参考資料

Application Note

フラッシュ バンク スワップを介したライブ ファームウェア アップデート

Ryan Kim *Korea Sales – Major Account*

概要

このアプリケーション ノートでは、バンク スワップと CSC (カスタマ セキュア コード) を使用したライブ ファームウェア アップデート アルゴリズムについて説明し、デュアル バンク メモリをサポートする MSPM0 デバイスに実装する方法について説明します。この例では、LP-MSPM0G3519 EVM を使用し、新しいファームウェア イメージは UART 通信を介して送信されます。

目次

商標

1 概要

デバイスがフィールドにリリースされた後、開発者は予期しないソフトウェアのバグを特定したり、新しい機能を追加したりする可能性があります。このような場合、マイコンが動作している間でもファームウェアの更新が必要です。ただし、ブートローダーまたは SWD (シリアル ワイヤ デバッグ) を使用する従来の更新方法は、通常、マイコンの動作を停止する必要があるため、このシナリオには適していません。

この制限を克服するために、ライブファームウェア アップデート メソッドを適用し、システムの動作を停止せずにファームウェアをリアルタイムで更新できます。このアプローチは、デュアル バンクのフラッシュ メモリ アーキテクチャを利用しています。これにより、一方のバンクで既存のファームウェアを実行し、もう一方のバンクに新しいイメージをプログラムできます。更新と検証のプロセスが完了した後、バンク スワップが実行され、新しいファームウェアが有効になります。

2025 年 11 月現在、デュアル バンク メモリをサポートし、バンク スワップ経由のライブ ファームウェア アップデートに利用できるマイコン製品が 17 種類あります。この実装では、新しいファームウェアはバンク 1 にダウンロードされ、既存のアプリケーションはバンク 0 で引き続き実行されます。CSC (カスタマ セキュア コード) は、ファームウェア バージョンを検証し、バンク スワップをトリガする必要があるかどうかを判断するために使用されます。新しいファームウェア イメージは、CRC32 (JAMCRC) チェックサムを含む定義済みデータ フレーム構造を使用して、UART 通信を経由して PC からマイコンに送信され、信頼性が高くエラーのないデータ転送を確保します。

ファームウェアのライブ アップデートを効率的に実行できるように、次のソフトウェア コンポーネントとツールが用意されています。

- TI が提供する
  - CSC (カスタマ セキュア コード) イメージ:バージョン チェック、およびバンク スワップ操作。
  - バンク 0 / バンク 1 アプリケーション イメージ:アクティブなファームウェアと更新されたファームウェアがそれぞれ格納されています。UART 経由で新しいファームウェアを受信します。CRC32 (JAMCRC) を使用してファームウェアを検証します
  - データ フレーム ジェネレータ (uart_frame_gui.exe):CRC32 (JAMCRC) 検証を使用して、raw .bin ファイルをフレーム化されたデータ ストリームに変換する PC 側ユーティリティ。
- ユーザー指定
  - Tera Term (https://teratermproject.github.io/index-en.html):フレーム化したファームウェア イメージを PC からマイコンに送信するために使用 (XDS-110 デバッガ経由)
  - LP-MSPM0G3519 (https://www.ti.com/tool/LP-MSPM0G3519):この評価ボードには、XDS-110 デバッガと MSPM0G3519SPZR が両方搭載されています。また、XDS-110 は USB から UART へのブリッジも搭載しており、PC から新しい FW を送信する目的で使用できます。

CSC、アプリケーション イメージ、データ フレーム ジェネレータ、スライドは、https://e2e.ti.com/cfs-file/__key/communityserver-discussions-components-files/908/Live-Firmware-Update-via-Bank-Swap_5F00_Shared-files_5F00_260116.zip のリンクからダウンロードしてください。

MSPM0 の CSC (カスタマ セキュア コード) とマルチ バンク機能の詳細については、『MSPM0 ファミリのフラッシュ マルチ バンク機能』アプリケーション ノートを参照してください。

デュアル バンク アーキテクチャをこれらのサポート ツールと組み合わせることで、開発者はデバイスの動作を中断せずに、展開されたシステムのファームウェアを安全かつ効率的に更新でき、信頼性の高い現場でのファームウェア アップデート ソリューションを実現できます。

2 詳細説明

2.1 概要

2.1.1 ライブ ファームウェア アップデート フロー

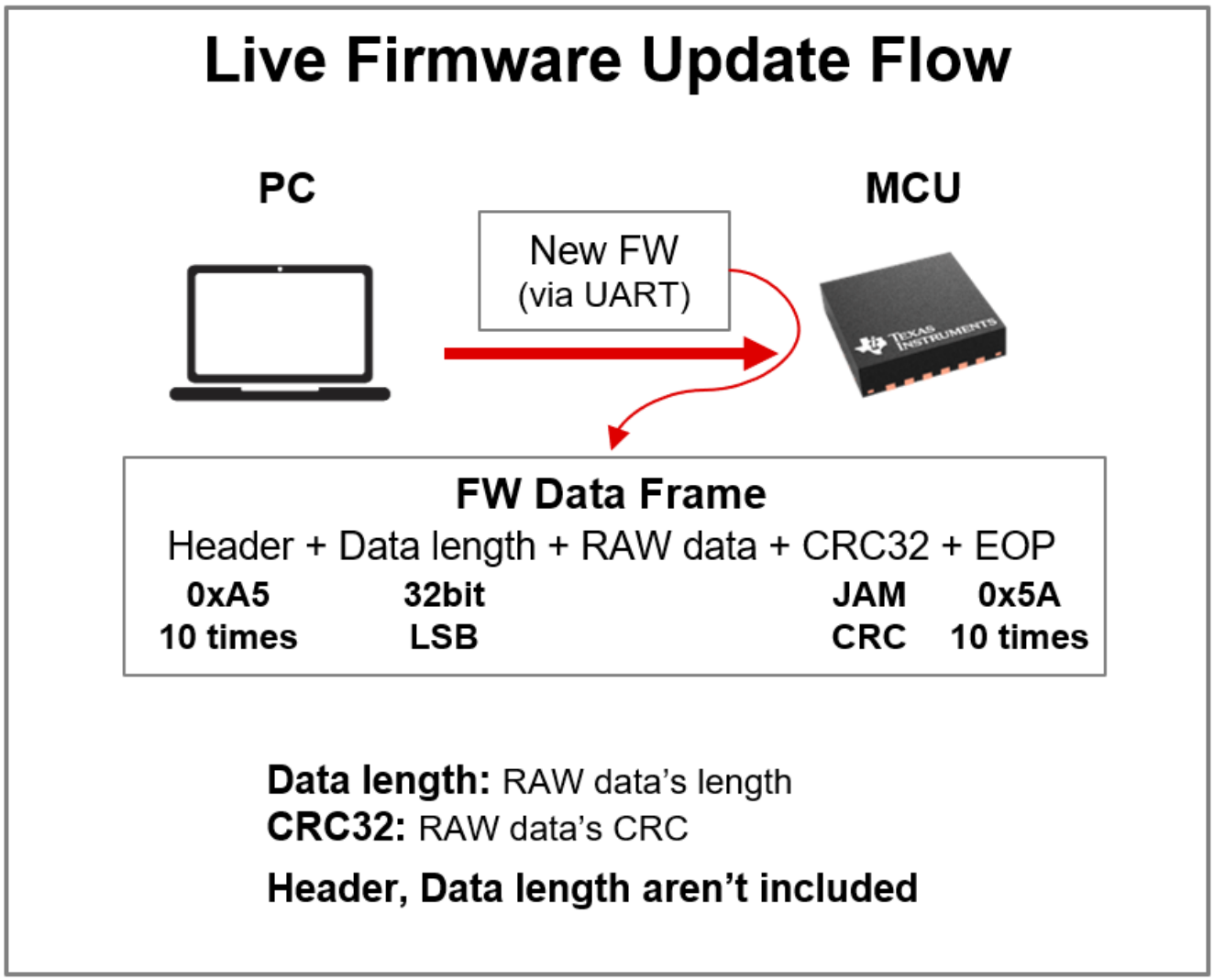

図 2-1. ライブ ファームウェア アップデート フロー

PC は、UART 経由でファームウェア データ フレームをマイコンに送信します。ファームウェア データ フレームは、ヘッダ (0xA5 × 10)、データ長 (32 ビット、LSB ファースト)、未加工データ、CRC32 (JAMCRC)、およびパケットの終了 (EOP、0x5A × 10) で構成されます。データ長フィールドは未加工データのサイズを表し、CRC32 値は未加工データに対して計算されます。

2.1.2 メモリ構成

表 2-1. メモリ構成

メモリ領域	サブ領域	アドレス (例:MSPM0G3519)	注記
バンク 0 (実行)	CSC	0x0000.0000 ～ 0x0000.1999	バンク 1 CSC と同じように、バンク スワップを決定します
	アプリケーション	0x0000.2000 ～ 0x0003.FFFF	アプリを実行しています

表 2-1. メモリ構成 (続き)

メモリ領域	サブ領域	アドレス (例:MSPM0G3519)	注記
バンク 1 (非アクティブ)	CSC	0x0004.0000 ～ 0x0004.1999	バンク 0 CSC と同じように、バンク スワップを決定します
	アプリケーション	0x0004.2000 ～ 0x0007.FFFF	古い FW または新しい FW はここでダウンロードされます

メモリ領域はバンク 0 とバンク 1 に分割され、各バンクはさらに CSC (カスタマ セキュア コード) セクションとアプリケーション (App) セクションに分割されます。

CSC は、各バンクの先頭に保存されているファームウェア バージョンを読み取り (バンク 0:0x0000.2000、バンク 1:0x0004.2000、両方 uint32_t 形式)、最新のファームウェアを含むバンクを決定し、バンク スワップが必要かどうかを決定します。

たとえば、バンク 0 にすでに最新のファームウェアが含まれている場合、CSC はバンク スワップをトリガしません。ただし、バンク 1 に新しいファームウェア バージョンが含まれている場合、CSC はバンク スワップを開始します。

チェックが完了した後、CSC はバンク 0 のアプリケーションに実行を転送します。

2.2 ブロック図

以下に、ライブ ファームウェア アップデート システムのシステム ブロック図を示します。

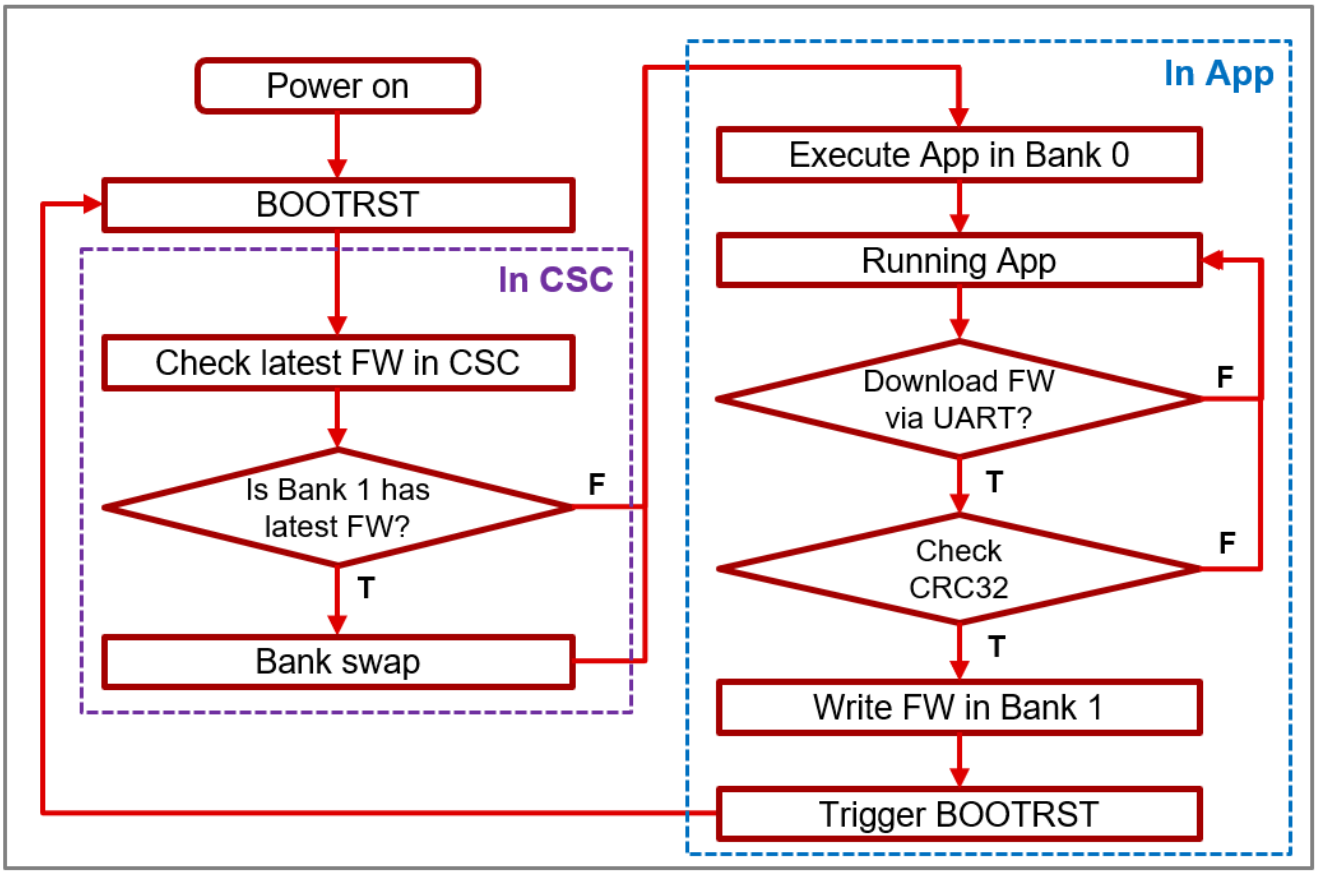

図 2-2. ライブ ファームウェア アップデートのブロック図

システムは、CSC とアプリケーションの 2 つの実行ドメインで構成されています。

CSC (カスタマ セキュア コード) は、ファームウェア バージョンを検証し、バンク スワップを実行するかどうかを決定する責任があります。

アプリケーションは、UART 経由での新しいファームウェアのダウンロード、CRC32 を使用したデータの整合性検証、アプリケーション イメージに含まれているユーザー定義関数の実行を処理します。

2.3 コード

2.3.1 CSC (カスタマ セキュア コード、Bankswap_CSC_G3519_v2)

2.3.1.1 CSC - メイン関数 (Bankswap_CSC_G3519_v2.c)

```
#define BANK0_APP_START 0x00002000
#define BANK1_APP_START 0x00042000

uint32_t gVer_info_LB0, gVer_info_LB1;
bool gBankswap_conduct;

int main(void)
{
    SYSCFG_DL_init();

    /* Get version information */
    gVer_info_LB0 = *((uint32_t*)BANK0_APP_START);
    gVer_info_LB1 = *((uint32_t*)BANK1_APP_START);

    /* Decide bank swap */
    if (gVer_info_LB0 < gVer_info_LB1) {
        uint32_t* ptr_version_info = (uint32_t*)BANK1_APP_START + 1; // 0x00042004
        uint32_t* ptr_SP = (uint32_t*)BANK1_APP_START + 64; // 0x00042256

        gBankswap_conduct = true;

        /* Check App data format */
        for(int i=0; i<63; i++) {
            if(ptr_version_info[i] != 0xFFFFFFFF) {
                gBankswap_conduct = false;
            }
        }

        /* Check stack pointer, it depends on device */
        /* MSPM0G3519 RAM 64kB - 0x20210000 */
        if(*ptr_SP != 0x20210000)  {
            gBankswap_conduct = false;
        }
    }

    if (DL_SYSCTL_isINITDONEIssued())
    {
        start_app((uint32_t *)VECTOR_ADDRESS_BANK0);
    }
    else  //Init and SWAP
    {
        if (gBankswap_conduct){
            DL_SYSCTL_executeFromUpperFlashBank(); // set flash bank swap bit
            delay_cycles(160);
            DL_SYSCTL_issueINITDONE(); // Issue INITDOEN to trigger System Reset -> swap to bank1
        }else{
            DL_SYSCTL_executeFromLowerFlashBank(); // still execute program from bank0
            delay_cycles(160);
            DL_SYSCTL_issueINITDONE(); // Issue INITDOEN to trigger System Reset -> jump to bank0
app program
        }
    }
}
```

CSC (カスタマ セキュア コード) は、バンク 0 とバンク 1 のファームウェア バージョンをチェックし、バンク スワップが必要かどうかを判定します。バージョン情報は、各バンクのアプリケーション リージョンの先頭に保存されます。具体的には、バンク 0 アプリケーション バージョンはアドレス 0x00002000 に、バンク 1 アプリケーション バージョンはアドレス 0x00042000 にあります。

バージョン情報を確認した後、バンク 1 のファームウェアが最新バージョンである場合、CSC はバンク スワップの実行を決定する前にファームウェア ヘッダを検証します。ファームウェアのヘッダが正しくない場合、またはバンク 0 に最新バージョンがすでに含まれている場合、バンク スワップは実行されません。

2.3.1.2 CSC - リンカ ファイル (Bootloader.cmd)

```
--define=_BOOT_SIZE_=(8*1024)

MEMORY
{
    FLASH_BOOT      (RX)  : origin = 0x00000000,  length = _BOOT_SIZE_
    FLASH_APP       (RX)  : origin = _BOOT_SIZE_, length = (0x00040000 - _BOOT_SIZE_)
}

SECTIONS
{
    .intvecs      : > 0x00000000
    .text         : palign(8) {} > FLASH_BOOT
    .const        : palign(8) {} > FLASH_BOOT
    .cinit        : palign(8) {} > FLASH_BOOT
    .pinit        : palign(8) {} > FLASH_BOOT
    .rodata       : palign(8) {} > FLASH_BOOT
    .ARM.exidx    : palign(8) {} > FLASH_BOOT
    .init_array   : palign(8) {} > FLASH_BOOT
    .binit        : palign(8) {} > FLASH_BOOT
}
```

CSC 境界は、8,192 バイト (0x0000 ～ 0x2000) に設定されます。Arm Cortex-M0+ VTOR (ベクタ テーブル オフセット レジスタ) の整列要件 0x100 と 1kB のフラッシュ消去セクタ サイズ (0x400) により、ユーザーは境界を 0x400 に合わせる必要があります。

2.3.2 アプリ (Bankswap_G3519_gpio_output_toggle_v2_SW_Version55_CRC32)

2.3.2.1 アプリ – メイン関数 (Bankswap_G3519_gpio_output_toggle_v2_SW_Version55_CRC32.c)

```
#define APP_VERSION 55 // 0 ~ 4294967295

/* Version information */
__attribute__((section(".version_info"), retain))
const uint32_t gVersionInfo[64] = {APP_VERSION, [1 ... 63] = 0xFFFFFFFF};

while (1) {
    /* Wait until new FW is downloaded */
    while (!gFrameReady) {
        __WFI();
    }
    gFrameReady = false;

    func_CRC32_check();

    if (gCRCChecksumMatch) {
        gCRCChecksumMatch = false;
        func_reset_sectors();
        func_program_to_bank1();
        DL_SYSCTL_resetDevice(DL_SYSCTL_RESET_BOOT);
    }
}
```

新しいファームウェアをダウンロードした後、「gFrameReady」フラグが設定され、以下のプロセスが順にトリガされます: CRC チェック (func_CRC32_check)、フラッシュ セクタのリセット (func_reset_sectors)、フラッシュ (func_program_to_bank1) への書き込み。完了すると、BOOTRST がトリガされ、CSC はバンク 0 とバンク 1 の間でバンク スワップを実行します。

2.3.2.2 アプリ – UART ISR (Bankswap_G3519_gpio_output_toggle_v2_SW_Version55_CRC32.c)

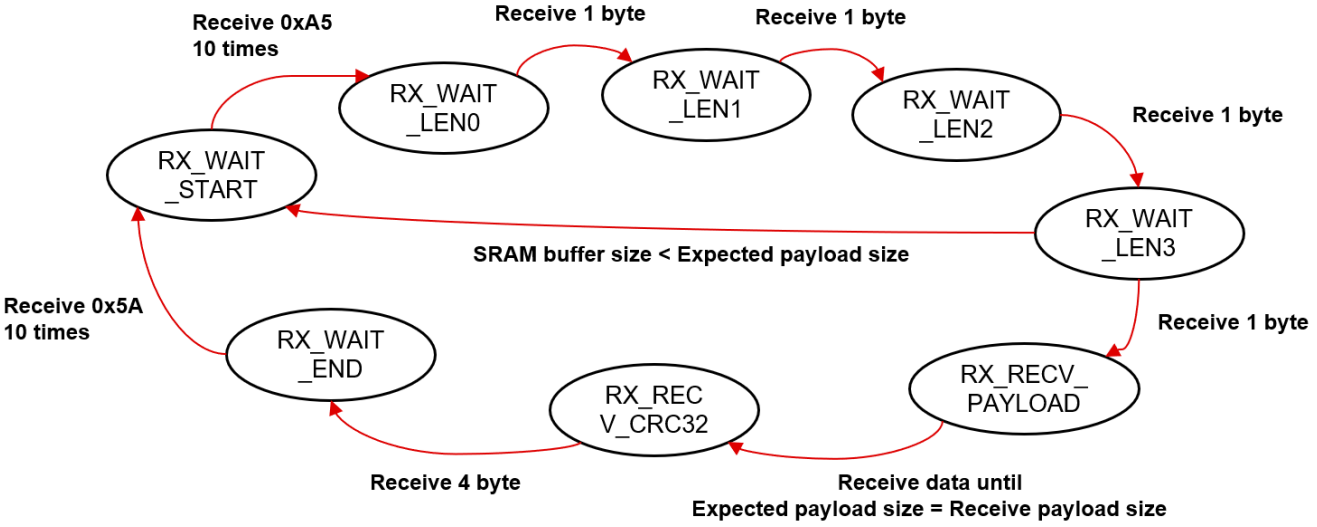

図 2-3. アプリ – UART ISR ステート マシン

UART ISR (割り込みサービス ルーチン) は、ステート マシン アーキテクチャに基づいて設計されています。このプロセスは次の 5 つの状態に分割されます。ヘッダ (RX_WAIT_START)、想定データ長 (RX_WAIT_LEN0、1、2、3)、ペイロード (RX_RECV_PAYLOAD)、CRC32 (RX_RECV_CRC32)、EOP (パケットの終了、RX_WAIT_END) です。

この実装では、バッファ オーバーフローを防ぐために、想定ペイロード サイズも検証します。

2.3.2.3 アプリ - リンカ ファイル (device_linker.cmd)

```
--define=_BOOT_SIZE_=(8*1024)
--define=_VERSION_SIZE_=(256)
--define=_TOTAL_SIZE_=(8*1024+256)

MEMORY
{
    BOOT            (RX)  : origin = 0x00000000,  length = _BOOT_SIZE_
    FLASH_VERSION   (RWX) : origin = _BOOT_SIZE_,  length = _VERSION_SIZE_
    FLASH           (RX)  : origin = _BOOT_SIZE_ + _VERSION_SIZE_, length = 0x00040000 -
_BOOT_SIZE_ - _VERSION_SIZE_
    SRAM_BANK0      (RWX) : origin = 0x20200000,  length = 0x00010000
    SRAM_BANK1      (RWX) : origin = 0x20210000,  length = 0x00010000
    BCR_CONFIG      (R)   : origin = 0x41C00000,  length = 0x000000FF
    BSL_CONFIG      (R)   : origin = 0x41C00100,  length = 0x00000080
    DATA            (R)   : origin = 0x41D00000,  length = 0x00004000
}
SECTIONS
{
    .version_info : palign(8) {} > FLASH_VERSION
    .intvecs:   > _TOTAL_SIZE_
    .text   : palign(8) {} > FLASH
    .const  : palign(8) {} > FLASH
    .cinit  : palign(8) {} > FLASH
    .pinit  : palign(8) {} > FLASH
    .rodata : palign(8) {} > FLASH
    .ARM.exidx    : palign(8) {} > FLASH
    .init_array   : palign(8) {} > FLASH
    .binit        : palign(8) {} > FLASH
}
```

アプリケーション領域は、CSC の直後にあり、アドレス 0x0000.2000 から始まります。

アプリケーション領域の先頭にあるバージョン情報 (.version_info) は uint32_t 形式で保存されます。この形式は 4 バイトしか必要としません。ただし、Arm Cortex-M0+ VTOR (ベクタ テーブル オフセット レジスタ) の整列要件 0x100 (256 バイト) のため、バージョン情報セクションには 0x100 (256 バイト) が割り当てられます。

フラッシュ メモリの容量は、MCU モデルによって異なります。したがって、ユーザーはデバイスごとに適切なメモリ境界を設定する必要があります。この例は、256kB のフラッシュ メモリを搭載した MSPM0G3519 をベースにしています。

2.4 実装

2.4.1 実装の概要

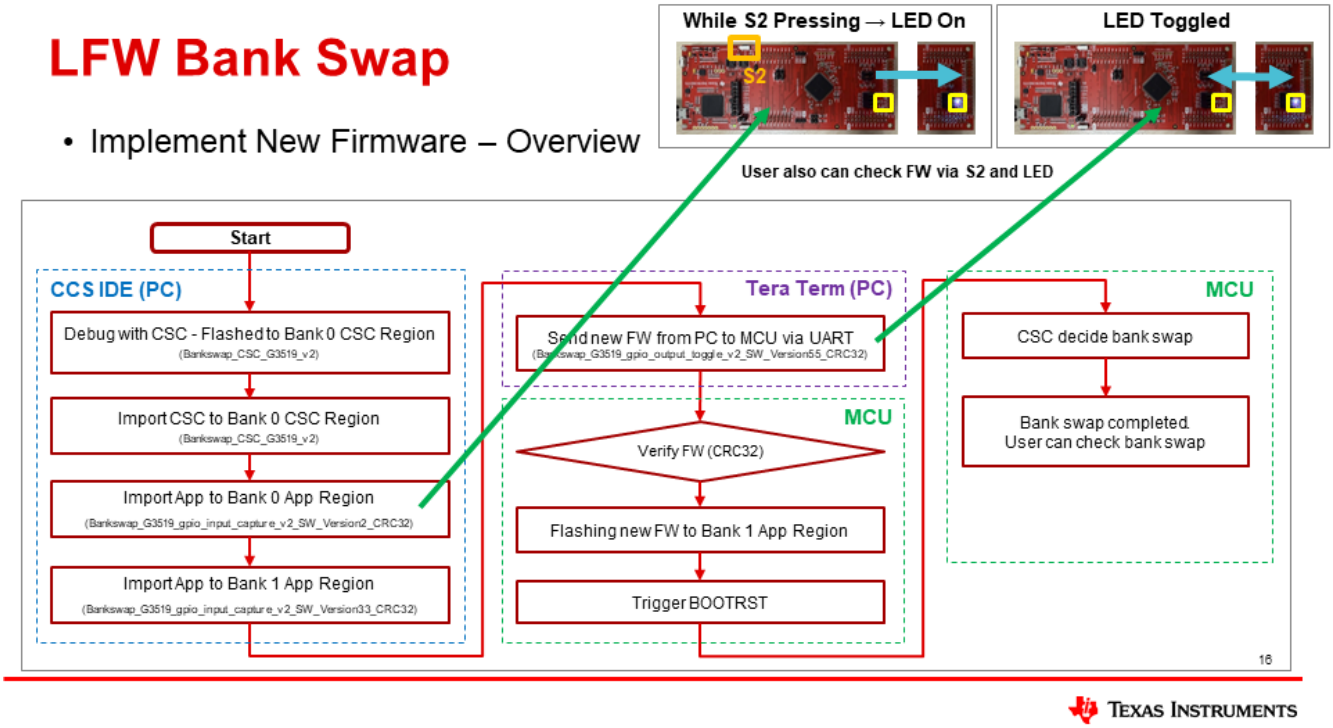

図 2-4. 実装の概要

ファームウェアの更新完了は CCS IDE で確認できますが、スイッチや LED インジケータを使用して確認することもできます。

新しいファームウェアを送信する前に、アプリケーション「Bankswap_G3519_gpio_input_capture_v2_SW_Version2_CRC32 (NRST 押下なし)」または「Bankswap_G3519_gpio_input_capture_v2_SW_Version33_CRC32 (NRST 押下)」が、バンク 0 アプリケーション リージョンで実行されます。この間、ユーザーが S2 を押すと LED が点灯します。

新しいファームウェアを送信した後、アプリケーション Bankswap_G3519_gpio_output_toggle_v2_SW_Version55_CRC32 が、バンク 0 のアプリケーション リージョンで動作します。LED が点滅を開始し、ファームウェアの更新が正常に適用されたことを示します。

2.4.2 実装プロセス

2.4.2.1 CCS プロジェクト ファイルをインポート (TI CCS IDE)

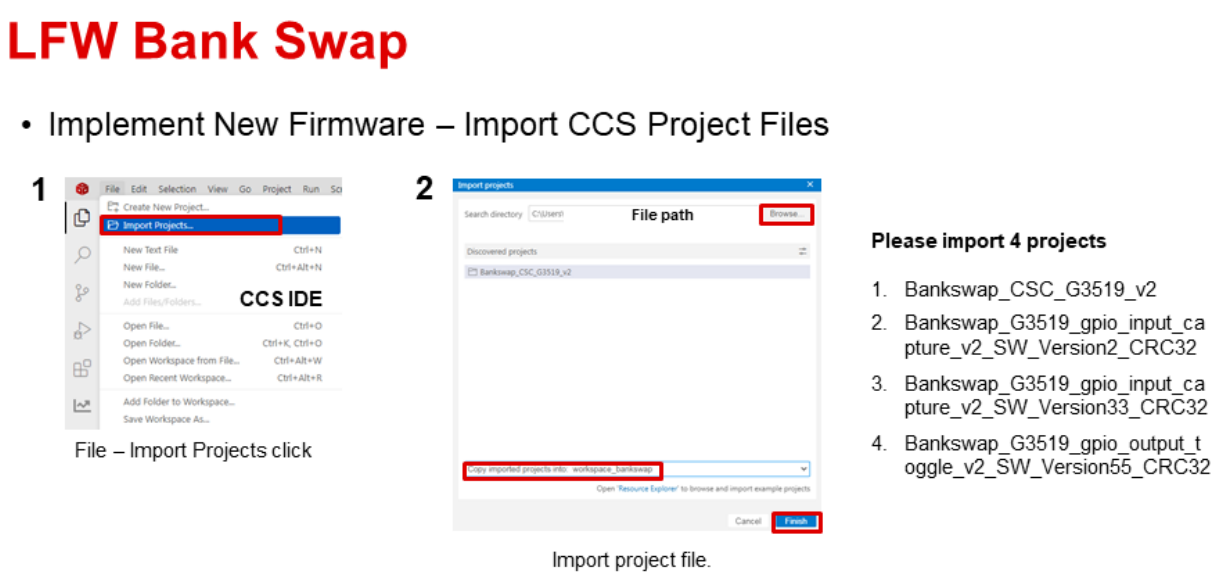

図 2-5. CCS プロジェクト ファイルのインポート

次の 4 つのプロジェクトをインポートします。

- Bankswap_CSC_G3519_v2
- Bankswap_G3519_gpio_input_capture_v2_SW_Version2_CRC32
- Bankswap_G3519_gpio_input_capture_v2_SW_Version33_CRC32
- Bankswap_G3519_gpio_output_toggle_v2_SW_Version55_CRC32

2.4.2.2 マイコンの工場出荷時リセットの実行 (TI CCS IDE)

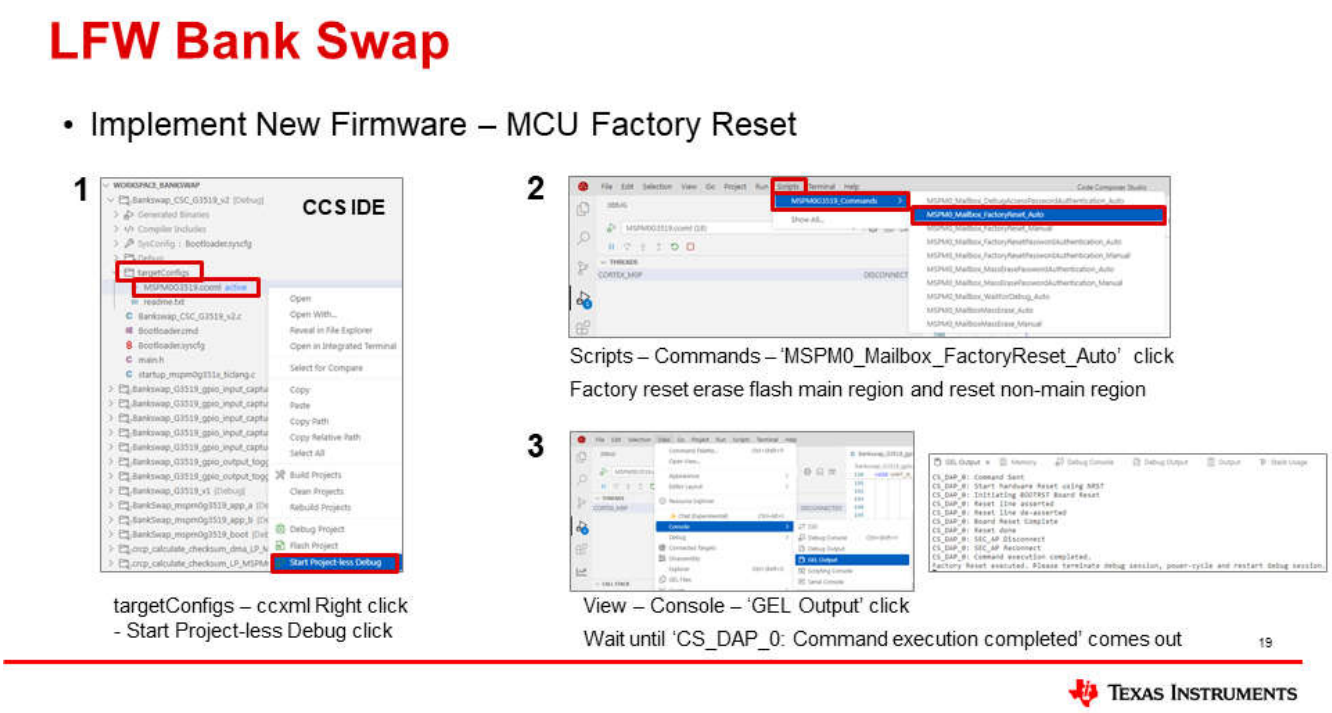

図 2-6. マイコンの工場出荷時リセットの実行し

潜在的な競合を回避するために、TI は工場出荷時リセットを実行することを強く推奨します。CCS IDE には工場出荷時リセット機能が内蔵されており、ユーザーは 4 つの「MSPM0G3519.ccxml」ファイルのいずれかを選択して、リセットを実行できます。

2.4.2.3 CCS (TI CCS IDE) で CSC アプリケーションをビルド

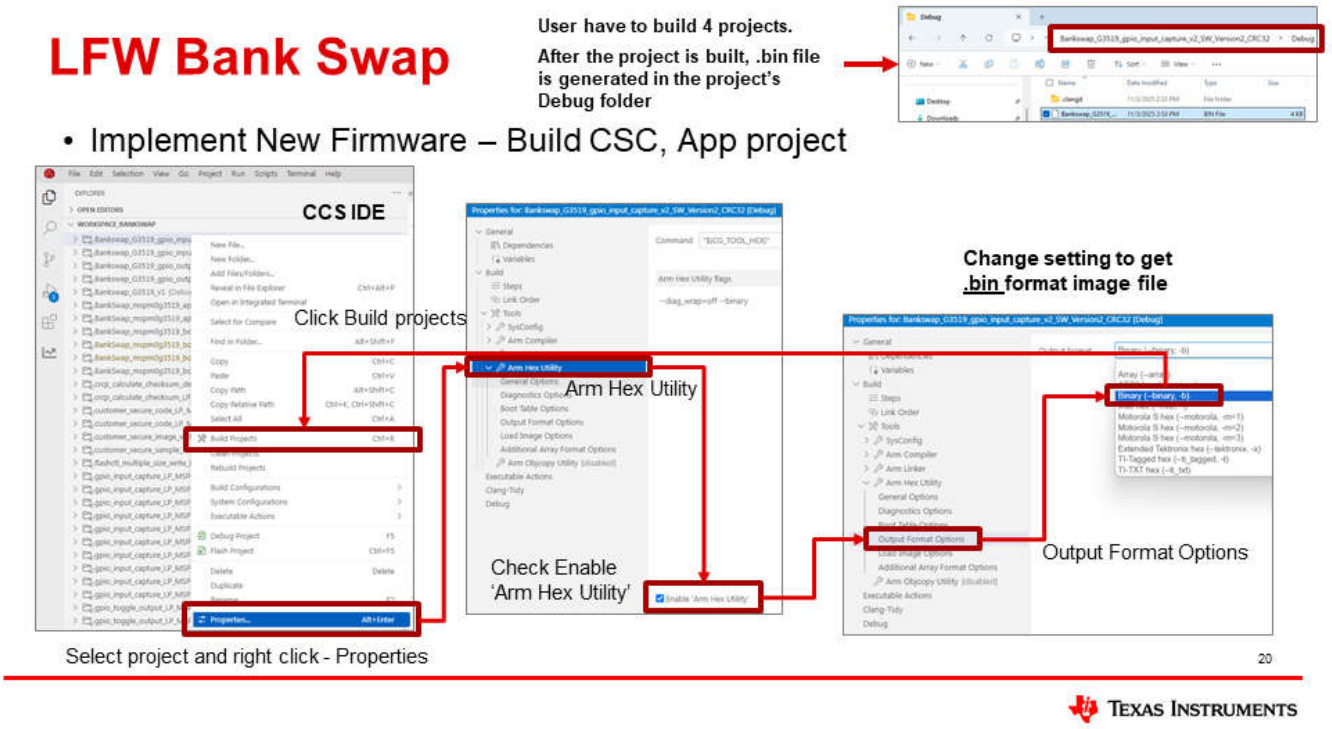

図 2-7. CCS での CSC アプリケーションのビルドし

プロジェクトをビルドする前に、Arm Hex ユーティリティ オプションを有効にします。この機能を有効にしたら、4 つのプロジェクトすべてのビルドに進みます。

2.4.2.4 CCS (TI CCS IDE) でデバッグを開始し、イメージをマイコンにダウンロード

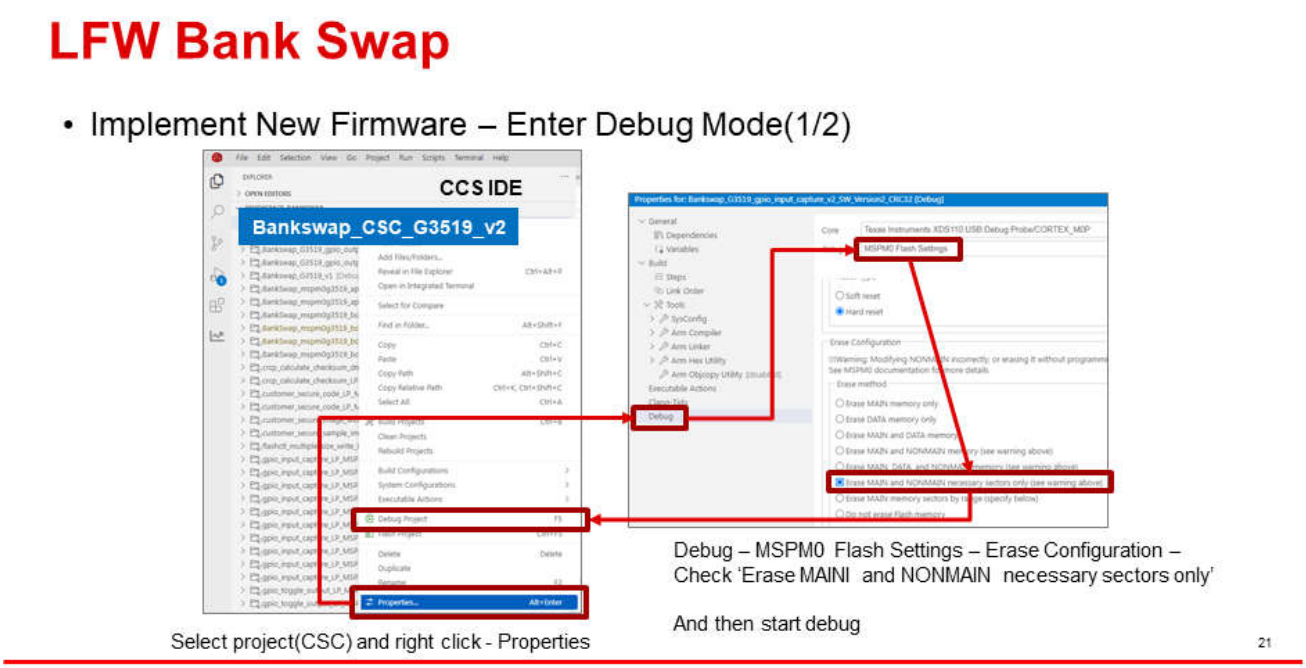

図 2-8. CCS 1 でデバッグを開始し、イメージをマイコンにダウンロード

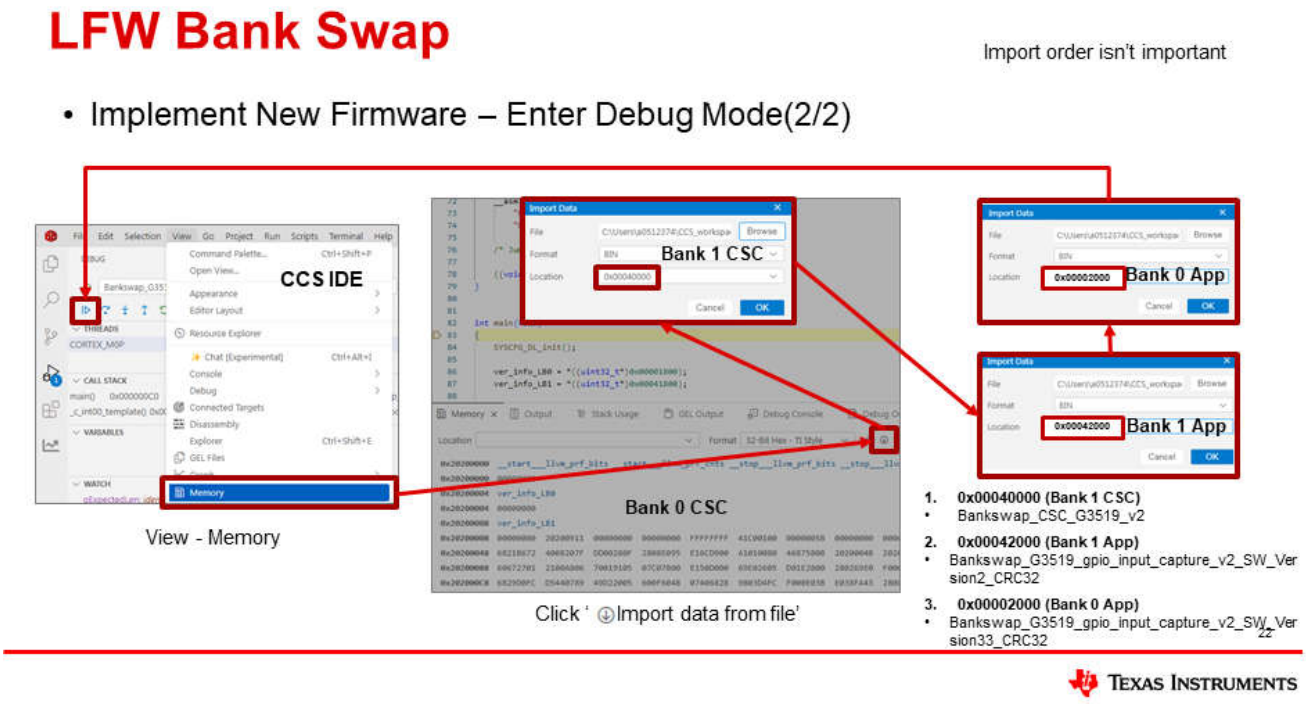

図 2-9. CCS 2 でデバッグを開始し、イメージをマイコンにダウンロード

CSC プロジェクト (Bankswap_CSC_G3519_v2) のデバッグを開始する前に、ユーザーはフラッシュ設定を調整する必要があります。「Erase MAIN and NONMAIN necessary sectors only」(MAIN および NONMAIN の必要なセクタのみを消去) が選択されているか確認してください。

デバッグ モードに移行すると、CSC プロジェクトはバンク 0 にプログラムされます。次に、CSC をバンク 1 に、アプリケーションをバンク 0 とバンク 1 の両方にプログラムする必要があります。

2.4.2.5 送信するデータ フレームを生成 (uart_frame_gui.exe)

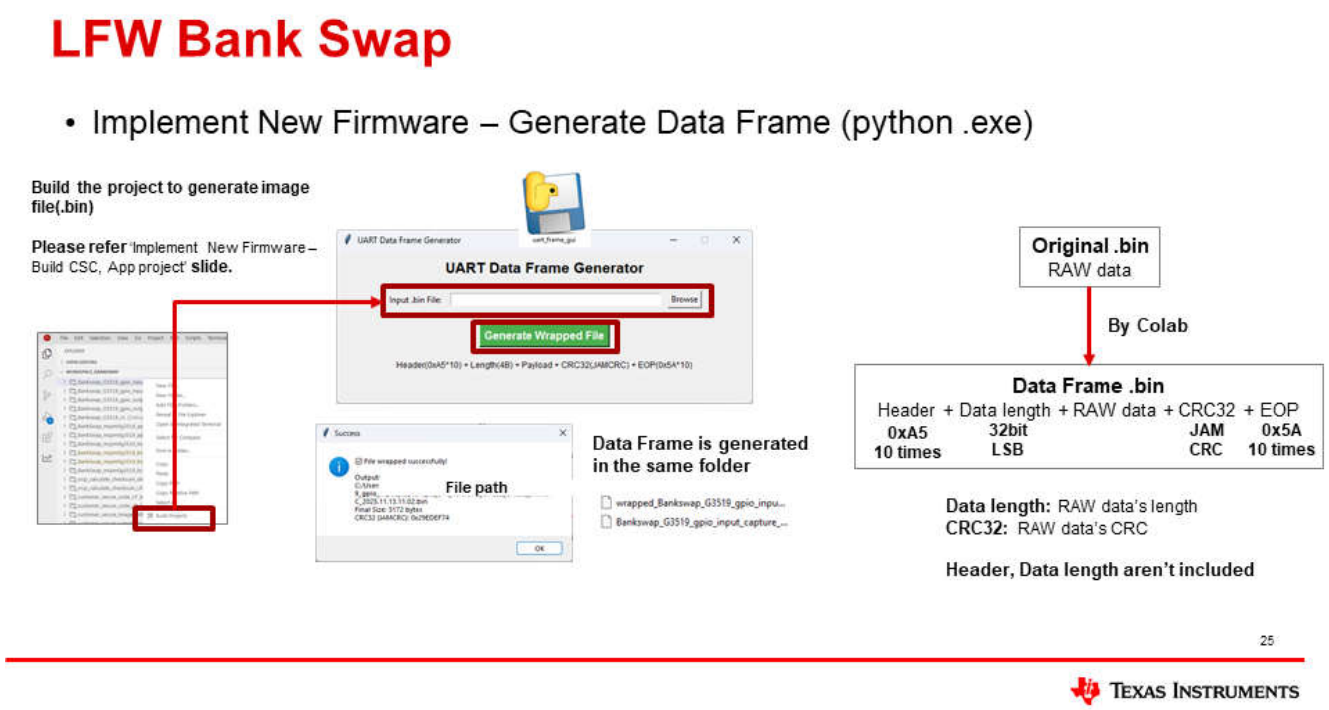

図 2-10. 送信するデータ フレームの生成し

効率性と安定性を向上させるために、データフレーム構造を使用します。ファームウェア データ フレームは、ヘッダ (0xA5 × 10)、データ長 (32 ビット、LSB ファースト)、未加工データ、CRC32 (JAMCRC)、およびパケットの終了 (EOP、0x5A × 10) で構成されます。データ長フィールドは未加工データのサイズを表し、CRC32 値は未加工データに対して計算されます。

TI では、ファームウェア データ フレームの生成を簡素化する実行可能ツールを提供しています。未加工のバイナリ (.bin) ファイルをアップロードすることで、対応するファームウェア データ フレームが自動的に生成されます。

2.4.2.6 PC (Tera Term) の UART 経由で新しい FW を送信

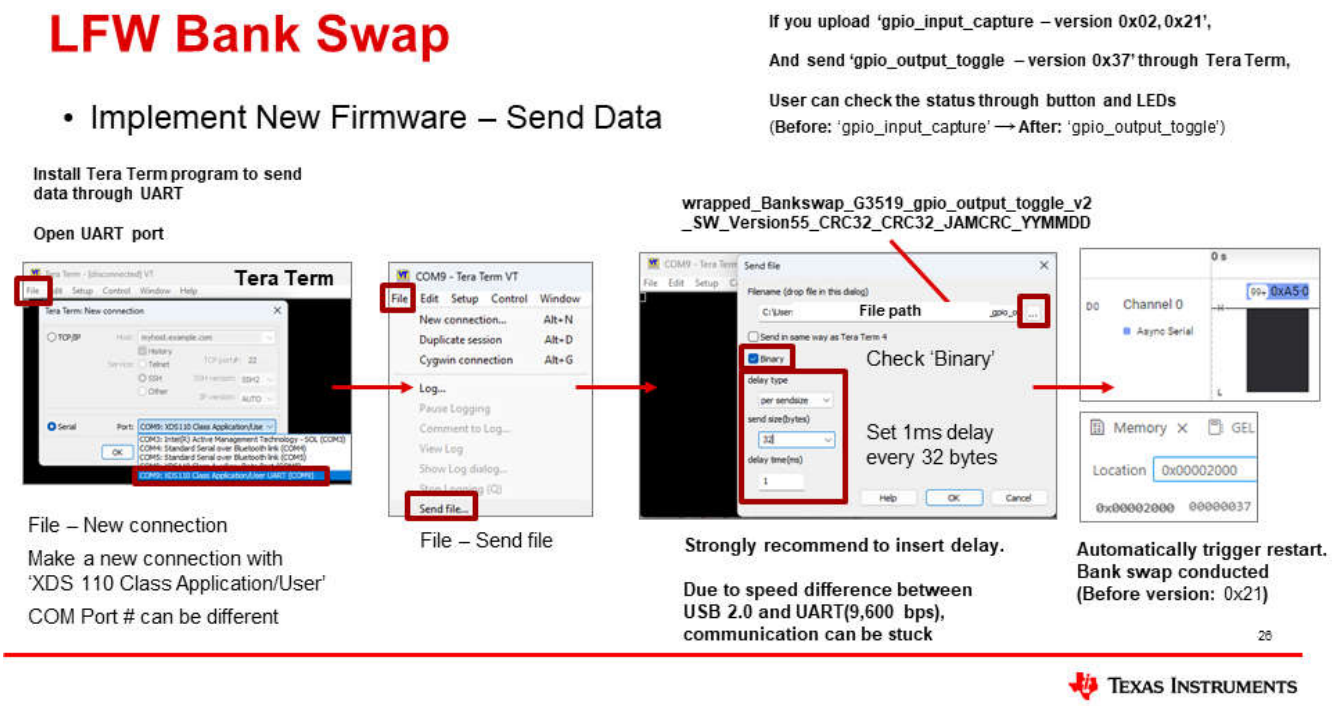

図 2-11. PC の UART 経由による新しい FW の送信

USB-to-UART ブリッジとして機能する XDS-110 を接続します。Tera Term を使用すると、ファームウェア データ フレーム (.bin) がマイコンに転送されます。

USB 2.0 と UART の速度差により、通信がストールすることがあります。この問題に対処するため、TI では遅延挿入を推奨します。この実装では、32 バイトごとに 1ms の遅延が追加されます。この問題は、XDS-110 に関連しており、マイコンには影響しません。

転送が完了すると、アプリケーションは CRC32 を使用してファームウェアを検証します。ファームウェアが有効な場合、バンク 1 にプログラムされます。フラッシュ プロセスが完了すると、アプリケーションは BOOTRST をトリガし、その実行が CSC に転送されます。次に、CSC は、どのバンクに最新のファームウェアが含まれているかをチェックします。ファームウェアのバージョン情報は、各アプリケーション イメージの先頭に保存されます (バンク 0：0x0000.2000、バンク 1：0x0004.2000)。

新しくフラッシュされたファームウェアが最新バージョンである場合、CSC はバンク スワップを開始し、バンク 0 とバンク 1 を交換します。更新されたファームウェアを確認する手順については、次のセクションを参照してください。

2.4.2.7 更新された FW (TI CCS IDE) を確認

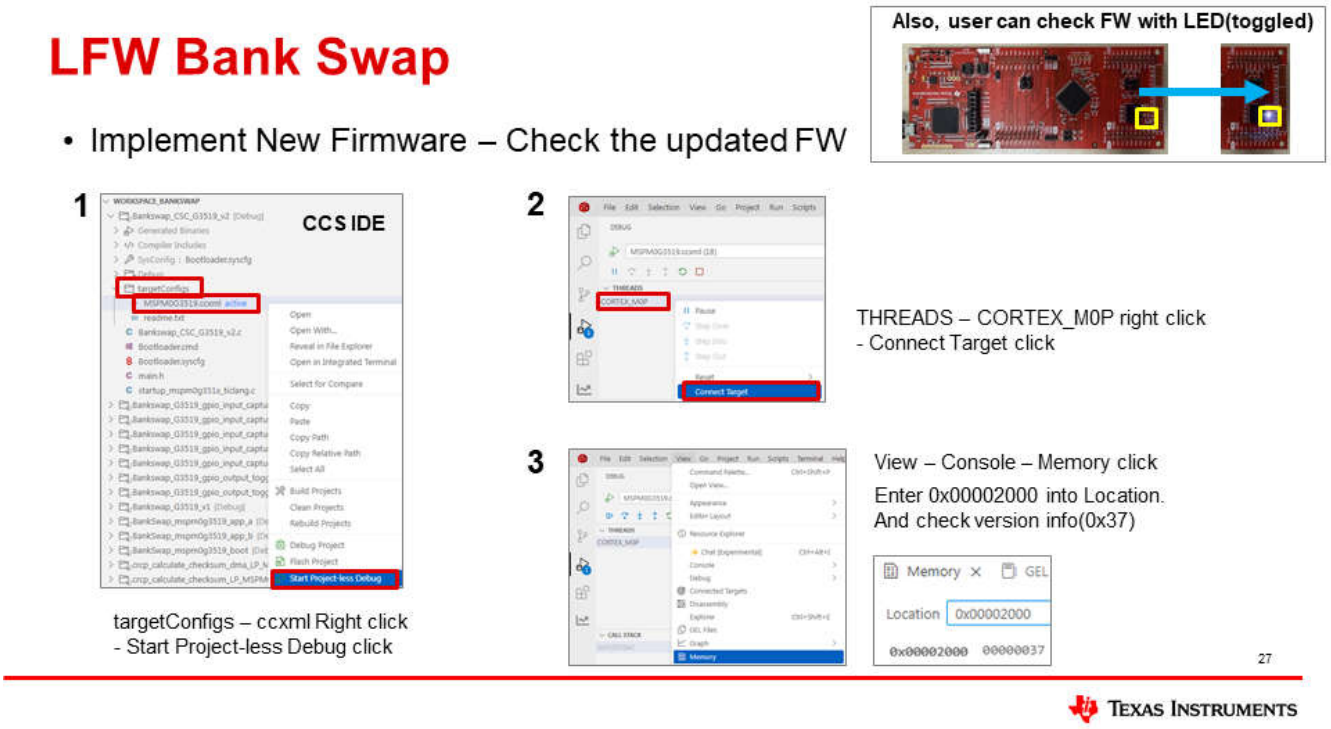

図 2-12. 更新された FW の確認し

ユーザーは、CCS IDE を使用してファームウェアがスワップされたかどうかを確認できます。デバッグ ビューを開き、アドレス 0x0000.2000 のバンク 0 でアプリケーション ファームウェアのバージョンを確認します。ファームウェア バージョンは、LSB に揃えて uint32_t フォーマットで保存されます。

3 まとめ

このドキュメントでは、バンク スワップと CSC の基本理論について説明しています。また、サンプル コードとプログラム (.exe) を使用して、ファームウェアのライブ アップデートを実装する方法について手順を述べています。

この例を使用すると、アプリケーション プログラムの実行中にデバッグ ツール (SWD) を使用せずにファームウェアを更新できます。

これは、バンク スワップを使用したファームウェアのライブ アップデートの基本例であり、ユーザーはこのコードに基づいてセキュリティや安定性などの機能を追加することができます。

4 参考資料

- MSPM0 ファミリのフラッシュ マルチ バンク機能 - https://www.ti.com/lit/an/spradn2/spradn2.pdf?ts=1763307924246&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252Fko-kr%252FMSPM0G3519
- MSPM0 G シリーズ 80MHz マイコン テクニカルリファレンス マニュアル (改訂版 C) - https://www.ti.com/lit/ug/slau846c/slau846c.pdf?ts=1763536485745&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252Fko-kr%252FMSPM0G3519
- Arm M0+ ドキュメント「ベクタ テーブル オフセット レジスタ」- https://developer.arm.com/documentation/dui0662/b/Cortex-M0--Peripherals/System-Control-Block/Vector-Table-Offset-Register

重要なお知らせと免責事項

TI は、技術データと信頼性データ (データシートを含みます)、設計リソース (リファレンス デザインを含みます)、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、TI 製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した TI 製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションに該当する各種規格や、その他のあらゆる安全性、セキュリティ、規制、または他の要件への確実な適合に関する責任を、お客様のみが単独で負うものとします。

上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている TI 製品を使用するアプリケーションの開発の目的でのみ、TI はその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。TI や第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、TI およびその代理人を完全に補償するものとし、TI は一切の責任を拒否します。

TI の製品は、 TI の販売条件 、 TI の総合的な品質ガイドライン 、 ti.com または TI 製品などに関連して提供される他の適用条件に従い提供されます。TI がこれらのリソースを提供することは、適用される TI の保証または他の保証の放棄の拡大や変更を意味するものではありません。 TI がカスタム、またはカスタマー仕様として明示的に指定していない限り、TI の製品は標準的なカタログに掲載される汎用機器です。

お客様がいかなる追加条項または代替条項を提案する場合も、TI はそれらに異議を唱え、拒否します。

最終更新日：2025 年 10 月