参考資料

*Application Note*

# *C2000 F29x マイコンのライブ ファームウェア アップデート (リセットなし)*

*Sira Rao, Alex Wasinger*

**概要**

このドキュメントでは、F29H85x など A/B スワップ可能なフラッシュ バンクを備えたデバイスでの、デバイス リセットなしのライブ ファームウェア アップデート (LFU) について詳しく説明しています。

## 目次

**1 はじめに**....2
**2 LFU のビルディング ブロック**....2
**3 提案された設計の詳細**....2
3.1 フラッシュ バンク構成....2
3.2 LFU の概念とパフォーマンスに影響を及ぼす要因....3
3.3 LFU 用ハードウェア サポート....3
**4 アプリケーションの LFU のフロー**....4
**5 結果と結論**....5
**6 実装例**....5
**7 まとめ**....5
**8 参考資料**....6

# 1 はじめに

サーバー電源のようなアプリケーションでは、ダウンタイムを短縮するためにシステムが継続的に動作することが望まれます。ファームウェアのアップグレード時には通常、バグ修正、新機能の追加、改良などのために、システムの運用を停止することで、ダウンタイムが発生します。これは冗長モジュールで対応できますが、その結果、システム全体としてのコストは増加することになります。ライブ ファームウェア アップデート (LFU) と呼ばれる別の方法により、システムがまだ動作している間にファームウェアをアップデートすることができます。

F29x マイコンでは、新しいファームウェアへの切り替えは、デバイス リセットありとなしの両方で実行できます。このユーザー ガイドは、リセットなしの切り替えを対象としています。リセットありの実装については、『F29H85x のファームウェア ワイヤレス アップグレードの例』を参照してください。**この例は、HS-FS (非セキュア) デバイス向けであることに注意してください。**

# 2 LFU のビルディング ブロック

LFU の設計は、次のようないくつかのビルディング ブロックで構成されています。

- LFU コマンドを発行してアプリケーション イメージを送信するデスクトップ ホスト アプリケーション
- ホストと通信してアプリケーション イメージをプログラムするための、ターゲット デバイスのフラッシュ内の LFU セカンダリ ブートローダー (SBL)
  - 実装例の説明については、『F29H85x のシリアル フラッシュ プログラミング』アプリケーション ノートを参照してください
- LFU サポートを実装したマイコン アプリケーション
- A/B スワップ フラッシュ バンクのサポート付きターゲット マイコン
- LFU サポート付きコンパイラ (将来のコンパイラ リリースで提供予定)

# 3 提案された設計の詳細

## 3.1 フラッシュ バンク構成

LFU は現在、以下に示すフラッシュ MAIN 領域のアドレス マップを備えた HS-FS デバイスのバンク モード 1 でのみ利用できます。

| FRI | READ PORT | SIZE | START ADDRESS | END ADDRESS | FLASH BANKS (SWAP = 0) | FLASH BANKS (SWAP = 1) |
|---|---|---|---|---|---|---|
| FRI-1 (CPU1 program) | RP0 | 1MB | 0x10000000 | 0x100FFFFF | FLC1.B0/B1 | FLC1.B2/B3 |
| | RP1 | 1MB | 0x10100000 | 0x101FFFFF | FLC2.B0/B1 | FLC2.B2/B3 |
| | RP2 | 1MB | 0x10200000 | 0x102FFFFF | N/A | N/A |
| | RP3 | 1MB | 0x10300000 | 0x103FFFFF | N/A | N/A |
| FRI-2 (CPU3 program) | RP0 | 1MB | 0x10400000 | 0x104FFFFF | N/A | N/A |
| | RP1 | 1MB | 0x10500000 | 0x105FFFFF | N/A | N/A |
| FRI-3 (Update region) | RP0 | 1MB | 0x10600000 | 0x106FFFFF | FLC1.B2/B3 | FLC1.B0/B1 |
| | RP1 | 1MB | 0x10700000 | 0x107FFFFF | FLC2.B2/B3 | FLC2.B0/B1 |

**図 3-1. フラッシュ MAIN 領域のアドレス マップ**

SWAP = 0 の場合、アクティブなフラッシュ バンクは FLC1.B0/B1 および FLC2.B0/B1 です。SWAP = 1 の場合、アクティブなフラッシュ バンクは FLC1.B2/B3 および FLC2.B2/B3 です。アクティブと非アクティブの両方のフラッシュ バンクは、以下に示すように、SBL 用のセクションとアプリケーション用のセクションに分割されています。

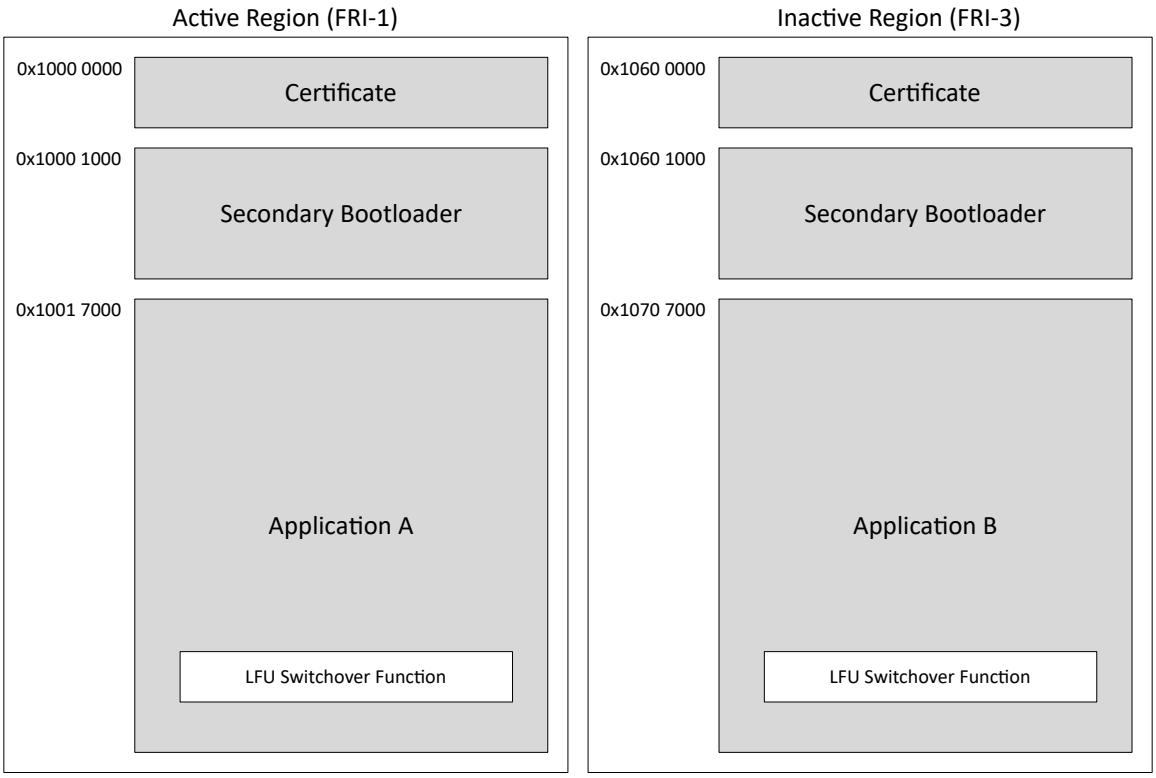


**図 3-2. フラッシュ バンク**

ファームウェア更新を実行する際は、非アクティブなフラッシュ バンクが消去され、新しい証明書、SBL、およびアプリケーション イメージがプログラムされます。

## 3.2 LFU の概念とパフォーマンスに影響を及ぼす要因

LFU 対応ファームウェアを作成するときの主な検討事項は、LFU 中の動作の継続性と LFU 切り替え時間です。これら 2 つは密接に関連しています。動作の継続性は、状態を維持し、RAM 内の既存の静的変数とグローバル変数をファームウェア バージョン間で同じアドレスに保ち、新しいファームウェアがアクティブになったときにそれらの変数を再初期化しないことによって実現されます。LFU 切り替え時間は、ファームウェアを変更している間にデバイスが割り込みを無効化して費やす時間であり、フラッシュ バンクとベクタ テーブルの A/B スワップによって最小化されます。

## 3.3 LFU 用ハードウェア サポート

### *3.3.1 A/B スワップ可能なフラッシュ バンク*

*図 3-1* に示すように、アクティブおよび非アクティブのフラッシュ バンクのハードウェア サポートがあります。これらのバンクは、SSU_GEN_REGS.BANKMAP レジスタを XOR することで、いつでもスワップできます。

### *3.3.2 割り込みベクタ テーブル スワップ*

割り込みベクタ テーブルの更新は、LFU ルーチンの中で最も時間を要する部分の 1 つです。切り替え時間を短縮するために、アプリケーションの実行中にシャドウ ベクタ テーブルが更新されます。切り替え時間中、割り込みが無効化されている間、シャドウとアクティブのベクタ テーブルが PIPE_REGS.INT_VECT_MAPPING レジスタを使用してスワップされます。

# 4 アプリケーションの LFU のフロー

以下で、コードおよびサンプル プロジェクトを参照する際に、SBL = flash_based_uart_sbl_with_lfu および LFU アプリケーション = lfu_uart_cpu1_application とします。

LFU に関連する詳細手順を以下に示します。

1. *SBL をブートし、アプリケーションを実行します*: デバイス リセット時に、SBL 内で実行が開始されます。通信ペリフェラルが初期化され、アプリケーションのコードスタート ルーチンに分岐します。コードでは、これは SBL の main() 関数 (flash_based_uart_sbl.c) で行われます。
2. *LFU を開始します*: ユーザーは、ホストで起動される LFU コマンドを使用し、ターゲット マイコン上で LFU を呼び出します。
3. *アプリケーションが LFU コマンドを受信します*: アプリケーションは、自らの通信ペリフェラル ISR で LFU コマンドを受信します。コードでは、これは LFU アプリケーションの SBL_uartNotifyISR() (uart_led_blinky_cpu1.c) で行われます。この関数は、SBL にジャンプしてコマンドを処理するようアプリケーションに通知するためのフラグを設定します。
4. *LFU コマンドを処理します*: アプリケーションは SBL に分岐し、LFU コマンドを解析して処理します。コードでは、これは SBL の commandJumpTable() 関数 (flash_based_uart_sbl.c) で行われます。この関数は、コマンドを読み取り、適切な動作を実行します。
5. *新しいファームウェアをダウンロードしてフラッシュにプログラムします*: SBL の LFU フローは、ホストからアプリケーションのイメージを受信し、そのイメージを非アクティブなフラッシュ バンクにプログラムします。古いファームウェアのバックグラウンド タスク機能の実行は停止していますが、アプリケーションに影響を及ぼさないよう、制御 ISR は引き続き利用可能です。コードでは、これは SBL の cpu1LFUFlow() 関数 (sbl_command_flow.c) で行われます。
6. *アプリケーションに戻り、成功を報告します*: ファームウェア イメージが正常にプログラムされると、SBL はアプリケーションに分岐し、LFU コマンドが成功したことを報告します。
7. *LFU 切り替え関数を RAM にコピーします*: アプリケーションは、実行可能ではない非アクティブなフラッシュにある新しいファームウェアの LFU 切り替え関数を RAM にコピーし、その関数に分岐します。古いファームウェアは新しいファームウェアの切り替え関数のコピーを実行するので、固定アドレスに配置する必要があります。コードでは、LFU アプリケーションの performLFUSwitchover() 関数 (lfu_switchover.c) で実行されます。
8. *新しい PIPE ベクタ テーブルを初期化します*: シャドウ PIPE ベクタ テーブルに新しいファームウェアの ISR が設定されます。コードでは、これは LFU アプリケーションの lfuSwapBanks() 関数 (lfu_switchover.c) で実行されます。
9. *最適な LFU 切り替えポイントを待ちます*: ソフトウェア フラグ付きの単純なステート マシンを使用して、制御 ISR の終了とアイドル時間の開始を判定します。これは、制御ループ割り込み間のアイドル時間を最大限に利用できるため、切り替えの実行に最適なタイミングです。コードでは、lfuSwapBanks 関数 (lfu_switchover.c) が、INT_myCPUTIMER0_ISR() ルーチン (uart_led_blinky_cpu1.c) の最後に設定される lfu_switchover_proceed フラグを読み取ることで、これを実現します。
10. *LFU 切り替えを実行します*:
    a. 割り込みが無効化されます
    b. 割り込みテーブルがスワップされます
    c. アクティブと非アクティブのフラッシュ バンクがスワップされます
    d. スタック ポインタが再初期化されます
    e. 割り込みが再度有効になります
    f. 実行は新しいファームウェアの main() ルーチンに分岐します

    コードでは、これは LFU アプリケーションの lfuSwapBanks() 関数 (lfu_switchover.c) のセクションで、タイムクリティカルな位相としてマークされています。
11. 新しいファームウェアではデバイスの初期化がスキップされ、バックグラウンド制御ループが直ちに実行を開始します

## 5 結果と結論

図 5-1 に、実際の LFU 切り替えを示します。上の波形は LFU 切り替えを表し、下側の波形は ISR CPU 負荷を示しています。切り替えは 300ns (約 60 CPU サイクル) で発生します。

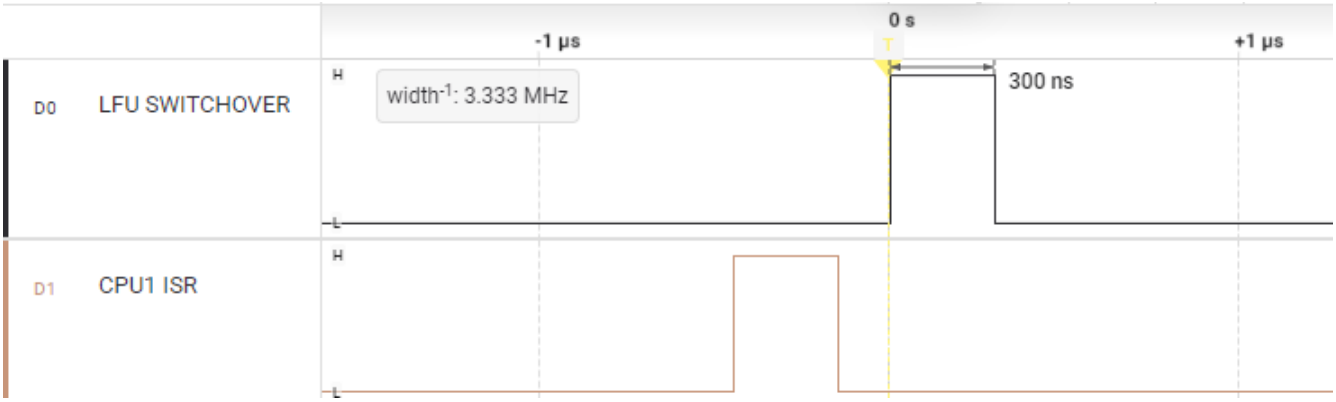


**図 5-1. LFU 切り替え**

## 6 実装例

### F29H85x

F29 SDK で入手可能な場所: F29_SDK/examples/driverlib/single_core/flash/flash_based_sbl_with_lfu

## 7 まとめ

このアプリケーションノートは、リアルタイム制御アプリケーション向けの LFU、特にダウンタイムなしで動作を必要とする高可用性システムの体系的な実装を示します。新しいアプリケーションの LFU ソフトウェア フロー、ハードウェア LFU サポートなど、利用可能な LFU ビルディング ブロックを使用することで、新しいファームウェアへの切り替えは 60 CPU クロック サイクル以内に完了できます。

# 8 参考資料

1. テキサス インスツルメンツ、『F29H85x および F29P58x リアルタイム マイコン テクニカル リファレンス マニュアル』テクニカル リファレンス マニュアル
2. テキサス インスツルメンツ、『F29H85x、F29P58x、F29P32x リアルタイム マイコン データシート』データシート
3. テキサス インスツルメンツ、『F29H85x のシリアル フラッシュ プログラミング』アプリケーション ノート
4. テキサス インスツルメンツ、『C2000 マイコンでのデバイス リセットなしのライブ ファームウェア アップデート』アプリケーション ノート

# 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