PIC18-Q24 ファミリの MCU によるイミュータブルブートローダの構築

Robert Perkel （Applications Engineer）

PIC18-Q24 ファミリのマイクロコントローラを使用してイミュータブルブートローダを作成する方法の詳細をご覧ください。

アプリケーションのプロトタイピングを行う場合、多くの場合、マイクロコントローラ上のファームウェアは必要なアプリケーションコードだけです。ファームウェアのバグが発見された場合、アプリケーションを再プログラムして再実行するのは比較的簡単なプロセスです。しかし、設計が大量生産されると、すべてのデバイスを追跡して再プログラムすることは現実的ではありません。

この問題の解決策は、ブートローダーとして知られています。ブートローダーは、マイクロコントローラーで「起動」し、アプリケーションの前にコア機能を初期化するように設計された小さなコードです。8 ビットマイクロコントローラの場合、通常、ブートローダの主な目的は、プログラマを必要とせずにファームウェアのアップグレードを可能にすることです。通常、新しいファームウェアは、アプリケーションに応じて、UART、SPI、I2C、および/または USB から受信されます。

ブートローダはゼロから作成することもできますが、周辺ハードウェアおよびソフトウェアライブラリを設定するための無料のグラフィックツールである MPLAB® Code Configurator(MCC)には、事前に作成されたドライバが用意されています。デスクトップコンピュータ用の Python ライブラリは、UART 経由でファームウェアライブラリと通信してファームウェアをアップデートするために提供されています。現在(執筆時点)、ライブラリは UART 経由アップロードをサポートしており、SPI、I^2C、および USB のサポートが計画されています。8 ビット ブートローダー ライブラリの詳細については、こちらの **Web ページ**を参照してください。

ブートローダを使用するもう 1 つの利点は、マイクロコントローラを永久にロックダウンし、外部プログラミングを防止する一方で、将来ファームウェアをアップグレードする機能を保持するオプションです。PIC18-Q24 ファミリのマイクロコントローラには、ブートローダベースのアプローチを補完するいくつかの強化されたコード保護機能が含まれています。

- **プログラミングおよびデバッグ インターフェイスの無効化 (PDID)**
- **ソフトウェアインサーキットシリアルプログラミング(ICSP)**

- **コード保護 (メモリ読み取り無効)**
- **ワンタイム プログラマブル ストレージ エリア フラッシュ (SAF)**
- **ブート/アプリケーションコードのセグメント書き込み保護**

PDID は、外部のプログラマとデバッガの永続的で不可逆的なロックアウトです。これにより、マイクロコントローラが外部ツールによって再プログラムされるのを防ぎます。この機能は、ソフトウェア ICSP イネーブル機能と連動し、ブートローダがプログラムの読み取り/検証(詳細は次の文を参照)や実行時のデバイス検出など、一部の ICSP インターフェイス機能を一時的に再度有効にすることができますが、これには書き込み操作や消去操作は含まれません。コード保護機能を使用して、ICSP の読み取りおよび検証操作をブロックできます。
もう 1 つの強化されたコード保護機能は、ワンタイムプログラマブル SAF です。SAF は、データを格納する目的で、プログラムのフラッシュ メモリの一部を実行不可としてマークする機能です。これにより、フラッシュメモリ領域のこの領域はワンタイムプログラマブルになり、外部ツールとマイクロコントローラ自体の両方から消去または再書き込みされるのを防ぎます。
最後に、PIC18-Q24 ファミリには、ブート、アプリケーション、EEPROM メモリ用のセグメント化された書き込み保護が含まれています。これらのビットは、外部プログラマが選択したメモリ領域を変更するのを防ぎますが、マイクロコントローラからの内部自己書込みは防止しません。
これらの強化されたコード保護ビットをブートローダと組み合わせて使用すると、「イミュータブルブートローダ」、つまり上書きや消去ができないブートローダを作成できます。これは、PDID やコード保護と併用して、ファームウェアをロックダウンし、知的財産を保護することもできます。

PIC18-Q24 ファミリのマイクロコントローラの詳細については、こちらの Web ページをご覧ください。