	0x31	0x82	读取/写入	读取
MSP430F6777A	0x22	0x82	读取/写入	读取
MSP430F6778	0x95	0x81	读取/写入	读取
MSP430F67781	0xA4	0x81	读取/写入	读取
MSP430F67781A	0x32	0x82	读取/写入	读取
MSP430F6778A	0x23	0x82	读取/写入	读取
MSP430F6779	0x96	0x81	读取/写入	读取
MSP430F67791	0xA5	0x81	读取/写入	读取
MSP430F67791A	0x33	0x82	读取/写入	读取
MSP430F6779A	0x24	0x82	读取/写入	读取
MSP430FG6425	0x37	0x82	读取/写入	读取
MSP430FG6426	0x36	0x82	读取/写入	读取
MSP430FG6625	0x35	0x82	读取/写入	读取
MSP430FG6626	0x34	0x82	读取/写入	读取
MSP430SL5438A	0xEE	0x81	读取/写入	读取

Copyright © 2021 Texas Instruments Incorporated

表 2-16. MSP430FRxx JTAG 特性

器件	0x1A04 上的器件 ID	0x1A05 上的器件 ID	对 SRAM 进行 Data Quick 访问[3]	对 FRAM 进行 Data Quick 访问	JTAG ID
MSP430FR2000	0x20	0x83	读取/写入	读取/写入	0x98
MSP430FR2032	0x78	0x82	读取/写入	读取/写入	0x98
MSP430FR2033	0x75	0x82	读取/写入	读取/写入	0x98
MSP430FR2100	0x21	0x83	读取/写入	读取/写入	0x98
MSP430FR2110	0xFB	0x82	读取/写入	读取/写入	0x98
MSP430FR2111	0xFA	0x82	读取/写入	读取/写入	0x98
MSP430FR2153	0x1D	0x83	读取/写入	读取/写入	0x98
MSP430FR2155	0x1E	0x83	读取/写入	读取/写入	0x98
MSP430FR2310	0xF1	0x82	读取/写入	读取/写入	0x98
MSP430FR2311	0xF0	0x82	读取/写入	读取/写入	0x98
MSP430FR2353	0x0D	0x83	读取/写入	读取/写入	0x98
MSP430FR2355	0x0C	0x83	读取/写入	读取/写入	0x98
MSP430FR2422	0x11	0x83	读取/写入	读取/写入	0x98
MSP430FR2433	0x40	0x82	读取/写入	读取/写入	0x98
MSP430FR2475	0x2B	0x83	读取/写入	读取/写入	0x98
MSP430FR2476	0x2A	0x83	读取/写入	读取/写入	0x98
MSP430FR2512	0x1C	0x83	读取/写入	读取/写入	0x98
MSP430FR2522	0x10	0x83	读取/写入	读取/写入	0x98
MSP430FR2532	0x3F	0x82	读取/写入	读取/写入	0x98
MSP430FR2533	0x3D	0x82	读取/写入	读取/写入	0x98
MSP430FR2632	0x3E	0x82	读取/写入	读取/写入	0x98
MSP430FR2633	0x3C	0x82	读取/写入	读取/写入	0x98
MSP430FR2675	0x29	0x83	读取/写入	读取/写入	0x98
MSP430FR2676	0x28	0x83	读取/写入	读取/写入	0x98
MSP430FR4131	0xF2	0x81	读取/写入	读取/写入	0x98
MSP430FR4132	0xF1	0x81	读取/写入	读取/写入	0x98
MSP430FR4133	0xF0	0x81	读取/写入	读取/写入	0x98
MSP430FR5041	0x0F	0x83	读取/写入	读取/写入	0x99
MSP430FR5043	0x17	0x83	读取/写入	读取/写入	0x99
MSP430FR50431	0x18	0x83	读取/写入	读取/写入	0x99
MSP430FR5720	0x70	0x81	读取/写入	读取/写入	0x91
MSP430FR5721	0x77	0x80	读取/写入	读取/写入	0x91
MSP430FR5722	0x71	0x81	读取/写入	读取/写入	0x91
MSP430FR5723	0x72	0x81	读取/写入	读取/写入	0x91
MSP430FR5724	0x73	0x81	读取/写入	读取/写入	0x91
MSP430FR5725	0x78	0x80	读取/写入	读取/写入	0x91
MSP430FR5726	0x74	0x81	读取/写入	读取/写入	0x91
MSP430FR5727	0x79	0x80	读取/写入	读取/写入	0x91
MSP430FR5728	0x7A	0x80	读取/写入	读取/写入	0x91
MSP430FR5729	0x7B	0x80	读取/写入	读取/写入	0x91
MSP430FR5730	0x7C	0x80	读取/写入	读取/写入	0x91
MSP430FR5731	0x7E	0x80	读取/写入	读取/写入	0x91
MSP430FR5732	0x75	0x81	读取/写入	读取/写入	0x91

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

表 2-16. MSP430FRxx JTAG 特性 (continued)

器件	0x1A04 上的器件 ID	0x1A05 上的器件 ID	对 SRAM 进行 Data Quick 访问[3]	对 FRAM 进行 Data Quick 访问	JTAG ID
MSP430FR5733	0x7F	0x80	读取/写入	读取/写入	0x91
MSP430FR5734	0x00	0x81	读取/写入	读取/写入	0x91
MSP430FR5735	0x76	0x81	读取/写入	读取/写入	0x91
MSP430FR5736	0x77	0x81	读取/写入	读取/写入	0x91
MSP430FR5737	0x01	0x81	读取/写入	读取/写入	0x91
MSP430FR5738	0x02	0x81	读取/写入	读取/写入	0x91
MSP430FR5739	0x03	0x81	读取/写入	读取/写入	0x91
MSP430FR5847	0x53	0x81	读取/写入	读取/写入	0x99
MSP430FR5848	0x54	0x81	读取/写入	读取/写入	0x99
MSP430FR5849	0x55	0x81	读取/写入	读取/写入	0x99
MSP430FR5857	0x57	0x81	读取/写入	读取/写入	0x99
MSP430FR5858	0x58	0x81	读取/写入	读取/写入	0x99
MSP430FR5859	0x59	0x81	读取/写入	读取/写入	0x99
MSP430FR5867	0x5B	0x81	读取/写入	读取/写入	0x99
MSP430FR5868	0x5C	0x81	读取/写入	读取/写入	0x99
MSP430FR5869	0x5D	0x81	读取/写入	读取/写入	0x99
MSP430FR5870	0x5E	0x82	读取/写入	读取/写入	0x99
MSP430FR5872(1)	0x60	0x82	读取/写入	读取/写入	0x99
MSP430FR5887	0xC1	0x81	读取/写入	读取/写入	0x99
MSP430FR5888	0xC2	0x81	读取/写入	读取/写入	0x99
MSP430FR5889	0xC3	0x81	读取/写入	读取/写入	0x99
MSP430FR5922(1) DGG 封装	0x61	0x82	读取/写入	读取/写入	0x99
MSP430FR5922(1) PM 或 RGC 封装	0x62	0x82	读取/写入	读取/写入	0x99
MSP430FR5947	0x5F	0x81	读取/写入	读取/写入	0x99
MSP430FR5948	0x60	0x81	读取/写入	读取/写入	0x99
MSP430FR5949	0x61	0x81	读取/写入	读取/写入	0x99
MSP430FR5957	0x63	0x81	读取/写入	读取/写入	0x99
MSP430FR5958	0x64	0x81	读取/写入	读取/写入	0x99
MSP430FR5959	0x65	0x81	读取/写入	读取/写入	0x99
MSP430FR5962	0xA6	0x82	读取/写入	读取/写入	0x99
MSP430FR5964	0xA4	0x82	读取/写入	读取/写入	0x99
MSP430FR5967	0x67	0x81	读取/写入	读取/写入	0x99
MSP430FR5968	0x68	0x81	读取/写入	读取/写入	0x99
MSP430FR5969	0x69	0x81	读取/写入	读取/写入	0x99
MSP430FR5970	0x5B	0x82	读取/写入	读取/写入	0x99
MSP430FR5972(1)	0x5D	0x82	读取/写入	读取/写入	0x99
MSP430FR5986	0xDF	0x81	读取/写入	读取/写入	0x99
MSP430FR5987	0xA9	0x81	读取/写入	读取/写入	0x99
MSP430FR5988	0xAA	0x81	读取/写入	读取/写入	0x99
MSP430FR5989(1)	0xAB	0x81	读取/写入	读取/写入	0x99
MSP430FR5992	0xA3	0x82	读取/写入	读取/写入	0x99
MSP430FR5994	0xA1	0x82	读取/写入	读取/写入	0x99

Copyright © 2021 Texas Instruments Incorporated

表 2-16. MSP430FRxx JTAG 特性 (continued)

器件	0x1A04 上的器件 ID	0x1A05 上的器件 ID	对 SRAM 进行 Data Quick 访问(3)	对 FRAM 进行 Data Quick 访问	JTAG ID
MSP430FR59941	0xA2	0x82	读取/写入	读取/写入	0x99
MSP430FR6035	0xED	0x82	读取/写入	读取/写入	0x99
MSP430FR60371	0xEF	0x82	读取/写入	读取/写入	0x99
MSP430FR6037	0xEC	0x82	读取/写入	读取/写入	0x99
MSP430FR6041	0x14	0x83	读取/写入	读取/写入	0x99
MSP430FR6043	0x12	0x83	读取/写入	读取/写入	0x99
MSP430FR60431	0x1A	0x83	读取/写入	读取/写入	0x99
MSP430FR6045	0xEB	0x82	读取/写入	读取/写入	0x99
MSP430FR60471	0xEE	0x82	读取/写入	读取/写入	0x99
MSP430FR6047	0xEA	0x82	读取/写入	读取/写入	0x99
MSP430FR6820 DGG 封装	0x55	0x82	读取/写入	读取/写入	0x99
MSP430FR6820 PM 或 RGC 封装	0x56	0x82	读取/写入	读取/写入	0x99
MSP430FR6822(1) DGG 封装	0x59	0x82	读取/写入	读取/写入	0x99
MSP430FR6822(1) PM 或 RGC 封装	0x5A	0x82	读取/写入	读取/写入	0x99
MSP430FR6870	0x4C	0x82	读取/写入	读取/写入	0x99
MSP430FR6872(1)	0x4E	0x82	读取/写入	读取/写入	0x99
MSP430FR6877	0xC4	0x81	读取/写入	读取/写入	0x99
MSP430FR6879	0xC6	0x81	读取/写入	读取/写入	0x99
MSP430FR6887	0xBE	0x81	读取/写入	读取/写入	0x99
MSP430FR6888	0xBF	0x81	读取/写入	读取/写入	0x99
MSP430FR6889	0xC0	0x81	读取/写入	读取/写入	0x99
MSP430FR6920 DGG 封装	0x4F	0x82	读取/写入	读取/写入	0x99
MSP430FR6920 PM 或 RGC 封装	0x50	0x82	读取/写入	读取/写入	0x99
MSP430FR6922(1) DGG 封装	0x53	0x82	读取/写入	读取/写入	0x99
MSP430FR6922(1) PM 或 RGC 封装	0x54	0x82	读取/写入	读取/写入	0x99
MSP430FR6927	0xB2	0x81	读取/写入	读取/写入	0x99
MSP430FR6928	0xB3	0x81	读取/写入	读取/写入	0x99
MSP430FR6970	0x49	0x82	读取/写入	读取/写入	0x99
MSP430FR6972(1)	0x4B	0x82	读取/写入	读取/写入	0x99
MSP430FR6977	0xAC	0x81	读取/写入	读取/写入	0x99
MSP430FR6979	0xAE	0x81	读取/写入	读取/写入	0x99
MSP430FR6987	0xA6	0x81	读取/写入	读取/写入	0x99
MSP430FR6988	0xA7	0x81	读取/写入	读取/写入	0x99
MSP430FR6989(1)	0xA8	0x81	读取/写入	读取/写入	0x99

表 2-17. MSP430ixx JTAG 特性

器件	0x0FF0 上的器件 ID	0x0FF1 上的器件 ID	对 SRAM 进行 Data Quick 访问(1)	对闪存进行 Data Quick 访问(2)
MSP430i2020	0x40	0x20	读取/写入	读取

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

表 2-17. MSP430ixx JTAG 特性 (continued)

器件	0x0FF0 上的器件 ID	0x0FF1 上的器件 ID	对 SRAM 进行 Data Quick 访问(1)	对闪存进行 Data Quick 访问(2)
MSP430i2021	0x40	0x20	读取/写入	读取
MSP430i2030	0x40	0x20	读取/写入	读取
MSP430i2031	0x40	0x20	读取/写入	读取
MSP430i2040	0x40	0x20	读取/写入	读取
MSP430i2041	0x40	0x20	读取/写入	读取

(1) 对 SRAM 进行 DataQuick 访问：如果是读取/写入，则器件支持使用 IR_DATA_QUICK 指令在快速模式下读写 SRAM（请参阅节 2.3.3.3）。某些器件可能仅支持此指令用于读取操作或仅支持用于写入操作。

(2) 对闪存进行 DataQuick 访问：如果是读取/写入，则器件支持使用 IR_DATA_QUICK 指令在快速模式下读写闪存（请参阅节 2.3.3.3）。某些器件可能仅支持此指令用于读取操作或仅支持用于写入操作。

2.7 参考文献

MSP430 器件数据表

CC430 器件数据表

《MSP430x1xx 系列用户指南》

《MSP430x4xx 系列用户指南》

《MSP430x2xx 系列用户指南》

《MSP430F5xx 和 MSP430F6xx 系列用户指南》

《MSP430FR2xx 和 MSP430FR4xx 系列用户指南》

《MSP430FR57xx 系列用户指南》

《MSP430FR58xx、MSP430FR59xx 和 MSP430FR6xx 系列用户指南》

CC430 系列产品用户指南

IEEE 标准测试访问端口和边界扫描架构，IEEE 标准 1149.1

Copyright © 2021 Texas Instruments Incorporated

3 JTAG 编程硬件和软件执行

3.1 执行历史记录

有四种 Replicator 执行方案。最新版本包含在相关源代码 ZIP 文件中，并在本文档内进行了讨论。头两个执行方案的主要区别在于使用 srec_cat.exe 函数替代 FileMaker.exe。本文档中描述的是首选执行方案，不再对之前两个执行方案进行维护。

NOTE

在一个 MSP430F5437 作为主控器时，Replicator 软件的版本只支持 REP430F 硬件。不再支持更早版本的 Replicator 硬件。

3.2 实现概述

以下几个部分将介绍随本文档提供的示例（从 http://www.ti.com/cn/lit/zip/slau320 下载示例）。每个示例使用 MSP430F5437 作为主控制器来演示之前部分描述的软件函数，这个主控制器对特别选出的指定目标 MSP430 器件进行编程。示例 zip 文件中提供了完整的 C 源代码和工程文件。还提供了这个讨论中执行的系统电路原理图。

JTAG Replicator 编程器执行的关键特性如下所示：

- 支持所有基于闪存和基于 FRAM 的 MSP430 器件。提供的特定软件项目可用于以下目标器件的 Replicator 执行：
  - Replicator430：原始 MSP430 架构器件（包括支持 Spy-Bi-Wire 的器件的 Spy-Bi-Wire 实现）
  - Replicator430X：具有扩展地址空间（20 位）的 2xx 和 4xx 系列中的 MSP430 器件，也被称为 MSP430X 器件（仅限 4 线制 JTAG）
  - Replicator430Xv2：具有闪存和扩展地址空间（20 位）的 5xx 和 6xx 系列中的 MSP430 器件，也被称为 MSP430Xv2 器件（包括 4 线制 JTAG 和 Spy-Bi-Wire 实现）
  - Replicator430FR：FRAM 器件（FR2xx、FR4xx、FR5xx 和 FR6xx 系列）（包括 4 线制 JTAG 和 Spy-Bi-Wire 实现）

  NOTE

  在之前给定的同名独立文件夹中提供了 Replicator 源文件。在这些文件夹内，当适用于特定的器件类型时，会相应地分配文件名。例如，Replicator430 版本中使用的 JTAGfunc430.c 文件在 Replicator430X 版本中被命名为 JTAGfunc430X.c，在 Replicator430Xv2 版本中被重命名为 JTAGfunc430Xv2.c。
- 最大目标器件程序代码尺寸：大约 250KB 字节（这是由于 MSP430F5437 主机控制器 256KB 的有限内存资源）
- 编程速度（擦除、编程、验证）：在 1.5s 内处理约 8KB，在 8s 内处理约 48KB
- 快速验证和擦除检查：在 10ms 内处理 17KB
- 支持对 JTAG 访问保险丝的编程（永久禁用通过 JTAG 进行的器件存储器访问）
- 独立目标编程操作（无需个人计算机或者其他支持硬件或软件）

3.3 软件操作

主控器在其闪存中存储 JTAG 通信协议代码和目标程序（MSP430F5437 上有 256KB 可用内存）。编程软件本身占用 4KB 到 6KB 空间，剩余的全部大约 250KB 空间用于目标器件程序。Replicator 主机可以通过闪存仿真工具 (FET) 或 MSP430 串行编程适配器来加载目标源代码（更多有关器件编程工具的信息，请访问 MSP430 网站）。

编程器的基本功能如下：

1. 按下 GO 按钮生成一个硬件复位并启动主机控制器的 JTAG 通信例程来擦除、编程和验证目标器件。
2. 当系统处于活动状态时，编程器板上的黄色 LED 亮起。
3. 成功完成后，只有绿色 LED 亮起。
4. 如果发生一个错误或者到目标器件的通信失败，只有红色 LED 仍然点亮。

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

对于大小为 8KB 的目标程序，整个过程需要大约 3 秒。（在 Replicator.c 源文件中会执行一些并非严格要求用于擦除、编程和验证目标 MSP430 MCU 的代码，从而增加了额定编程时间。这些额外指令可被定制以适合单独系统的编程需要。）

若要对主机 MSP430F5437 进行编程，可以使用不同的开发环境：IAR Embedded Workbench® IDE (IAR) 或德州仪器 (TI) 的 Code Composer Studio™ IDE (CCS)。IAR 的免费版强加了对代码大小的限制。为了使用前面提到的 250KB，还是需要完整版的 IAR。文件夹结构提供 IAR 和 CCS 两种文件夹，每个文件夹包含开发环境专用文件。对于 IAR，为了打开 IAR Workbench，必须启动工作区文件（文件扩展名为 .eww）。使用 CCS 时，每个 Replicator 项目必须被导入至用户的工作区。这可以通过右键点击项目视图并选择环境菜单中的 "Import" 来完成。选择所需的 Replicator 文件夹后，项目被导入并可使用。

3.4 软件结构

3.4.1 编程器固件

编程软件被分成三个级别并且除目标程序之外还包含九个文件（请参阅表 3-1）。

表 3-1. 编程器固件

最高级别	指定将执行哪个编程函数（擦除、编程、验证、熔断保险丝）。	
	Replicator430.c Replicator430X.c Replicator430Xv2.c Replicator430FR.c	包含主部分，此部分可被修改以满足定制需要。在这个程序的主部分，目标器件被擦除、检查以实现成功擦除，并被编程。编程过程将提供的示例代码载入到目标器件的存储器空间。（提供的代码文件只是快速出现在端口引脚 P1.0 和/或者 P5.1 上，这个操作驱动与 FET 工具一同提供的插槽板上的 LED，此工具可从德州仪器 (TI) MSP430 组中获得。这是已编译的 FETXXX_1.s43 示例代码文件。）此文件必须由所需的用户程序替换，并添加到要编译并加载到主机的工程中。为了演示 MSP430 JTAG 接口的功能，还包括了额外代码，此代码操作目标器件的 I/O 端口和 RAM。为了实现成功通信，这些例程可被用于测试目标器件和 PCB。
	Config430.h Config430X.h Config430Xv2.h Config430FR.h	在最初对主机控制器固件进行编程之前，包含由用户设定的高级定义。这些所谓的“快速开始选项”在其它事项中指定由 Replicator 硬件提供的电压电平以及用于与目标器件通信的接口。
	Target_Code.h	包含目标器件程序代码的基本声明。如果应实现 C 头文件而不是使用汇编文件来对目标器件进行编程，请将 Target_Code.h 的内容替换为 srec_cat.exe 的输出，并从工程中删除 Target_Code.s43 (IAR) 或 Target_Code.asm (CCS)。Target_Code.h 文件由 srec_cat.exe 文件直接生成或者由 srec.bat 文件生成。
JTAG 函数	在这里定义了所有 MSP430 专用函数。在任何情况下都不应修改这些文件。	
	JTAGfunc.c JTAGfunc430X.c JTAGfunc430Xv2.c JTAGfunc430FR.c	包含闪存编程所需的 MSP430 专用函数。
	JTAGfunc.h JTAGfunc430X.h JTAGfunc430Xv2.h JTAGfunc430FR.h	包含用于 JTAG 通信的常数定义和函数原型。
	JSBW.c	包含仿真 Spy-Bi-Wire 接口的特殊函数以将 MSP430 器件从 LPMx.5 中唤醒（仅适用于 Replicator430Xv2 和 Replicator430FR）
低级函数	所有专门依赖于主机控制器的函数（JTAG 端口 I/O 和时序函数）位于此处。如果被执行的是一个主机控制器而不是 MSP430F5437，需要对这些文件进行修改。	
	LowLevelFunc430.c LowLevelFunc430X.c LowLevelFunc430Xv2.c	包含基本主机专用函数
	LowLevelFunc430.h LowLevelFunc430X.h LowLevelFunc430Xv2.h	包含主机专用定义和函数原型
器件	描述了与闪存编程有关的特性和 MSP430 器件间的区别。（仅适用于 Replicator430 和 Replicator430X）	
	Devices430.c Devices430x.c	区分与闪存编程有关的 MSP430 器件的函数。
	Devices430.h Devices430x.h	器件函数原型和针对闪存编程的定义。

Copyright © 2021 Texas Instruments Incorporated

表 3-1. 编程器固件 (continued)

Funclets	包含写入目标 RAM 的目标代码以加速写入器件的 NVM 和从器件的 NVM 中擦除。（这些文件只能使用 IAR Embedded Workbench 和 SRecord 来重建）	
	FlashErase.c FlashWrite.c	Funclets 用于加速闪存擦除和编程
	FramErase.c FramWrite.c	Funclets 用于加速 FRAM 擦除和编程

3.4.2 目标代码

3.4.2.1 Replicator430、Replicator430X 和 Replicator430Xv2 的目标代码下载

如前所述，目标器件的程序代码必须单独提供。有两种方法可以将提供的示例包含在被发送到主机的程序目标空间内。或者包括一个独立文件（例如，Target_Code.s43 (IAR) 或 Target_Code.asm (CCS)），或在 Target_Code.h 头文件中替代 C 阵列。这两个备选方法都必需提供二进制目标代码并且符合源代码所需要的格式。

为了以由编译器输出的 TI-txt 格式构建这些文件，可从 http://sourceforge.net/projects/srecord/ 下载一个被称为 SRecord 开源转换程序。SRecord 封装包括可执行的 srec_cat.exe。

这个可执行文件是一个命令行应用程序，此程序要求采用以下格式的参数：

```
srec_cat.exe Target_Code.txt -ti_txt -Output Target_Code.h -c_array -output_word -c_compressed
```

或

```
(IAR) srec_cat.exe Target_Code.txt -ti_txt -Output Target_Code.s43 -asm -output_word -a430
(CCS) srec_cat.exe Target_Code.txt -ti_txt -Output Target_Code.asm -asm -output_word -cl430
```

参数说明：

- srec_cat.exe：应用程序的名称
- Target_Code.txt -ti_txt：这是按照名称及其格式指定的输入文件
- -Output：此关键字用来表示以下参数将描述输出文件和格式
- Target_Code.x -[c_array,asm]：这是按照名称和格式指定的输出文件，是输入文件的转换结果。只在这个示例中允许 使用 C 头文件和汇编格式。根据您的用途选择一个格式。
- -output_word：由于源代码要求向目标器件写入字，此参数是必需的参数。否则，srec_cat.exe 将写入字节。
- -c_compressed：此语句是 c_array 输出之外的附加语句。如果被指定，输出并不用 0xFF 样式填充任何地址间隔，并且不增加文件大小。
- 下面的指令为汇编输出的附加指令。选择一个来指定您的格式。
  - -a430：写入在 Replicator 环境中可被 IAR Embedded Workbench 理解的汇编文件。
  - -cl430：写入在 Replicator 环境中可被 TI CCS 理解的汇编文件。

提供的文件 srec.bat 同时生成输出文件的全部三个类型（.h，.asm 和 .s43）。命令行格式为：srec Target_Code。除了生成实际的目标代码外，SRecord 还用于转换只能使用 IAR Embedded Workbench 重建的闪存写入/Fram 写入/擦除 funclet。在此情况下，SRecord 作为相应工程中的编译后处理进程启动，用于将 TI-txt 代码转换为源代码接受的格式。通过设置所需的编译器/连接器选项，TI-txt 格式可由 IAR 连接器输出（更多信息请见 IAR 工具指令指南或者之前描述的 funclet 项目）。这也可以在 CCS 中通过使用可执行的 hex430 命令行来完成。

NOTE

如果 TI-txt 源文件包含奇数段地址或奇数个数据字节，则可能需要填充额外的字节以生成适当的字对齐输出格式。使用具有一个 "--fill 0xFF --within <input> --range-padding 2" 过滤器的 srec_cat.exe 来解决这个问题。srec.bat 自动针对适当的字对齐来过滤输出格式。
例如，“srec_cat.exe Target_Code.txt -ti_txt --fill 0xFF --within Target_Code.txt -ti_txt --range-padding 2 -Output Target_Code.h -c_array -output_word -c_compressed”。

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

NOTE

如果使用包含目标代码的汇编源代码，请确保数组声明存储在 target_code.h 中。可在所含基本头文件中找到一个示例。

NOTE

SRecord 转换程序是开源的，具有更广泛的函数。更多信息和文档，请参阅 http://srecord.sourceforge.net/。

这个软件经测试可在版本 1.36 中正确运行，但是不一定与未来版本兼容。

NOTE

为了能够轻松地将软件移植到其他微控制器，所提供的源代码是用 ANSI C 编写的。一如既往，建议在开始新工程之前安装适用的 MSP430 开发软件的最新版本。

3.4.2.2 Replicator430FR (FRAM) 的目标代码下载

如何下载由 EW430 或 CCS 生成的 msp430.txt 文件：

若要下载与 Replicator430FR 配套的不同代码，请使用 EW430 或 CSS 生成 msp430.txt 文件。将此文件命名为 msp430Code.txt 并将其保存到 $ProjectDir$\Targetcode\SRecord 文件夹中。

通过运行 $ProjectDir$\Targetcode\SRecord 文件夹中的批处理文件“ConvertMsp430_txt.bat”来转换此 msp430Code.txt 文件。输出是包含转换后的目标代码的 C 头文件。将此 C 头文件复制到 Replicator430FR 代码文件夹中，并将其包含在 JTAGfunc430FR.c 文件中 (#include "msp430Code.h")。

调用 JTAGfunc430FR.c 文件中的函数 DownloadMsp430Code() 将转换后的程序加载到目标器件存储器中。

3.5 硬体安装

此硬件由主机控制器 MSP430F5437、可编程电压稳压器（此稳压器可为目标器件提供 V_{CC}2.1V 至 3.6V（步长 0.1V）的电压）和一个 BSL 接口连接器组成。如果需要具有熔断安全保险丝的选项，则需要使用可提供 8V 至 10V (200mA) 直流电的外部电源来确保运行，否则可以使用 4V 至 10V (200mA) 直流电。如果不需要具有熔断安全保险丝的选项，REP430F 也可由目标器件的 V_{CC} ≥ 3V 供电（通过 JTAG 引脚 2 或 4；请参阅图 3-1）。

3.5.1 主机控制器

为了获得最大编程速度，主机控制器 MSP430F5437 可在 18MHz 的最大 CPU 时钟频率下运行，此频率由 LFXT1 上提供的 PLL 和 XTAL 12 MHz 使用。主机的编程是通过标记为“主机 JTAG”的专用 JTAG 端口进行的（请参阅图 3-1）。

3.5.2 目标连接

目标 MSP430 器件通过标有“Target JTAG”的 14 引脚连接器连接至主机控制器/编程器，此连接器采用与所有可用 MSP430 仿真工具相同的标准信号分配方式，但还具有额外的两个引脚可用于 BSL 连接。可编程目标器件电源电压为 2.1V 至 3.6V，步长 0.1V，此电压在这个连接器的引脚 2 上提供，从而无需为目标系统提供额外电源。所需的 Spy-Bi-Wire 或者 4 线制 JTAG 和 GND 必须被连接。（在需要 TEST 引脚的器件上，还必须从编程器向目标 MSP430 器件提供 TEST 信号。）REP430F 内的主机控制器由 V_{CC}=3V 供电，而目标器件可由 2.1V 至 3.6V 的 V_{CC} 供电。为了避免 I/O 电平问题，REP430V 在目标器件和主机控制器之间包含电压电平转换器。电压转换器由主机控制器 V_{CC}=3V 从一侧供电，并且由目标器件 V_{CC}（在目标 JTAG 连接器的引脚 2 上提供）从另一侧供电。这样可使用正好为目标器件所需的 I/O 电平为目标器件供电。

为了能够对所有包含 JTAG 访问熔丝且基于 MSP430 闪存的器件进行编程，使用了电压转换器并由主机 MSP430 来控制 MOSFET 开关。MOSFET Q2 用一个 TEST 引脚来控制器件上的 V_{PP}；Q1 将 V_{PP} 连接至不需要 TEST 信号的器件上的 TDI。 U8 将主机控制器从目标 TEST 引脚上隔离开来，此时 V_{pp} 被连接至目标 TEST 输入端，而在 V_{pp} 被连接至目标 TDI 输入端时，U6 将主机控制器从目标 TDI 引脚上隔离开来。在对熔丝进行编程时 U7 将主机 TDI 信号连接至目标 TDO 引脚（针对的是没有 TEST 引脚的器件）。主机控制器程序涉及延迟，此延迟考虑

Copyright © 2021 Texas Instruments Incorporated

到了一个最大为 1ms 的 MOSFET 开关时间。Q1 和 Q2 应该具有一个小于 2Ω 的 $R_{导通}$电阻来大大减少对熔丝进行编程期间的压降。在对熔丝进行编程时，持续大约 2μs 的 100mA 最大电流有可能流入 TDI 引脚（或者 TEST 引脚，具体取决于目标器件）。

NOTE

专为特定 MSP430 目标器件或者不执行熔丝熔断功能的系统设计的 MSP430 闪存编程器有可能需要更少的中继或者根本就无需中继。此处所述的编程器系统是为与所有系列中任何基于闪存或基于 FRAM 的 MSP430 器件一同使用而开发的，包括支持所有存储器访问功能以及保险丝熔断能力。

NOTE

在与目标器件保持连接期间，切勿拔下 JTAG 电缆。确保在断开 JTAG 连接之前已完成调试或编程例程。

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

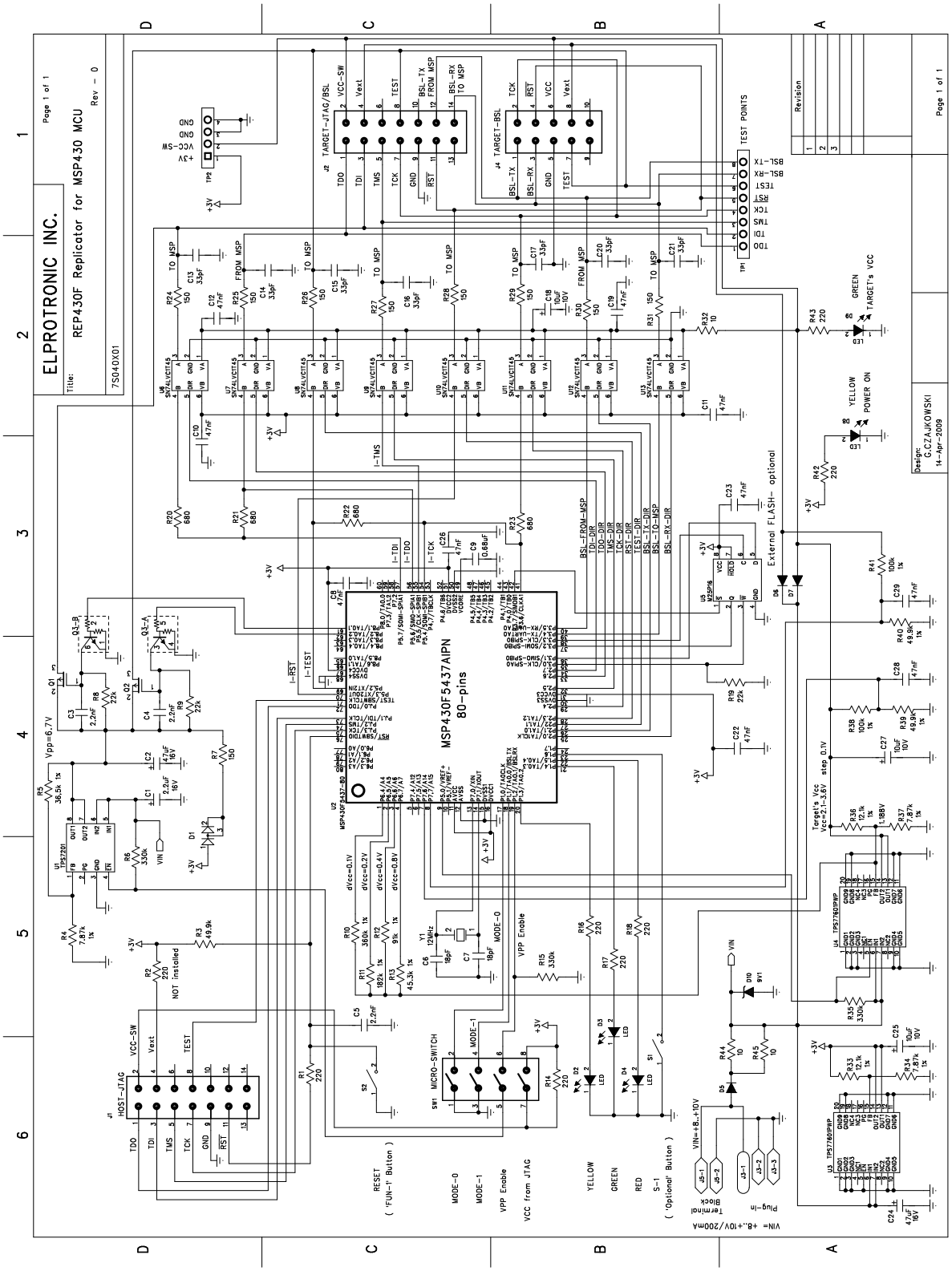

图 3-1. Replicator 应用电路原理图

3.5.3 主机控制器或者编程器电源

使用可调 LDO 从 8V 至 10V 直流输入电压生成板载电压：U3 生成 3.0V 的 V_{CC} 作为主机控制器 MSP430F5437 的电源电压，U4 生成 2.1V 至 3.6V 的 V_{CC} 作为目标器件的电源电压，而 U1 生成 6.7V 的 V_{PP} 来编程 JTAG 访问

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

保险丝。在对保险丝进行编程之时，一个 100mA 的峰值电流可流过 TEST 或者 TDI 输入引脚（请见相应的目标 MSP430 器件数据表）。

使用本地供电的目标系统时，目标器件的 V_{CC} 电平应连接至 JTAG 连接器的引脚 2，从而为 REP430F 内部的 I/O 电压转换器供电。这样就确保 REP430F 的 I/O 电平与目标器件的 I/O 电平相匹配。当目标器件由本地供电时，为目标器件供电的可编程 LDO 应该被禁用。当 REP430F 中的 JTAG 连接器的引脚 2 未连接至目标 V_{CC} 时，目标器件和 REP430F 的 I/O 电压轨之间可能会出现差异，并且主机和目标之间的通信可能会由于无效的逻辑电平而失败。在这些情况下，器件损坏有可能会发生。

3.5.4 第三方支持

Elprotronic Incorporated 提供了一个完整的 REP430F 系统，此系统与本用户指南中介绍的软件兼容。请访问 http://www.elprotronic.com 以了解更多信息。

Elprotronic Inc.
35 Austin Rumble Crt.
King City
ON, L7B 0B2
Canada
电话：+(905)539-0424
传真：+(905)539-0474
电子邮箱：info@elprotronic.com
网站：www.elprotronic.com

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

4 勘误表和修订信息

4.1 已知问题

说明

必须在慢速模式（逐字读取）下读取双端口存储器

权变措施

无

4.2 先前文档的修订和勘误表

以下是之前几个版本的*《MSP430 中的引导加载程序的应用以及闪存硬件和软件提议》*(SLAA096) 中的勘误表汇总。

- 附录 D：通用引导加载程序接口板：运算放大器 IC2 必须更换为 TL062D 或同等型号。

以下是之前几个版本的*《使用 JTAG 接口对基于闪存的 MSP430 进行编程》*(SLAA149) 中的变更汇总。

版本	日期	更改/注释
SLAA149F	2008 年 10 月	• 向第 3.4.2 节添加了时序要求。 • 删除了与第 3.7 节重复的第 4.4 节。 • 向“从 JTAG 控制中释放”部分添加了注释。
SLAA149E	2008 年 9 月	• 增加了图表 1。 • 改进了第 2.3.3 节。 • 更改了表 5 的标题。 • 增加了针对 5xx 系列的表 6。 • 更新了第 2.4.5 节的 5xx 信息。 • 修改并更新了第 3.1.2 节的 5xx 信息。 • 重命名了第 3.2 节并移动了此部分中的第 3.2.1.1 节。 • 针对 1xx、2xx、4xx 和 5xx 系列，将第 3.2 节、第 3.4 节、第 3.5 节和第 4 节分成了若干小节。 • 向表 4 添加了 IR_JMB_EXCHANGE
SLAA149D	2008 年 2 月	• 修改了第 3.5.1.1 节。在移入相应的地址值前，显示了将被载入的指令 IR_CNTRL_SIG_16BIT 而非 IR_ADDR_16BIT。 • 更新了图 10。 • 向第 2.3.2 节添加了与禁用中断有关的注释。 • 针对第 2.1 节的 JTAG 信号连接，引用了 MSP430 系列用户指南。
SLAA149C	2007 年 9 月	• 向第 A.3 节添加了与 MSP430 JTAG 限制有关的信息 • 将第 2.4.3 节中 JTAG 控制信号寄存器的位 11 的名称从 PUC 更改为 POR • 添加了第 A.1 节 • 更新了第 A.4 节有关 SRecord 转换工具的使用说明

Copyright © 2021 Texas Instruments Incorporated

5 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 NOVEMBER 23, 2019 to MAY 4, 2021	Page
• 更新了整个文档中的表格、图和交叉参考的编号格式。	3
• 向节 1.1 *关于本文档* 添加了即用型工具的链接	3
• 向节 2.1.1 添加了要点	4

English Document: SLAU320
Copyright © 2021 Texas Instruments Incorporated

重要声明和免责声明

TI 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安全、安保或其他要求。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 (https:www.ti.com/legal/termsofsale.html) 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。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2021，德州仪器 (TI) 公司

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
Copyright © 2022，德州仪器 (TI) 公司