Application Note

实现基于软件的密码保护调试

Zoey Wei

摘要

对于 MSPM0C110x 等 M0 中的低成本器件，没有硬件寄存器可用于帮助利用密码实现 SWD 保护，而只能选择完全打开或完全禁用的选项。但是，随着网络安全的普及，越来越多的应用程序（尤其是在汽车市场中）要求提供这种密码保护功能。本应用手册提供了一种基于软件实现此安全功能的新方法，适用于没有硬件来支持器件的低成本器件。

内容

商标

1 网络安全要求简介

随着网络通信、人工智能、互联网应用等技术与汽车行业的深度融合，智能互联车辆已成为汽车行业的战略方向。而随着技术的发展，智能互联车辆的信息安全问题也日益严峻，特别是近几年来，汽车信息安全召回事件层出不穷，这也引起了业界的高度关注。

2023 年 6 月，联合国统一车辆条例世界论坛（简称 UN/WP.29）发布了世界上第一个强制性汽车信息安全条例 R155，即“网络安全”。

该系列法规要求所有欧盟国家/地区和其他经合组织国家/地区的所有车型从 2024 年 7 月起都必须通过相关认证。这些法规明确规定，采取工艺措施来控制整个车辆生命周期中的相关风险，包括汽车制造商必须采取措施来监控安全威胁，并检测和防止网络攻击。

至于中国，虽然这不意味着必须满足信息安全要求，但只要汽车在上述国家/地区销售，就必须通过相关认证。而且，随着信息安全日益受到关注，对这一功能的需求也在增长。

1.1 MSPM0 的网络安全要求

在网络安全的基础上，汽车制造商对 MCU 芯片提出了以下要求：

1. 需要在上电后立即禁用 SWD，并且只有使用密码才能访问。
2. 不能使用其他通信和外部触发器，这意味着在这种情况下不允许使用 BSL。
3. 无法使用恢复出厂设置功能。

根据此要求，常用的汽车 MCU 具有集成硬件，以支持将调试访问模式更改为加密模式。MSPM0 的 G 和 L 系列具有 BOOTCFG0 寄存器以支持该功能。

表 1-1. L 和 G 系列的 BOOTCFG0 字段说明

位	字段	类型	复位	说明
31-16	SWDP_MODE	R/W	AABBh	串行线调试端口 (SW-DP) 访问策略。该策略设置是否允许通过 SWD 引脚（连接到任何 DAP）与器件进行任何通信。禁用时，无论 DEBUGACCESS 字段的配置如何，都无法进行 SWD 通信。 5566h = SW-DP 已完全禁用，无法通过 SW-DP 访问器件（0x5566 和所有其他非 0xAABB 值）。 AABBh = SW-DP 已启用，器件访问由 NONMAIN 中的附加策略设置。
15-0	DEBUGACCESS	W	AABBh	用于访问 AHB-AP、ET-AP 和 PWR-AP 调试访问端口的调试访问策略。请注意，如果 SWDP_MODE 设置为 DISABLED，则会忽略该字段的值，调试端口保持完全锁定。 5566h = 禁用通过 SWD 访问 AHB-AP、ET-AP 和 PWR-AP（0x5566 和所有其他非 0xCCDD 或 0xAABB 值）。 AABBh = 启用通过 SWD 访问 AHB-AP、ET-AP 和 PWR-AP。 CCDDh = 只有在执行 BCR 之前通过 DSSM 提供正确密码时，才能通过 SWD 访问 AHB-AP、ET-AP 和 PWR-AP。

表 1-2. C 系列的 BOOTCFG0 寄存器字段说明

位	字段	类型	复位	说明
31-16	SWDP_MODE	R/W	AABBh	串行线调试端口 (SW-DP) 访问策略。该策略设置是否允许通过 SWD 引脚（连接到任何 DAP）与器件进行任何通信。禁用时，无论 DEBUGACCESS 字段的配置如何，都无法进行 SWD 通信。 AABBh = 启用； FFFFh = 禁用（所有其他值）。
15-0	DEBUGACCESS	W	AABBh	用于访问 AHB-AP、ET-AP 和 PWR-AP 调试访问端口的调试访问策略。请注意，如果 SWDP_MODE 设置为 DISABLED，则会忽略该字段的值，调试端口保持完全锁定。 AABBh = 启用通过 SWD 访问 AHB-AP、ET-AP 和 PWR-AP； FFFFh = 禁用通过 SWD 访问 AHB-AP、ET-AP 和 PWR-AP（所有其他值）。

但是，对于 MSPM0 C 系列等低成本 MCU，此功能会因成本而被削减。本应用手册解释了如何通过软件实现加密调试，以使这种没有硬件支持的 MCU 也能满足网络安全要求。

此外，不仅限于 C 系列，L 和 G 系列也可以使用此软件方法实现更灵活的应用。

2 MSPM0 调试寄存器简介

MSPM0 使用 ARM® M0+ 内核，允许用户按照 ARM 介绍的程序将器件从 JTAG 切换到 SWD。与通用 ARM 内核相似，MSPM0 主要使用调试器根据 SWD 协议在 AP 和 DP 之间传输数据，以访问 MCU 内部。

MSPM0 器件支持调试处理器执行情况、器件状态和电源状态（通过 EnergyTrace™ 技术）。DEBUGSS 还提供了一个邮箱系统，可通过 SWD 与软件进行通信。

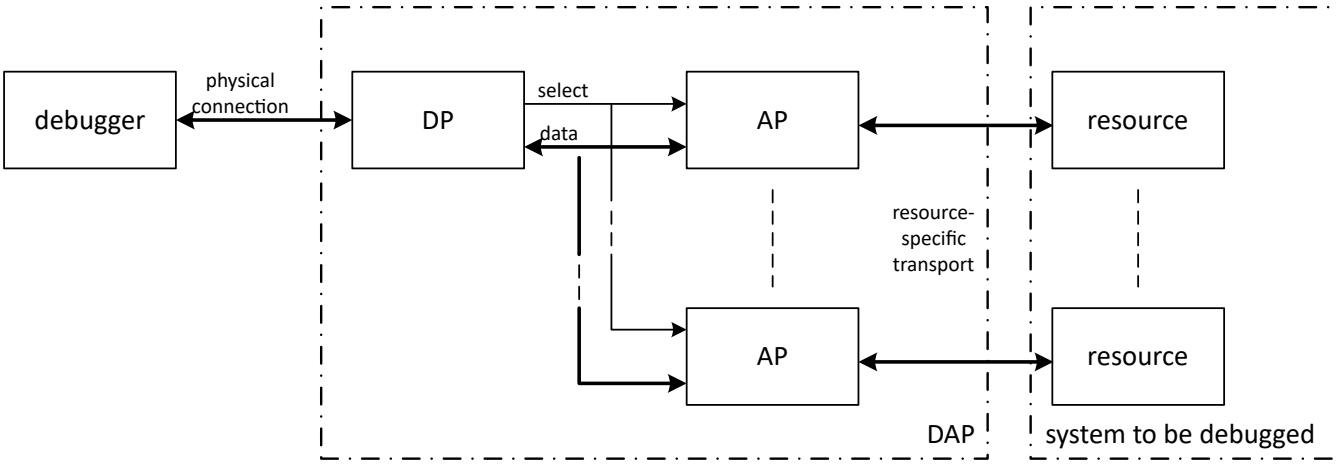

图 2-1. ARM Cortex 器件调试方框图

图 2-1 显示了 MSPM0 调试子系统方框图 MSPM0 器件支持调试处理器执行情况、器件状态和电源状态（通过 EnergyTrace 技术）。DEBUGSS 还提供了一个邮箱系统，可通过 SWD 与软件进行通信。

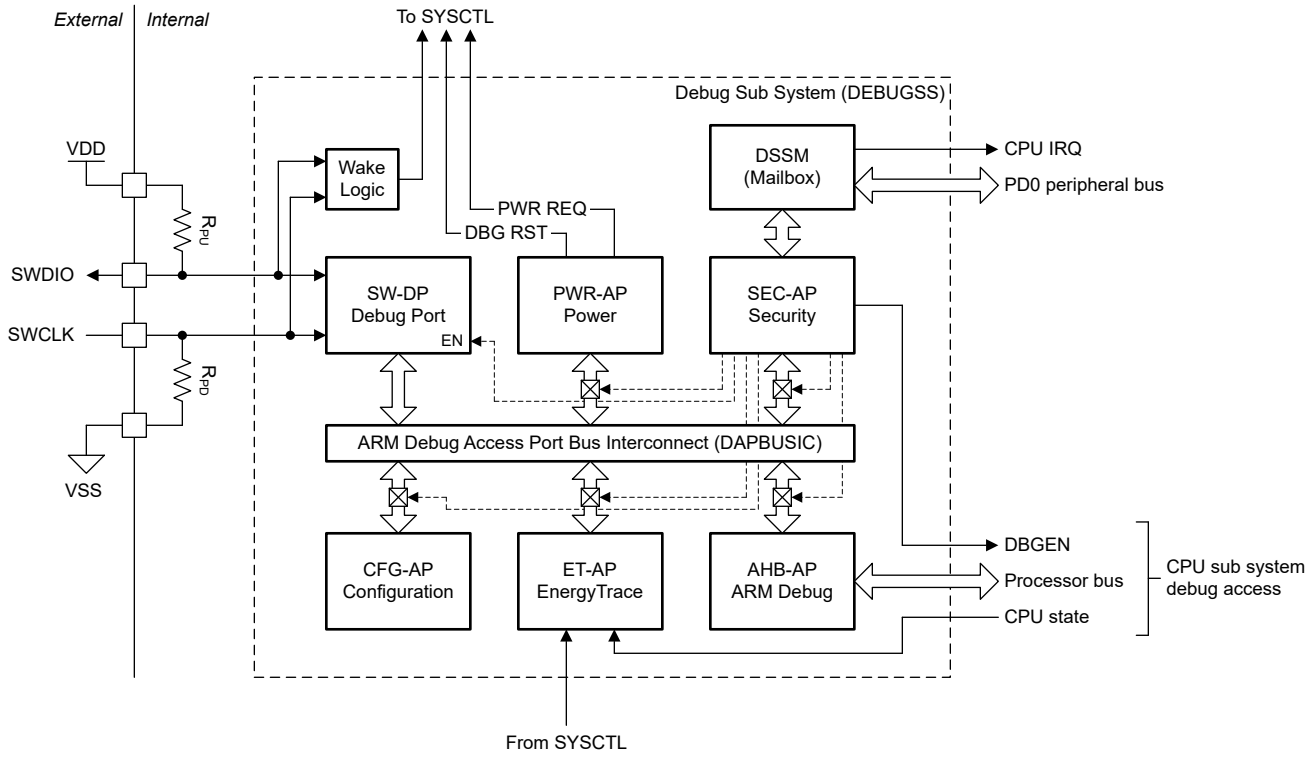

图 2-2. 调试子系统方框图

SWD 物理接口与 ARM 串行线调试端口 (SW-DP) 进行交互，以便在启用 SW-DP 时获得对调试访问端口总线互连 (DAPBUSIC) 的访问权限。

在 DEBUGSS 中有多个调试访问端口。

表 2-1. DEBUGSS 访问端口列表

AP	端口描述	用途
AHB-AP	MCPUSS 调试访问端口	处理器和外设的调试
CFG-AP	配置访问端口	访问器件类型信息，包括器件型号和器件版本。
SEC-AP	安全访问端口	在引导期间或通过 SWD 与器件上运行的软件进行通信期间访问调试子系统邮箱 (DSSM)，以便向器件传输命令。
ET-AP	EnergyTrace™ 技术访问端口	读取 EnergyTrace 技术的电源状态数据来进行功率感知调试
PWR-AP	电源访问端口	配置器件电源状态（与 PMCU/SYSCTL 连接），使能启用低功耗模式处理

AHB-AP、PWR-AP 和 ET-AP 提供完整的器件调试功能（处理器调试、外设和存储器总线访问、电源状态控制和处理器状态）。它们可通过 NONMAIN 中的 BOOTCFG0 寄存器禁用。

3 实施

该设计主要基于采用 XDS110 硬件的 SWD 协议。同时需要 MCU 和邮箱。

3.1 使用邮箱的调试器

调试子系统邮箱 (DSSM) 使调试探针能够通过 SWD 接口将消息传递到目标器件，目标器件能够将数据返回到调试探针。该设计提供了 CCS 脚本文件，以使调试器能够完成加密标识信息的传输，而 MCU SEC-AP 需要通过邮箱与应用软件进行通信。

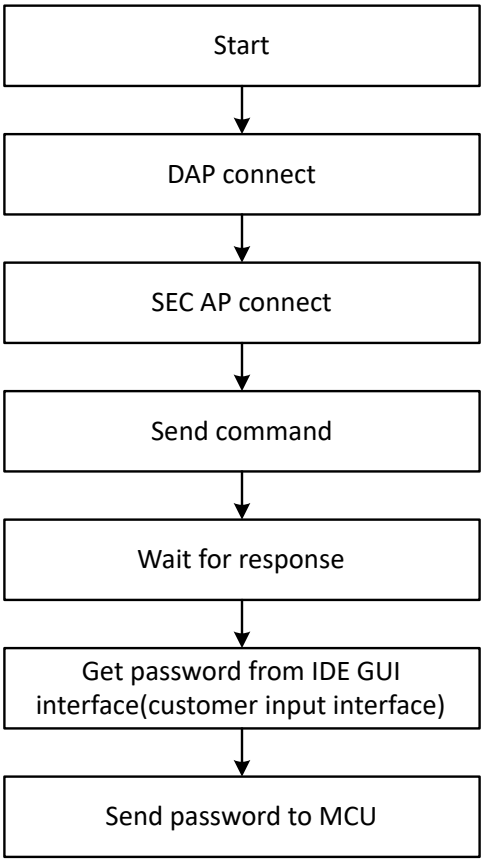

图 3-1. 调试器邮箱流程图

如图 2-1 所示，要访问内部资源，首先需要根据 SWD 协议连接 MCU 的 SW-DP 端口和 SEC-AP 端口。

然后，调试器发送密码验证命令。此命令由协议指定。当 MCU 处于密码验证阶段并且命令与 MCU 的内部引导设置一致时，MCU 将发送响应信号。收到响应后，调试器开始从外部发送密码输入，例如从 IDE GUI 接口到 MCU，尝试传递身份验证。

3.2 MCU

3.2.1 使用和配置 Nonmain

MSPM0 利用 SEC-AP 通过 DSSM 与应用软件通信。这样做的先决条件是需要保留 Arm 串行线调试端口，以确保可以接收数据。然后，可以在 DSSM 中处理数据。同时，为了防止黑客访问 MCU 应用代码，Nonmain 中禁用了 AHB-AP 端口。

对于恢复出厂设置，为防止 MCU 代码被轻松清除，请选择在 Nonmain 中禁用。

配置过程最重要的部分是密码设置。由于 MSPM0C 的 Nonmain 区域有额外的空间未分配给相应的寄存器，因此可在此存储软件密码。这样做还能让客户避免在以后更新代码时重复地将密码刻录到 MCU 中。

根据上述信息，MSPM0C 系列 Nonmain 配置如表 3-1 所示。

表 3-1. Nonmain 寄存器配置（MSPM0C 系列）

偏移	首字母缩写词	设置值	用途
41C00000h	BCRCONFIGID	0x00000003	BOOTCFG 的配置 ID
41C00004h	BOOTCFG0	0xAABBFFFF	SW-DP 已启用，但 AHB-AP、PWR-AP 和 ET AP 已禁用
41C00008h	BOOTCFG3	0xFFFFFFFF	禁用恢复出厂设置命令；禁用非主器件的静态写保护配置
41C0000Ch	SWPMAINLOW	0xFFFFFFFF	禁用较低器件闪存保护
41C00010h	SWPMAINHIGH	0xFFFFFFFF	禁用较高器件闪存保护
41C00014h	Password0	定制	无硬件寄存器。使用 Nonmain 保留空闲区域存储密码
41C00018h	Password1		
41C0001Ch	Password2		
41C00020h	Password3		

3.2.2 MSPM0 软件实现

实现此功能的代码与用户自己的应用程序位于相同的项目文件中。

根据 Nonmain 配置，MCU 可依托 SWD 与邮箱通信，但不能使用调试端口直接访问内核资源。为 MCU 编写的软件可根据 Mailbox 通信实现密码验证。

表 3-2. DSSM 寄存器功能

DSSM 寄存器	说明	调试探针	目标器件	操作
TX_DATA	数据缓冲区	RW	R	TXCTL.TRANSMIT 在调试探针进行写入时设置，并在目标器件进行读取时清零；TXIFG 也在调试探针进行写入时设置
TXCTL	流控制和状态	RW	R	无
RX_DATA	数据缓冲区	R	RW	RXCTL.RECEIVE 在目标器件进行写入时设置，并在调试探针进行读取时清零；RXIFG 也在目标器件进行写入时设置
RXCTL	流控制和状态	R	RW	无

MCU 的工作流程如图 3-2 所示。

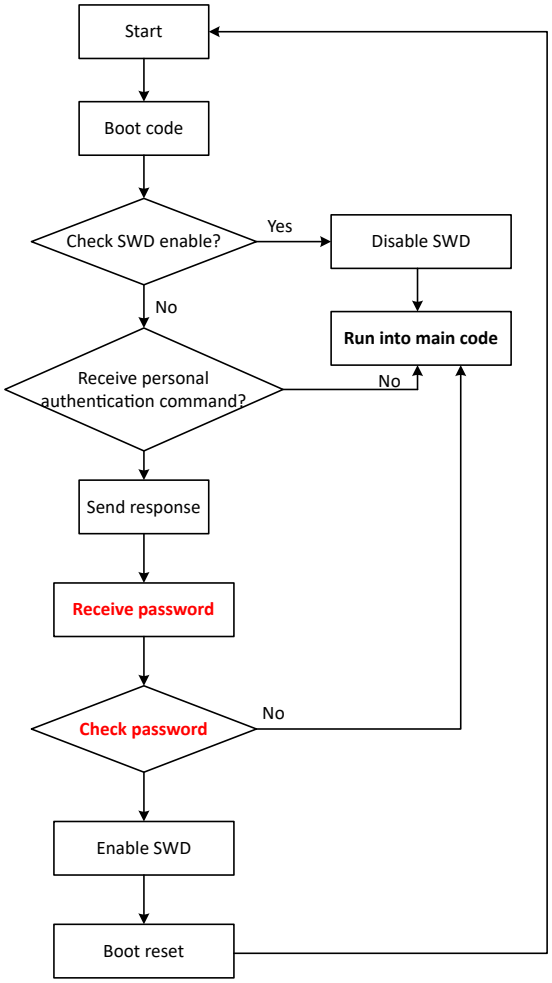

图 3-2. MCU 软件流程图

与调试器软件结合，如图 3-3 所示。下面是实现加密调试的过程。MCU 运行引导代码后，MCU 首先检查在 Nonmain 配置中是否打开了 AHB-AP。如果没有，则意味着此时处于加密状态，进一步检查 DSSM 寄存器中是否存在密码传输命令。如果有，则发送响应并使用 DSSM 寄存器接收密码。

收到四个 32 位密码后，MCU 会检查这是否为正确密码。如果正确，则会再次更改 Nonmain 配置以启用 AHB-AP 端口。

为了使 Nonmain 生效，需触发 MCU 以通过软件执行引导复位。然后，MCU 重新通电并检测到 AHB-AP 现在已开启。为了确保 MCU 在此调试完成后再次锁定，检测到 AHB-AP 端口打开后再次更改 Nonmain 配置。但是，此配置完成后 MCU 不会复位，因此修改后的 Nonmain 在调试完成之前不会立即生效。

使用密码打开 AHB-AP 端口后，MCU 运行到客户自身代码并可继续调试，再次禁用 AHB-AP 端口。

对于常规 MCU 启动，无需进行调试；在检查 SWD 是否禁用之后，MCU 会进入主应用程序代码。

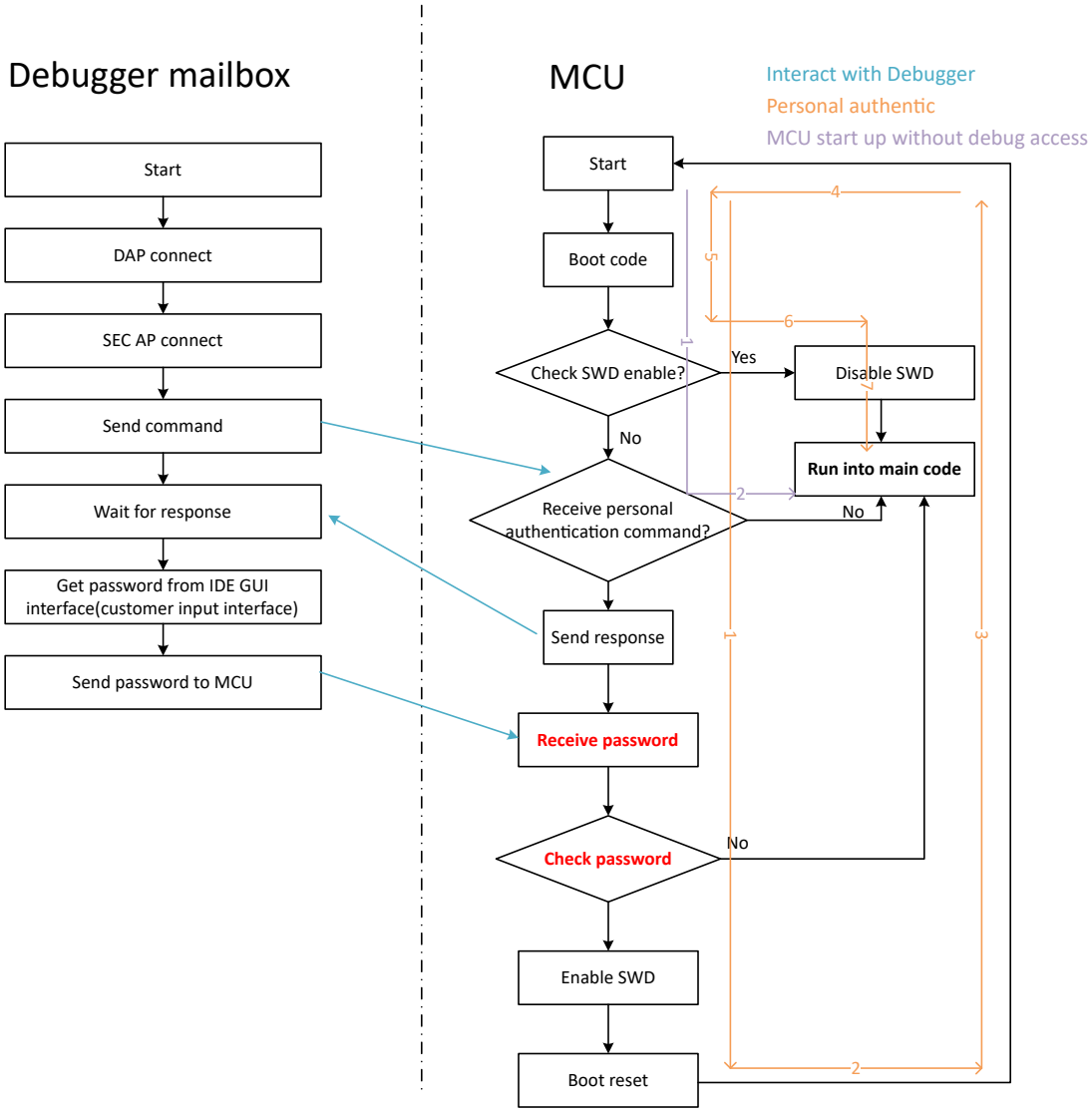

图 3-3. MCU 和调试器流程图

在整个实现过程中，会多次修改 Nonmain，这也会增加 MCU 安全风险。特别指出，NONMAIN 是闪存的专用区域，用于存储 BCR 使用的配置数据，与 SWD 策略、闪存存储器等有关。修改 Nonmain 时，如果电源故障之类的异常操作导致 Nonmain 配置不正确，则会导致 MCU 变为“砖块”，永远不会连接。为了避免此问题，在代码中添加了一个监视器，以便在开机和修改 Nonmain 后检测配置是否正确。

请注意，在 MCU 中实现加密调试的代码位于启动文件中，并在 MCU 上电后立即执行。这将跳至 C 初始化例程。

4 执行

本节提供了关于如何实现加密调试功能的分步说明。

所用材料：

- CCS IDE
- 一个代码项目
- 两个可定制的 CCS 脚本
- XDS110 硬件

客户自己的应用程序位于 main.c 中。特定应用程序连接到 节 3.2.2 中提到的 MCU 功能软件，并且位于启动文件中。此外，请注意 .cmd 文件也发生了变化。

CCS 脚本用于将调试器设置为发送相关的邮箱消息，以支持整个连接工作流程和密码发送。

备注

TI 提供了代码项目和 CCS 脚本，其中包含的演示代码可实现受密码保护的调试功能。如果用户需要进一步的帮助，请联系 TI 代表。

4.1 首次刷写

对于首次刷写，用户需要配置 NONMAIN 以存储密码。所需设置如图 4-1 所示：

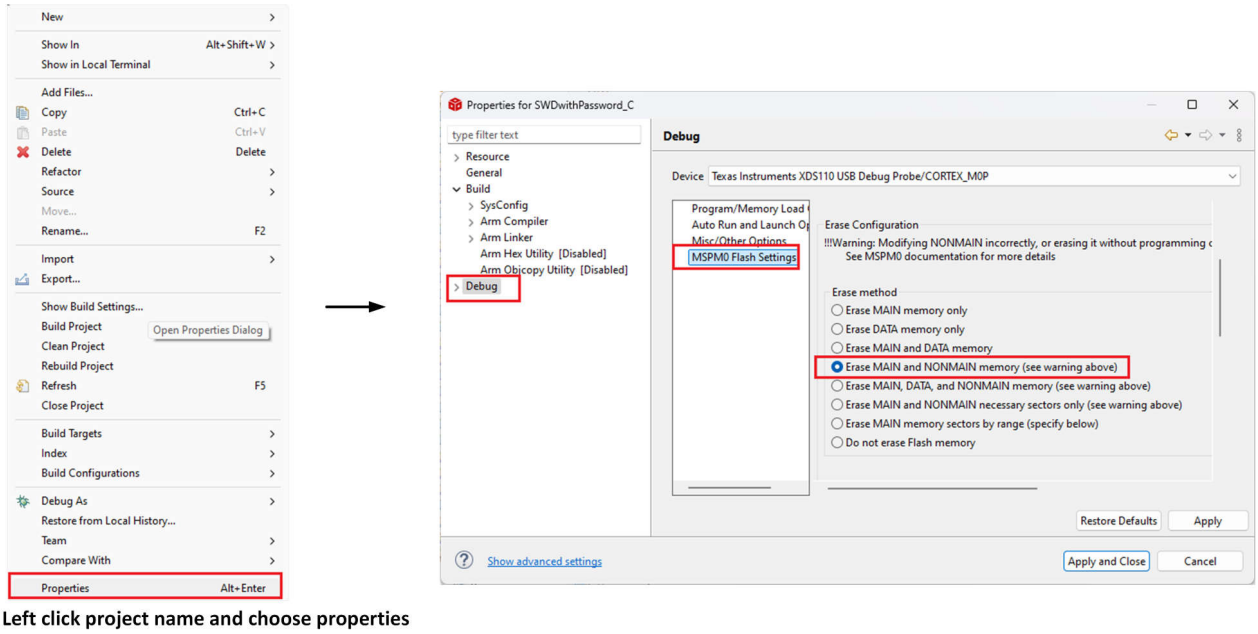

图 4-1. 启用 Nonmain 刷写

4.2 访问锁定的 MCU

如果 MCU 已被密码锁定，以下步骤展示了如何重新连接 MCU 并调试新项目。

由于在首次编程过程中已配置了 Nonmain，因此无需为后续编程配置 Nonmain。对于之后烧录或调试的项目，需要进行一些更改。首先，在项目属性中禁用 Nonmain 刷写，然后注释掉 boot_configwithPassword.c 中的 Nonmain 配置代码，如图 4-2 和图 4-3 所示。

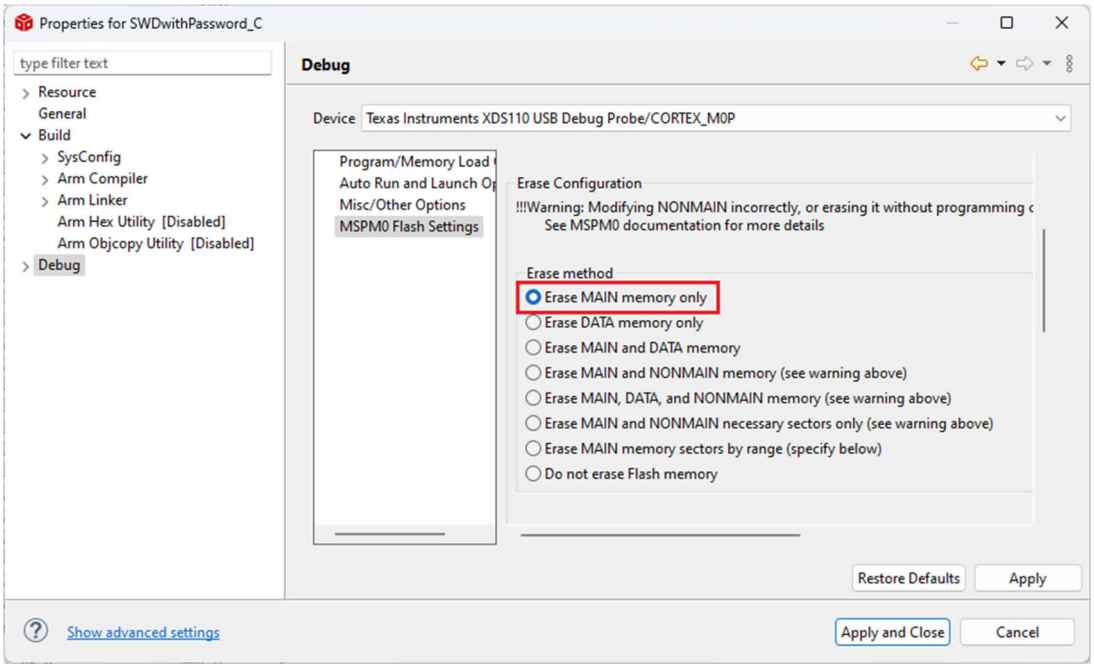

图 4-2. 禁用 Nonmain 刷写

```
/* Bootcode configuration
 * The second time need to be commented */

//PLACE_IN_MEMORY(".BCRConfig")
//const BCR_Config BCRConfig_origin =
//{
//    .bcrConfigID          = 0x3,
//    .debugAccess          = BCR_CFG_DEBUG_ACCESS_DIS,
//    .swdpMode             = BCR_CFG_SWDP_EN,
//    .factoryResetMode     = BCR_CFG_FACTORY_RESET_DIS,
//    .staticWriteProtectionNonMain  = BCR_CFG_NON_MAIN_STATIC_PROT_DIS,
//    .staticWriteProtectionMainLow  = CFG_DEFAULT_VALUE,
//    .staticWriteProtectionMainHigh = CFG_DEFAULT_VALUE,
//    .reserved = 0xFFFFFFFFU,
//    .password0 = DebugAccess_Password0,
//    .password1 = DebugAccess_Password1,
//    .password2 = DebugAccess_Password2,
//    .password3 = DebugAccess_Password3,
//};
```

图 4-3. 注释 Nonmain 配置代码

完成修改后，需要执行以下步骤来解锁和调试 MCU。

1. 在 CCS 中连接 MCU 并打开目标配置以加载 CCS 脚本。

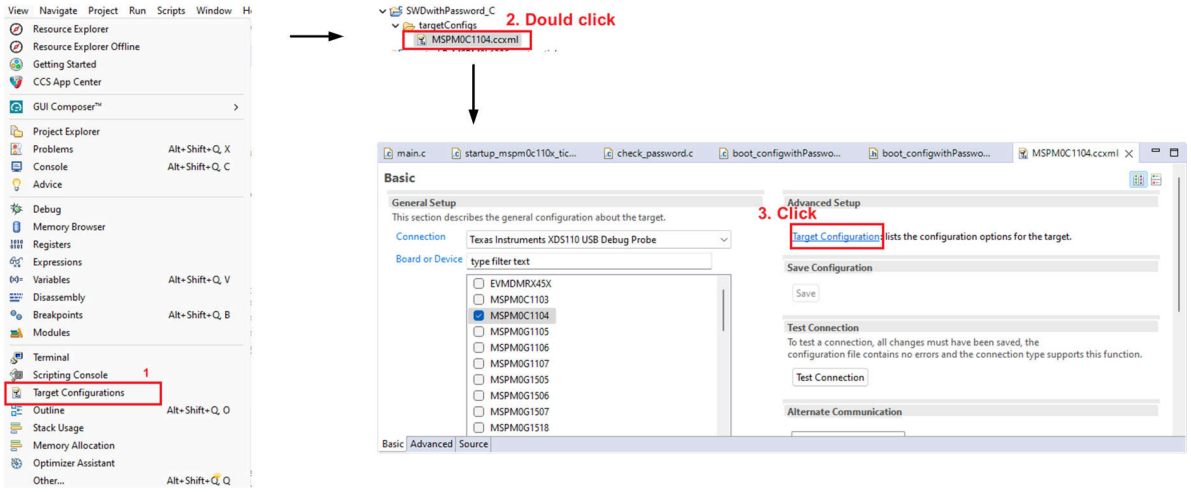

图 4-4. 打开目标配置

2. 在项目中添加两个 CCS 脚本。

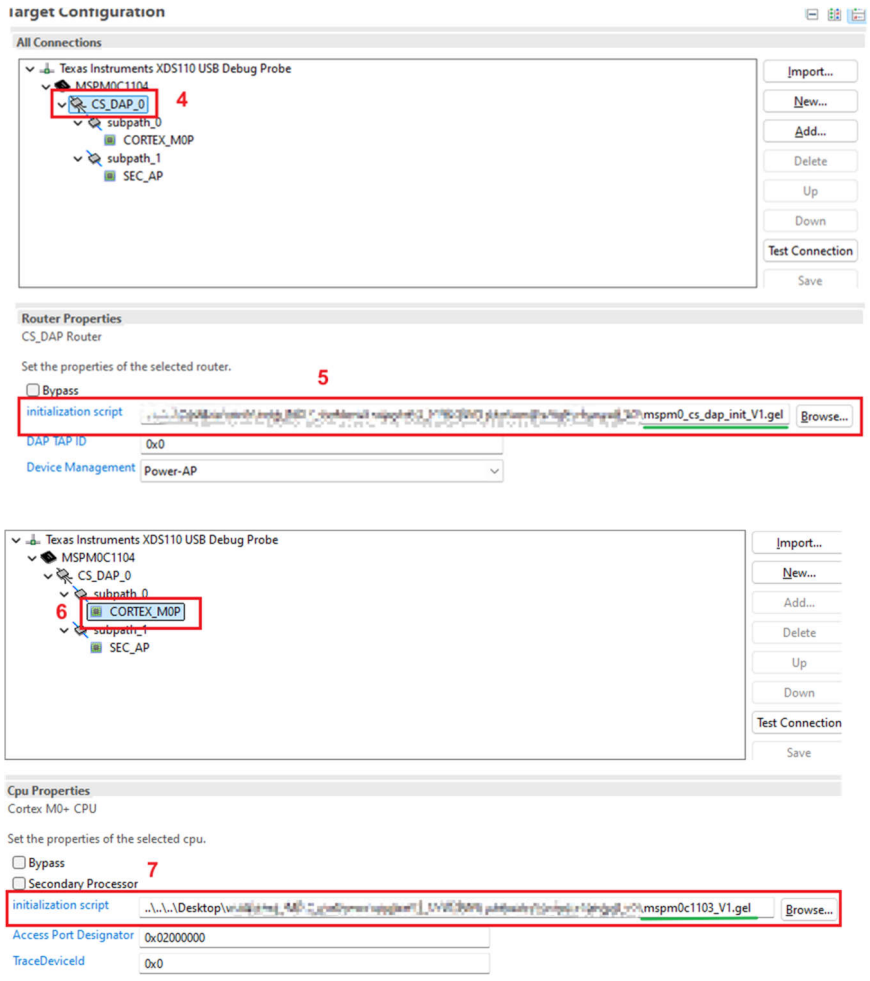

图 4-5. 添加 脚本

3. 通过 CCS GUI 界面输入密码并按 Ctrl + S 保存配置。

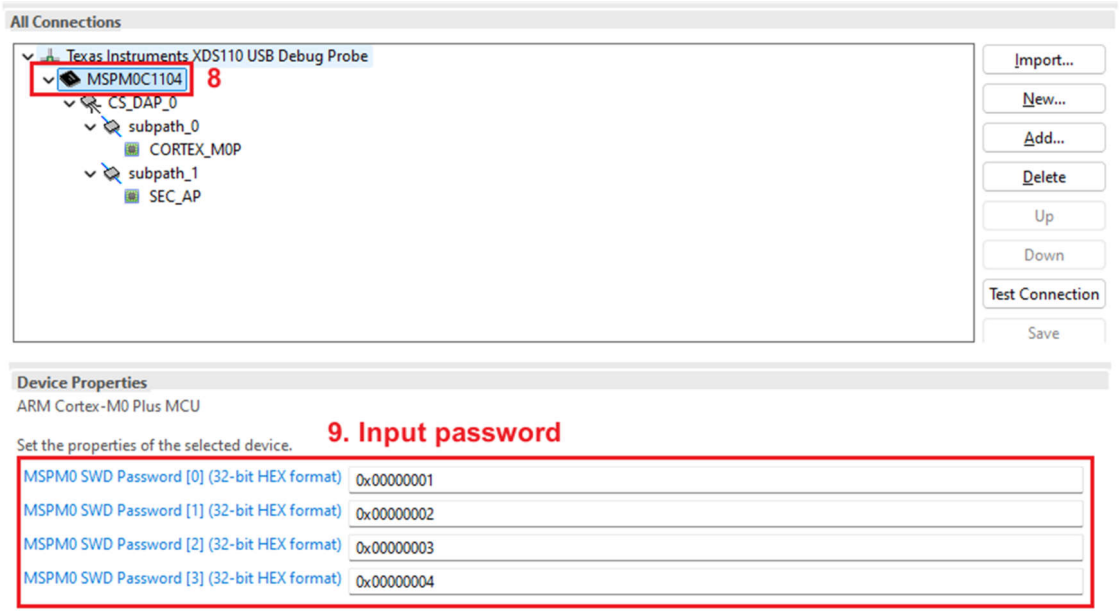

图 4-6. 通过 GUI 输入密码

4. 启动配置。

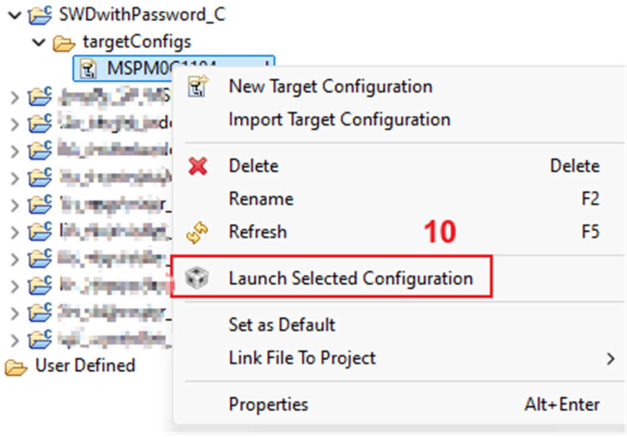

图 4-7. 启动配置

5. 使用脚本。继续下拉复位引脚，将 MCU 设置为复位模式，并且不运行代码。然后单击 MSPM0_PasswordAuthForMSPM0C。继续按住复位按钮。

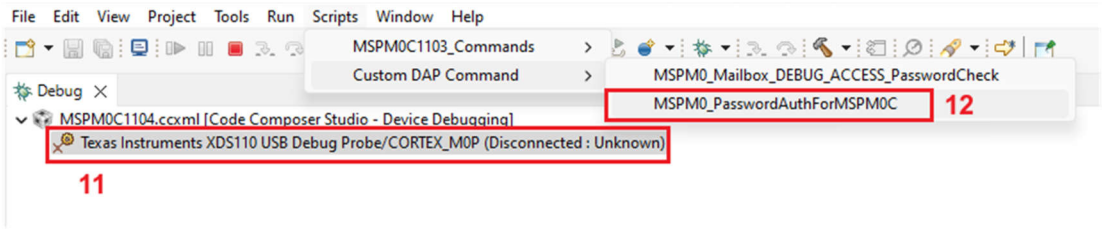

图 4-8. 运行脚本

6. 出现此界面后（显示 DAP 和 SEC-AP 已连接并等待响应），上拉复位引脚。

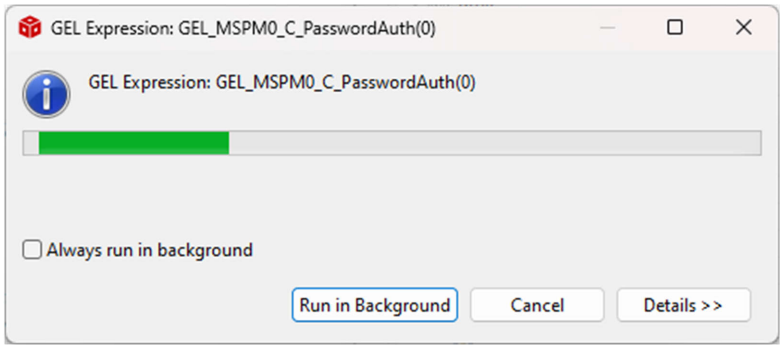

图 4-9. 显示邮箱进程的消息

7. 当复位引脚拉至高电平时，调试器尝试连接到 DAP 并开始 MCU 身份验证。如果验证通过，则会显示图 4-10 中所示的以下信息。此外，调试器发送的密码也会显示在控制台中。

```
CS_DAP_0: GEL Output: Attempting SEC_AP connection
CS_DAP_0: GEL Output: Command = 0x000000F0
CS_DAP_0: GEL Output: Respond = 0x0000CAF0
CS_DAP_0: GEL Output: Data 1 Sent = 0x00000001
CS_DAP_0: GEL Output: Data 2 Sent = 0x00000002
CS_DAP_0: GEL Output: Data 3 Sent = 0x00000003
CS_DAP_0: GEL Output: Data 4 Sent = 0x00000004
CORTEX_M0P: GEL Output: FINISHED
```

图 4-10. 控制台信息

8. MCU 已解锁并访问存储器或闪存代码。

备注

请注意，调试完成后，即执行引导复位以使 Nonmain 配置有效。MCU 再次锁定。为了便于复位，下拉复位引脚 1s，然后断电并再次上电。为对于未进行足够的密码尝试来访问 MCU 的用户，图 4-11 会显示一条错误消息。

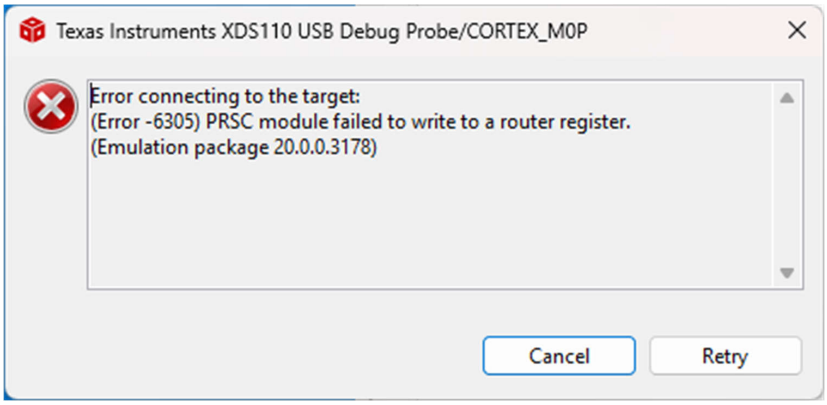

图 4-11. 连接 MCU 失败

5 如何自定义密码

如果用户想要更改邮箱通信协议，下一节将详细介绍如何更改密码。

5.1 密码

1. 演示代码 - boot_configwithPassword.h

```
#define DebugAccess_Password0                                   (0x00000001)
#define DebugAccess_Password1                                   (0x00000002)
#define DebugAccess_Password2                                   (0x00000003)
#define DebugAccess_Password3                                   (0x00000004)
```

5.2 密码长度

1. 演示代码 – boot_configwithPassword.h：密码

```
#define PASSWORD_WORD_LEN                                         (4U)
#define DebugAccess_Password0                                   (0x00000001)
#define DebugAccess_Password1                                   (0x00000002)
#define DebugAccess_Password2                                   (0x00000003)
#define DebugAccess_Password3                                   (0x00000004)
```

2. 演示代码 – boot_configwithPassword.h：BCR_Config 结构中的密码

```
/* Bootcode user configuration structure */
typedef struct
{
    /*! Configuration signature */
    uint32_t bcrConfigID;
    /*! Enable/disable AHB-AP, ET-AP, PWR-AP.
     * One of @ref BCR_CFG_DEBUG_ACCESS */
    BCR_CFG_DEBUG_ACCESS debugAccess;
    /*! Enable/disable SWD port access. One of @ref BCR_CFG_SWDP_MODE */
    BCR_CFG_SWDP_MODE swdpMode;
    /*! The factory reset mode. One of @ref BCR_CFG_FACTORY_RESET */
    BCR_CFG_FACTORY_RESET factoryResetMode;
    /*! Non Main Flash Static Write Protection.
    * One of @ref BCR_CFG_NON_MAIN_STATIC_PROT */
    BCR_CFG_NON_MAIN_STATIC_PROT staticWriteProtectionNonMain;
    /*! Programs static write protection of first 32K bytes.
     * One bit corresponds to one sector, LSB is Sector 0. Setting a bit
     * to 0 disables write, setting a bit to 1 enables write Possible values:
     *    - 0x0 to 0xFFFFFFFF */
    uint32_t staticWriteProtectionMainLow;
    /*! Programs static write protection of first 32K bytes.
     * One bit corresponds to eight sectors. Setting a bit
     * to 0 disables write, setting a bit to 1 enables write Possible values:
     *    - 0x0 to 0xFFFFFFF0 */
    uint32_t staticWriteProtectionMainHigh;
    /*! Reserved */
    uint32_t reserved;
    uint32_t password0;
    uint32_t password1;
    uint32_t password2;
    uint32_t password3;
    //Can add password length if needed
} BCR_Config;
```

3. 演示代码 – boot_configwithPassword.c：将 BCRConfig_origi 变量更改为配置 Nonmain

```
PLACE_IN_MEMORY(".BCRConfig")
const BCR_Config BCRConfig_origin =
{
    .bcrConfigID          = 0x3,
    .debugAccess          = BCR_CFG_DEBUG_ACCESS_DIS,
    .swdpMode             = BCR_CFG_SWDP_EN,
    .factoryResetMode     = BCR_CFG_FACTORY_RESET_DIS,
    .staticWriteProtectionNonMain  = BCR_CFG_NON_MAIN_STATIC_PROT_DIS,
    .staticWriteProtectionMainLow  = CFG_DEFAULT_VALUE,
    .staticWriteProtectionMainHigh = CFG_DEFAULT_VALUE,
    .reserved = 0xFFFFFFFFU,
    .password0 = DebugAccess_Password0,
    .password1 = DebugAccess_Password1,
    .password2 = DebugAccess_Password2,
    .password3 = DebugAccess_Password3,
};
```

4. 演示代码 – check_password.c：Para_init 函数

```
void Para_init(void)
{
    AHPAccess = false;
    BCRConfig_update.bcrConfigID          = 0x3;
    BCRConfig_update.debugAccess          = BCR_CFG_DEBUG_ACCESS_DIS;
    BCRConfig_update.swdpMode             = BCR_CFG_SWDP_EN;
    BCRConfig_update.factoryResetMode     = BCR_CFG_FACTORY_RESET_DIS;
    BCRConfig_update.staticWriteProtectionNonMain  = BCR_CFG_NON_MAIN_STATIC_PROT_DIS;
    BCRConfig_update.staticWriteProtectionMainLow  = CFG_DEFAULT_VALUE;
    BCRConfig_update.staticWriteProtectionMainHigh = CFG_DEFAULT_VALUE;
    BCRConfig_update.reserved = 0xFFFFFFFFU;
    BCRConfig_update.password0 = DebugAccess_Password0;
    BCRConfig_update.password1 = DebugAccess_Password1;
    BCRConfig_update.password2 = DebugAccess_Password2;
    BCRConfig_update.password3 = DebugAccess_Password3;
}
```

5. 演示代码 – check_password.c：Nonmain_check 函数
6. mspm0_cs_dap_init_V2： Password_LENGTH

```
#define PASSWORD_LENGTH                                                 (4U)
```

7. mspm0_cs_dap_init_V2：GEL_MSPM0_C_PasswordAuth(autoReset) 函数。密码长度已更改，无法将 CCS GUI 用作人机界面，因此用户需要在 CCS 脚本中添加实际密码，并发送调试器进行传输。

6 总结

本文档提供了一种实现 MCU 加密调试的方法，以便于没有硬件条件的设备仍可以通过软件满足网络安全功能。向用户提供了示例程序和脚本，这些程序和脚本可以通过文档中清晰的步骤快速实现该功能。此外，该文档还提供了相关说明，告诉用户如何定制代码。

7 参考资料

- 德州仪器 (TI)，MSPM0 C 系列器件，网页
- 德州仪器 (TI)，MSPM0 L 系列器件，网页
- 德州仪器 (TI)，MSPM0 G 系列器件，网页
- 德州仪器 (TI)，Code Composer Studio™，网页

重要通知和免责声明

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 产品的相关应用。 严禁以其他方式对这些资源进行复制或展示。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