参考資料

User's Guide

MSPM0 ブートローダ

TEXAS INSTRUMENTS

概要

MSPM0 ブートローダー (BSL とも呼ばれる) は、標準のシリアル インターフェイスを使用して、デバイスのメモリ (アプリケーション メモリ (フラッシュ) とデータ メモリ (RAM) の両方) を変更します。

ブートローダーを起動するには、特定のエントリ シーケンスに従う必要があります。電源投入時に毎回実行されることはありません。また、ブートローディング セッションをリセットまたは特定のコマンドを使用して終了し、アプリケーションを起動することができます。

目次

商標

1 はじめに

1.1 BSL 機能の概要

ブートストラップ ローダ (BSL) は、UART や I2C のような標準のシリアル インターフェイスを使用してデバイスのメモリをプログラムまたは検証できます。

シリアル インターフェイスでアクセス可能な ROM BSL の主な機能には、次のものがあります。

- フラッシュ メモリのプログラミングと消去
- プログラミングを検証するため、コードまたはデータ領域 (最小領域サイズ 1KB) の 32 ビット CRC を返す機能
- コードまたはデータの読み出しをイネーブルにする機能 (デフォルトではディセーブル)
- メイン フラッシュへのポインタを使用して、ファームウェアのバージョン番号を返す機能
- ハードウェアの GPIO を起動するように指定する機能
- アクセスを常に 256 ビットのパスワードで保護
- ブルートフォース攻撃に抵抗するため、セキュリティ アラート処理を構成可能
- 新しいインターフェイスをフラッシュ プラグインとして追加する機能
- カスタム ブートローダーを使用する機能

1.2 用語

ブートローダ (BSL) - データをデバイスのメモリにロードするために使用されるブート ルーチン

ブートコード (BCR) - BOOT リセットの後で実行されるスタートアップ ルーチンで、アプリケーションを実行するためにデバイスを構成します

BCR の構成 - メイン以外のフラッシュ メモリに置かれている、ユーザーが構成可能なすべてのブートコード パラメータを含む構成構造

BSL の構成 - メイン以外のフラッシュ メモリに置かれている、ユーザーが構成可能なすべてのブートローダ パラメータを含む構成構造

1.3 その他の資料

1. テクニカル リファレンス マニュアル
    a. MSPM0 G シリーズ マイコン
    b. MSPM0 L シリーズ マイコン
    c. MSPM0 H シリーズ マイコン
    d. MSPM0 C シリーズ マイコン
2. ツールおよびサンプル
    a. MSPM0 ソフトウェア開発キット
3. アプリケーション ノート
    a. MSPM0 ブートローダーの実装

2 BSL オプション

MSPM0 デバイスは、デフォルトでサポートされている BSL 機能に基づいて、3 つのカテゴリに分類できます。

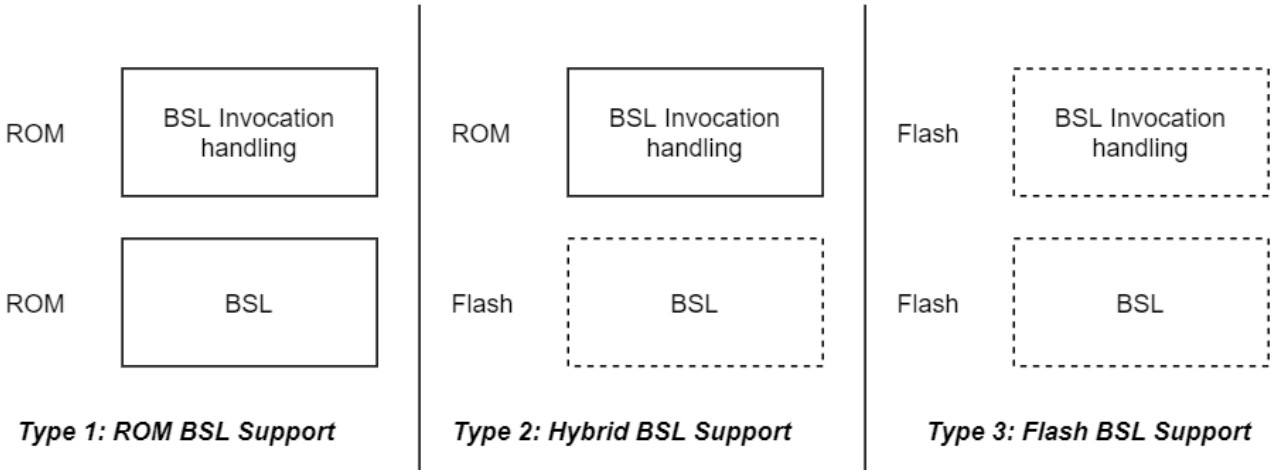

図 2-1. MSPM0 デバイスの BSL オプション

タイプ 1:ROM BSL のサポート

ROM BSL 実装のデバイスでは、ブートローダーは ROM に組み込まれています。これは、メモリを更新するための呼び出しとコア操作を処理します。このユーザーガイドの次のいくつかのセクションでは、ROM BSL の機能、アーキテクチャ、プロトコルについて詳しく説明します。

- UART と I2C の各インターフェイスをサポート
- 工場出荷時の新しいデバイスですぐに使用可能
- これらのデバイスでは、セカンダリ BSL またはインターフェイス プラグインの形で、フラッシュ メモリへのカスタム実装もサポート。

セカンダリ BSL またはインターフェイス プラグインの詳細については、以降のセクションを参照してください。

タイプ 2:ハイブリッド BSL のサポート

これらのデバイスでは、応答は ROM とフラッシュの間で共有されます。ROM ブートコード (BCR) は、BCR の構成でブートローダーがイネーブルのときに、ブートごとに呼び出し条件をチェックします。この部品は ROM BSL と同様に扱われます。条件が満たされると、制御がフラッシュ メモリ内の BSL に転送され、コアの BSL 動作が実装されます。

- デバイスで使用可能なすべてのインターフェイスをサポートするカスタム BSL 実装も可能
- 新しいデバイスには、フラッシュ BSL (およびオプションではアプリケーション イメージ) をデバッガ インターフェイス経由で初めてロードする必要がある。

このカテゴリのフラッシュ BSL の実装および構成プロセスは、セカンダリ BSL と同じ手順に従います。詳細については、セクション 6 を参照してください。

タイプ 3:フラッシュ BSL のサポート

これらのデバイスには、BSL アクティビティ用の ROM サポートがありません。フラッシュの実装では、BSL 起動基準と機能を完全に処理する必要があります。

- デバイスで使用可能なすべてのインターフェイスをサポートするカスタム BSL 実装も可能
- 新しいデバイスには、フラッシュ BSL (およびオプションではアプリケーション イメージ) をデバッガ インターフェイス経由で初めてロードする必要がある。

表 2-1. デバイス内での BSL のサポート

デバイス ファミリ	デバイス	ROM BSL のサポート	ハイブリッド BSL のサポート	フラッシュ BSL のサポート
MSPM0Gx	MSPM0Gx	✓		
MSPM0Lx	MSPM0Lx	✓		
MSPM0Cx	MSPM0C1105/6		✓	
	MSPM0C1104			✓
MSPM0Hx	MSPM0H321x		✓	

3 BSL のアーキテクチャ

3.1 設計

ブートローダの有効な起動条件が検出されると、ブートコードによってブートローダが起動されます。起動するのは、BCR の構成の BSL モード フィールドでブートローダがイネーブルのときのみです。

ブートローダが開始されると、最初に「初期フェーズを実行します。ここで、BSL の構成の初期チェックが行われ、デバイスはブートローダ動作用に構成されます。

次に、ブートローダは「インターフェイス自動検出」フェーズに入ります。このフェーズでは、BSL は使用可能なすべての BSL ROM インターフェイスと、フラッシュ プラグイン インターフェイス (登録されている場合) を構成します。その後、BSL はすべてのインターフェイスを 1 つずつ使用してデータをポーリングします。いずれかのインターフェイスで有効な接続パケットが受信されると、そのインターフェイスがアクティブなインターフェイスとして以後の通信に使用され、他のすべてのインターフェイスはディセーブルされます。インターフェイス検出は 10 秒間実行されます。インターフェイスが検出されない場合、デバイスは STANDBY モードになります。

次に、ブートローダーは「コマンド受信」フェーズに入ります。このフェーズでは、BSL はホストからのコマンドを無限ループで待機します。有効なコマンドが受信されると、そのコマンドが処理され、BSL コアからの応答がホストに返送されます。それから、BSL はループに戻り、以後のコマンドを待ちます。「アプリケーションの開始」コマンドが受信されると、ブートローダはシステム リセットをトリガします。その後でブートコードが実行され、アプリケーションが起動されます。このフェーズのタイムアウトも 10 秒です。有効なコマンドが受信されない場合、ブートローダはロックされ、SLEEP モードに移行します。

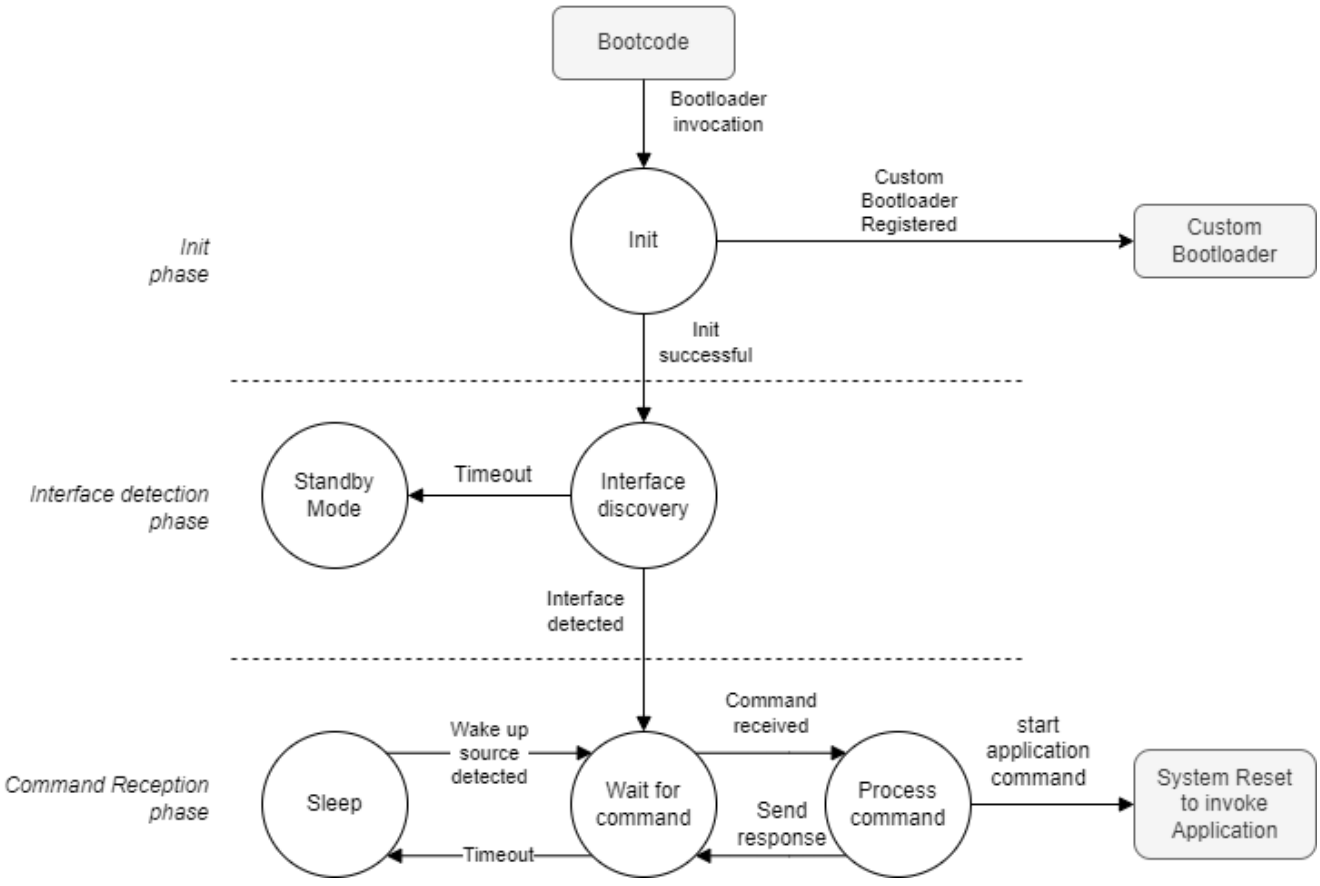

図 3-1. BSL アーキテクチャ

3.1.1 タイムアウト機能

アクティビティが検出されないとブートローダはタイムアウトし、低消費電力モードに移行して消費電力を低減します。

これは、次の 2 つのフェーズで実装されています。

1. インターフェイスの自動検出
2. コマンドの受信

3.1.1.1 インターフェイスの自動検出

インターフェイスの検出フェーズで、どのインターフェイスからも有効な接続コマンドが 10 秒間受信されない場合、ブートローダは STANDBY モードに移行します。

この状態を終了し、再度 BSL 起動条件を作り出してブートローダを使用するには、POR が必要です。

3.1.1.2 コマンドの受信

コマンド受信フェーズで、有効なコマンドが 10 秒間受信されない場合、ブートローダは SLEEP モードに移行します。SLEEP モードからデバイスをウェークアップするには、アクティブ インターフェイスでデータ トランザクションを実行する必要があります。

攻撃にさらされる場所を減らすため、ブートローダは SLEEP モードに移行する前にロックされます。このため、低消費電力モードからウェークアップした後で、256 ビットの BSL パスワードを送信してブートローダを再度ロック解除する必要があります (「ブートローダのロック解除コマンド」を参照)。

注

ブートローダーは、タイムアウトのチェックに LFCLK を使用します。アプリケーションが外部クロックを LFCLK のソースとして構成した場合、アプリケーションの BSL 要求によって呼び出すとき、BSL でも同じクロックが使用されます

3.2 BSL の起動

BSL 起動の条件が満たされ、BCR の構成でブートローダーがイネーブルのとき、ブートローダーはブート コードからのみ起動されます。

BCR の構成で高速ブート モードがイネーブルのとき、ブートローダはメールボックスのデバッグ コマンドとアプリケーション要求でのみブートローダを起動できます。実行時間を短縮するため、他のチェックはスキップされます。

3.2.1 ブランク・デバイス

ブートコードは、スタック・ポインタ (0x00000000) およびリセット・ベクトル (0x00000004) アドレスの消去状態をチェックして、ブランク・デバイスを検出します。両方のフラッシュ・メモリ・アドレスがブランクであることが検出されると、ブートローダが呼び出されます。

3.2.2 アプリケーション要求

アプリケーションからブートローダを起動するには、RESETLEVEL を BOOTLOADERENTRY に設定し、RESETCMD レジスタからリセットをトリガします。このシーケンスによってシステムがリセットされ、ブートコードが実行されてブートローダが呼び出されます。

システム・リセットが発行されるので、アプリケーションの終了時にすべてのペリフェラル構成がリセットされます。

3.2.3 GPIO ベースの起動

BSL の起動に使用される GPIO は、メイン メモリ以外の BSL の構成で設定できます。

新規デバイスは、テキサス・インスツルメンツによってデフォルト ピンが BSL の構成でプログラムされています。

GPIO ピン ベースの起動は、BCR の構成でディセーブルできます。デフォルトではイネーブルです。

POR の前に GPIO をアサートし、POR の後で少なくとも T_start ms の間、その状態を維持します。その後で、GPIO ピンの状態をアサート解除できます。

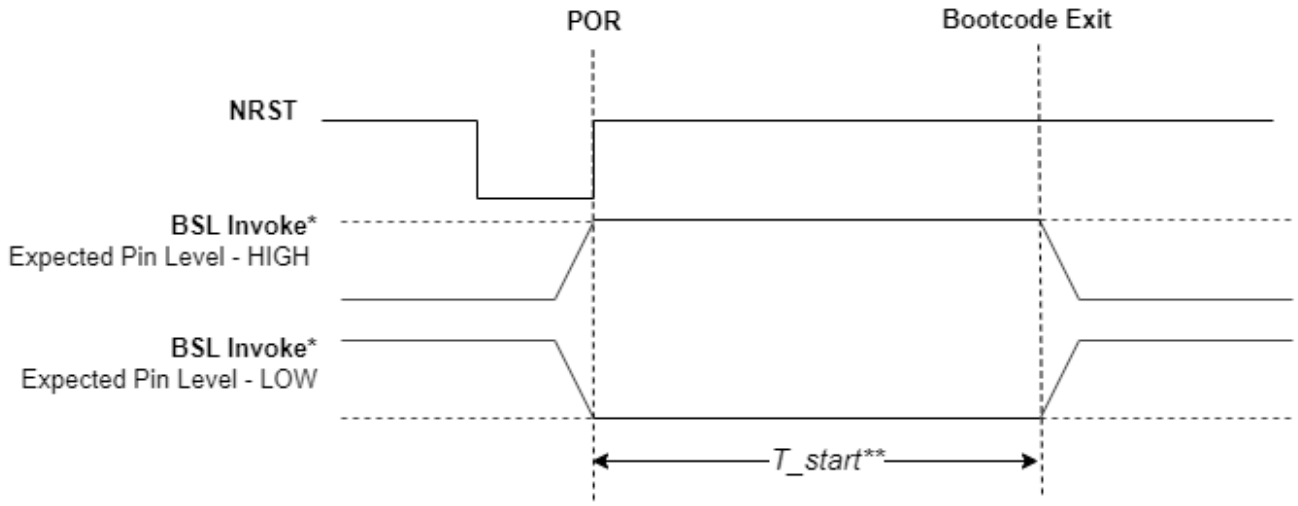

図 3-2. GPIO からの起動

** - 「BSL 起動」および「期待されるピンのレベル」として使用される GPIO ピンは、BSL の構成で設定できます*

** - T_start は、デバイス固有のデータシートに記載されているコールド ブートのスタートアップ時間を表します*

注

ピン ベースの BSL 起動がイネーブルのとき、構成された GPIO ピンは High または Low にします。フローティング状態のままにしておくと、予期しない BSL 実行が発生する可能性があります。

3.2.4 メールボックスのデバッグ コマンド

デバッグ インターフェイスが使用可能な場合、デバッグ サブシステム メールボックス (DSSM) 経由でブートローダ起動コマンドを送信できます。

DSSM コマンドの使用方法の詳細については、『MSPM0x テクニカル リファレンス マニュアル』の「DSSM コマンド」セクションを参照してください。

3.2.5 プリブート アプリケーションの検証

BCR の構成でアプリケーションの CRC/ハッシュ検証を構成すると、ブートごとにアプリケーションのメモリの整合性が検証されます。検証が失敗し、ブートローダーがイネーブルの場合、ブートローダーが起動されます。ブートローダーがディセーブルの場合、ブートローダーはブート障害を発生させます。詳細については、デバイスのテクニカル リファレンス マニュアルの「ブート構成」セクションを参照してください。

3.3 メモリ

3.3.1 SRAM メモリの使用法

SRAM メモリのレイアウトでは、ブートローダの動作に使用されるメモリについて説明します。

- データとスタックのセクション - BSL が動作するために使用されます。ブートローダを終了すると、SRAM のこれらのセクションはクリアされます。
- 可変バッファ・スペース - BSL の通信中に送受信されるデータ・パケットを格納するために使用されるバッファ空間

ホストによる読み取り / 書き込みアクセスが可能な SRAM メモリは、BSL バッファの開始アドレスから [SRAM の終了アドレス - 0x120] です。ここで、SRAM の終了アドレスは、各デバイスで利用可能な SRAM メモリによって決定されます。可変バッファ領域と同じ SRAM 領域が共有されるため、SRAM の書き込み / 読み取り動作中に上書きされる可能性があります。

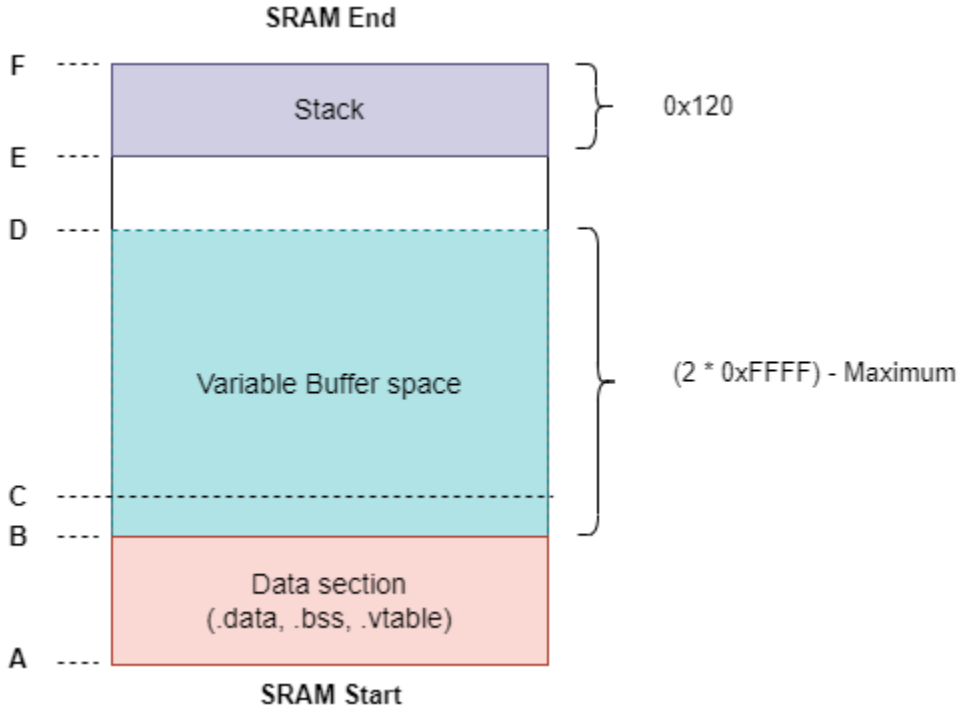

図 3-3. SRAM の使用法

A - SRAM 開始アドレス (0x20000000)

B - フラッシュ・プラグイン・インターフェイスが登録されていないとき、「デバイス情報の取得」コマンドの応答から取得できる「BSL バッファの開始アドレス」

C - 「デバイス情報の取得」コマンドへの応答から取得できる「BSL バッファの開始アドレス」。フラッシュ・プラグイン・インターフェイスが登録されていない場合、B と同じです

D - BSL バッファの終了アドレス = BSL バッファの開始アドレス + (2 * BSL の最大バッファ・サイズ)。ここで、BSL バッファの開始アドレスと BSL の最大バッファ・サイズは「デバイス情報の取得」コマンドの応答から取得できます

E - スタック割り当ての開始アドレス (E - 0x120)。「BSL の最大バッファ・サイズ」が 0xFFFF より小さい場合、これは「D」と同じです

F - デバイスで使用可能な SRAM メモリの終了アドレス。この値は、デバイス固有のデータシートで確認してください。

セクション B～C:

- BSL 構成に登録されているとき、フラッシュ・プラグインの動作に割り当てられるデータ・セクション

セクション C～D:

- データ・パケットの格納に使用されるバッファ領域
- 最大サイズは (2 * 0xFFFF)

セクション C～E:

- BSL コマンドによる SRAM の読み取りおよび書き込み動作に使用可能なメモリ

3.4 BSL の構成

メイン フラッシュ メモリ以外の BSL の構成では、BSL で使用される特定のパラメータをユーザーがカスタマイズできます。

利用可能な構成の詳細については、『MSPM0 テクニカル リファレンス マニュアル』の「構成メモリ」セクションを参照してください。

3.5 BSL のステータス

BSL のステータスには、BSL が応答しなくなったとき、その原因である、実行中に発生したエラーの詳細が示されます。この 32 ビットのステータス情報は、特定の SRAM アドレス「0x200000C0」に格納され、BCR の構成でデバッグ アクセスがイネーブルになっていれば、デバッガから読み取ることができます。

表 3-1. BSL のステータスの解釈

バイト 4	バイト 3	バイト 2	バイト 1
予約済み	ハードウェア エラー	ハードウェア エラーの詳細	ソフトウェア エラー

エラー	説明
0x01	BSL の構成の CRC エラー

ハードウェア エラー

エラー	説明	ハードウェア エラーの詳細
0x07	NMI 例外	NMIIDX - NMI 割り込みインデックス レジスタのデータ

4 ブートローダのプロトコル

4.1 パケット・フォーマット

BSL のデータ・パケットは階層構造です。BSL のコア・コマンドには、BSL によって処理される実際のコマンド・データが含まれています。標準の BSL コマンドに加えて、各コア・コマンドの前後に、ペリフェラル・インターフェイス・コード (PI コード) と呼ばれるラッパー・データを置くことができます。このラッパー・データは、使用されているペリフェラルやプロトコルに固有の情報で、BSL コア・コマンドを正しく送信するための情報が含まれています。ラッパーとコア・コマンドを組み合わせると、BSL データ・パケットが構成されます

PI コード	BSL コア・データ	PI コード

4.2 UART および I2C BSL プロトコル

UART および I2C BSL プロトコルのデータ パケットの構造は次のとおりです。

- **ヘッダー**のバイトは、使用されているプロトコルとパケットのタイプ (コマンドまたは応答パケット) を示します。
- **長さ**フィールドには、BSL コア データのサイズがバイト単位で含まれます。
- **BSL コア データ**には、コマンドに必要なコマンド / 応答 ID とアドレスおよびデータが含まれています
- **CRC** フィールドには、整合性をチェックするために「BSL コア データ」内のデータについて計算された CRC が含まれています。CRC は、デバイスに応じて、32 ビット CRC または 16 ビット CRC のいずれかになります。どちらの場合も、フィールドの長さは一定になります。16 ビット CRC の場合、残りのバイトには 0 が付加されます。

PI コード		BSL コア データ	PI コード
ヘッダー (1 バイト)	長さ (2 バイト)	BSL コアのコマンド / 応答	CRC (4 バイト)

データ パケットは、コア データ フィールドによってコマンド パケットまたは応答パケットに分類されます。

コマンド パケットは、BSL に送信される最初のパケットです。2 番目のパケットは、BSL から受信される応答パケットです。応答パケットには、BSL アクノリッジと BSL コア応答の 2 つのコンポーネントが含まれています。このうち、アクノリッジは送信されるすべてのコマンド パケットについて、BSL から受信されます。これに対して、BSL コア応答はすべてのコマンドについて受信されるわけではありません。

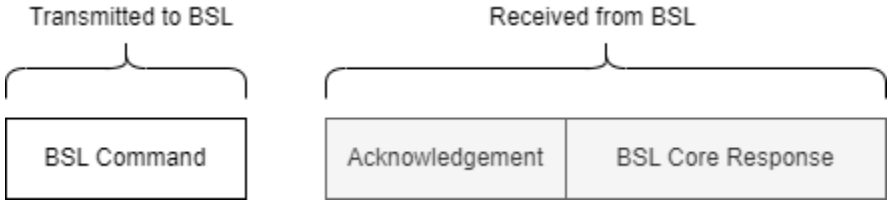

図 4-1. BSL プロトコル

4.2.1 BSL アクノリッジ

BSL ソフトウェアのペリフェラル・インターフェイス・セクションは、BSL データ・パケットのラッパー・セクションを解析します。データ送信でエラーが発生した場合、エラー・メッセージが直ちに送信されます。すべてのデータが正常に受信された後に ACK が送信されます。これは、データ・パケットが正しくフォーマットされ、BSL コア・ソフトウェアに渡されて解釈が行われたという意味で、コマンドが正しく実行されたという意味ではなく、コマンドが有効であったことも保証されていません。

BSL プロトコルでは、送信されるすべての BSL データ・パケットに対して、送信されるすべての BSL データ・パケットに加えて、単一バイトのアクノリッジ応答が行われると規定されています。BSL からのアクノリッジ応答値を、表に示します。ACK 以外のアクノリッジ・バイトが送信された場合、BSL は BSL データ・パケットを送信しません。ホスト・プログラマはアクノリッジ・エラーをチェックし、送信を再試行する必要があります。

データ	意味
0x00	BSL_ACK (パケットを正常に受信した)

データ	意味
0x51	BSL_ERROR_HEADER_INCORRECT
0x52	BSL_ERROR_CHECKSUM_INCORRECT
0x53	BSL_ERROR_PACKET_SIZE_ZERO
0x54	BSL_ERROR_PACKET_SIZE_TOO_BIG
0x55	BSL_ERROR_UNKNOWN_ERROR
0x56	BSL_ERROR_UNKNOWN_BAUD_RATE

4.2.2 ペリフェラルの構成

4.2.2.1 UART

次の構成で UART がイネーブルです。

- UART0 が使用されます
- ボーレートはデフォルトで 9600bps です。これは、「ボーレートの変更」コマンドで更新できます
- データ幅：8 ビット
- ストップ・ビット：1
- パリティなし
- RXD および TXD に使用されるピンは、BSL の構成から取得されます

4.2.2.2 I2C

BSL の I2C インターフェイスは、I2C ターゲットとして動作できます。ホストはコントローラとして動作し、通信を行います。

- I2C0 が使用されます
- I2C ターゲット・アドレスはデフォルトで 0x48 です。BSL の構成で設定できます
- SCL および SDA ラインには外部プルアップが必要です
- SDA および SCL に使用されるピンは、BSL の構成から取得されます

注

- BSL は、UART および I2C インターフェイス用にメイン以外で構成されているピンの詳細を確認しません。BSL は、ピン構成が正しいと想定します。
- UART と I2C の両方で同じピンを使用してはいけません。

4.2.2.3 CRC

CRC は、ハードウェア CRC モジュールを使用して計算されます。CRC32 をサポートするデバイスでは、32 ビットの CRC 値が使用されます。CRC16 のみをサポートするデバイスでは、16 ビットの CRC が使用され、残りのバイトには 0 が追加されて 4 バイトが構成されます。サポートされている CRC 多項式を特定するには、デバイス固有のデータシートを参照してください。

データの CRC は次のように計算されます。

- CRC32-ISO3309 (または) CRC16-CCITT 多項式
- ビット反転の構成
- 初期シード - 0xFFFFFFFF

4.3 ブートローダのコア コマンド

BSL コマンド	保護	CMD	開始アドレス	データ (バイト)	BSL コアの応答
CMD 接続	なし	0x12	-	-	なし
CMD ブートローダーのロック解除	なし	0x21	-	D1…D32 (パスワード)	あり
CMD フラッシュの範囲消去	あり	0x23	A1...A4	A1...A4 (終了アドレス)	あり
CMD 一括消去	あり	0x15	-	-	あり

BSL コマンド	保護	CMD	開始アドレス	データ (バイト)	BSL コアの応答
CMD データのプログラム	あり	0x20	A1...A4	D1...Dn、	あり
CMD データの高速プログラム	あり	0x24	A1...A4	D1...Dn、	なし
CMD メモリの読み戻し	あり	0x29	A1...A4	L1...L4	あり
CMD 工場出荷時のリセット	あり	0x30	-	D1...D16 (パスワード)	あり
CMD デバイス情報の取得	なし	0x19	-	-	あり
CMD スタンドアロン検証	あり	0x26	A1...A4	L1...L4	あり
CMD アプリケーションの開始	なし	0x40	-	-	なし
CMD ボーレートの変更	なし	0x52	-	D1 (ボーレート ID)	なし

略語:

A1...A4

アドレス バイト。A1 が最下位バイトです

D1...Dn

データ バイト。「n」は BSL の最大バッファ サイズによって制限されます

L1...L4

データ長バイト。C1 は最下位バイトです

4.3.1 接続

構造

ヘッダー	長さ		CMD	CRC32			
0x80	0x01	0x00	0x12	C1	C2	C3	C4

説明

接続コマンドは、特定のインターフェイス (UART または I2C) を経由してホストとターゲットとの間の接続を確立するために使用される、最初のコマンドです。

保護

なし

コマンドの戻り値

BSL アクノリッジのみ

例

ホスト:80 01 00 12 3A 61 44 DE

BSL:00

4.3.2 デバイス情報の取得

構造

ヘッダー	長さ		CMD	CRC32			
0x80	0x01	0x00	0x19	C1	C2	C3	C4

説明

このコマンドは、バージョン情報と、データ・トランザクションで使用可能なバッファ・サイズを取得するために使用されます。

保護

なし

コマンドの戻り値

BSL アクノリッジと、デバイス情報を含む BSL コアの応答。詳細については、「デバイス情報」を参照してください。

例

ホスト:80 01 00 19 B2 B8 96 49

BSL:00 08 19 00 31 00 01 00 01 00 00 00 00 01 00 C0 06 60 01 00 20 01 00 00 00 01 00 00 00 49 61 57 8C

4.3.3 ブートローダのロック解除

構造

ヘッダー	長さ		CMD	データ	CRC32			
0x80	0x21	0x00	0x21	D1...D32	C1	C2	C3	C4

説明

このコマンドは、ブートローダのロック解除に使用されます。ブートローダのロック解除後にのみ、セクション 4.3 に記載されている、保護されているすべてのコマンドが BSL によって処理されます。

ホストが誤ったパスワードを送信した場合、BSL はエラー応答を送信し、デバイスは 2 秒間スリープ・モードに移行します。この 2 秒間のウィンドウでは、いかなるコマンドも受信または処理されません。2 秒後に、ホストは次のパスワードを送信できます。誤ったパスワードが 3 回送信された場合、セキュリティ・アラートのアクションが実行されます。セキュリティ・アラートのアクションは、BSL の構成で設定できます。詳細については、セクション 4.5.1.1 を参照してください。

保護

なし

データ

32 バイトの BSL パスワードが、BSL の構成メモリに保存されます。新しいデバイスのデフォルト値はすべて 0xFF です。

コマンドの戻り値

BSL アクノリッジ、および動作ステータスについてのメッセージを含む BSL コア応答。詳細については、「BSL コア・メッセージ」セクションを参照してください。

例

ホスト:80 21 00 21 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 02 AA F0 3D

BSL:00 08 02 00 3B 00 38 02 94 82

4.3.4 データのプログラム

構造

ヘッダー	長さ		CMD	アドレス	データ	CRC32			
0x80	L1	L2	0x20	A1...A4	D1...Dn	C1	C2	C3	C4

説明

プログラム・コマンドは、A1...A4 から始まるメモリ・アドレスに、D1 から Dn までのデータを書き込みます。このコマンドは、ブロッキング書き込みを行います。プログラミングが完了すると、メッセージ応答がホストに送信されます。

メイン・フラッシュ (アプリケーション・メモリ)、メイン以外のフラッシュ (構成メモリ)、SRAM メモリをプログラミングできます。絶対アドレス範囲の詳細については、デバイス固有のデータシートを参照してください。

プログラムする前に、ホストからフラッシュ・メモリを消去します。メイン・フラッシュ領域を消去する方法の詳細については、「フラッシュの範囲消去」、「一括消去」を参照してください。メイン以外のフラッシュは、工場出荷時リセット・コマンドでのみ消去できます。

フラッシュ・コントローラの特性の関係で、フラッシュをプログラムするとき、データの開始アドレスと長さは 8 バイト単位にする必要があります。

注

ホストから SRAM メモリへのアクセスには制限があります。詳細については、セクション 3.3.1 を参照してください。

保護

あり

アドレス

プログラムするメモリ領域の開始アドレス。A1...A4 で、A1 は 32 ビット・アドレスの最下位バイトです。

データ

指定されたアドレスに書き込まれるデータ・バイト。送信可能なデータの最大サイズは、デバイスのバッファ・サイズによって制限されます。バッファ・サイズは、デバイス情報の取得コマンドで確認できます。

コマンドの戻り値

BSL アクノリッジ、および動作ステータスについてのメッセージを含む BSL コアの応答。詳細については、「セクション 4.4.1」セクション を参照してください。

例

ホスト:80 0D 00 20 00 00 00 00 00 00 00 04 00 00 00 08 7A DC AE B8

BSL:00 08 02 00 3B 00 38 02 94 82

4.3.5 データの高速プログラム

構造

ヘッダー	長さ		CMD	アドレス	データ	CRC32			
0x80	L1	L2	0x24	A1...A4	D1...Dn	C1	C2	C3	C4

説明

データの高速プログラム・コマンドは、データのプログラム・コマンドと同じですが、このコマンドはプログラミング処理を高速にするため、ノンブロッキング書き込みを行います。BSL は、このコマンドに対して、プログラミングが成功したかどうかを示す BSL コア・メッセージ応答を送信しません。

保護

あり

アドレス

プログラムするメモリ領域の開始アドレス。A1...A4 で、A1 は 32 ビット・アドレスの最下位バイトです。

データ

指定されたアドレスに書き込まれるデータ・バイト。送信可能なデータの最大サイズは、デバイスのバッファ・サイズによって制限されます。バッファ・サイズは、デバイス情報の取得コマンドで確認できます。

コマンドの戻り値

BSL アクノリッジ。

例

ホスト:80 0D 00 24 00 01 00 00 01 02 03 04 05 06 07 08 72 10 2A 18

BSL:00

4.3.6 データの読み戻し

構造

ヘッダー	長さ		CMD	アドレス	データ	CRC32			
0x80	0x09	0x00	0x29	A1...A4	L1...L4	C1	C2	C3	C4

説明

このコマンドは、アドレス A1...A4 から始まるデータを読み出すために使用されます。

このコマンドを使用してデータを読み出すには、BSL の構成で読み出しをイネーブルにする必要があります。この機能は、BSL 構成のデフォルトでディセーブルです。

データは、メイン・フラッシュ (アプリケーション・メモリ)、メイン以外のフラッシュ (構成メモリ)、SRAM メモリから読み出すことができます。

注

ホストから SRAM メモリへのアクセスには制限があります。詳細については、「SRAM メモリの使用」を参照してください。

保護

あり

アドレス

読み戻すメモリ領域の開始アドレス。A1...A4 で、A1 は 32 ビット・アドレスの最下位バイトです。

データ

読み取るデータのサイズをバイト単位で、L1...L4 により表します。ここで、L1 は最下位バイトです。読み取るデータの最大サイズは、デバイスのバッファ・サイズによって制限されます。バッファ・サイズは、デバイス情報の取得コマンドで確認できます。

コマンドの戻り値

BSL アクノリッジと、読み戻しコマンドが有効なら要求されたデータを含む BSL コア応答。詳細については、セクション 4.4.3 を参照してください。

読み戻しコマンドのアドレスまたは長さが無効、あるいは読み出しがディセーブルなら、BSL アクノリッジに続いて、メッセージ応答として対応するエラーが送信されます。

例

ホスト:80 09 00 29 00 0C 00 00 08 00 00 00 32 9D B0 35

BSL:00 08 09 00 30 FF FF FF FF FF FF FF FF F6 2B A1 73

4.3.7 フラッシュの範囲消去

構造

ヘッダー	長さ		CMD	アドレス	データ	CRC32			
0x80	0x09	0x00	0x23	A1...A4 (開始アドレス)	A1...A4 (終了アドレス)	C1	C2	C3	C4

説明

フラッシュの範囲消去コマンドは、フラッシュ・メモリの指定した領域を消去するために使用されます。フラッシュはセクタ (1KB) 単位で消去され、それより小さなサイズを消去することはできません。

開始アドレスと終了アドレスが異なるフラッシュ・セクタに存在する場合、BSL は開始アドレスと終了アドレスの間にあるすべてのフラッシュ・セクタを消去します。これは、これらのアドレスを含むセクタも対象になります。

このコマンドは、メイン・フラッシュ・メモリのみを消去できます。メイン以外は消去できません。

終了アドレスを開始アドレスよりも小さくしないでください。

保護

あり

アドレス

消去するメモリ領域の開始アドレス。A1...A4 で、A1 は 32 ビット・アドレスの最下位バイトです。

データ

消去するメモリ領域の終了アドレス。A1...A4 で、A1 は 32 ビット・アドレスの最下位バイトです。

コマンドの戻り値

BSL アクノリッジ、および動作ステータスについてのメッセージを含む BSL コアの応答。詳細については、「セクション 4.4.1」セクション を参照してください。

例

ホスト:80 09 00 23 00 01 00 00 FF 03 00 00 2B E6 BE D8

BSL:00 08 02 00 3B 00 38 02 94 82

4.3.8 一括消去

構造

ヘッダー	長さ		CMD	CRC32			
0x80	0x01	0x00	0x15	C1	C2	C3	C4

説明

一括消去コマンドは、デバイスで使用可能なメイン フラッシュ メモリ (アプリケーション メモリ) 全体を消去します。

BCR の構成メモリの一括消去構成は、この BSL コマンドに影響しません。

BCR の構成メモリでフラッシュ領域が静的書き込み保護されている場合、その領域は消去できません。

保護

あり

コマンドの戻り値

BSL アクノリッジ、および動作ステータスについてのメッセージを含む BSL コアの応答。詳細については、セクション 4.4.1 を参照してください。

例

ホスト: 80 01 00 15 99 F4 20 40

BSL：00 08 02 00 3B 00 38 02 94 82

4.3.9 工場出荷時リセット

構造

ヘッダー	長さ		CMD	データ	CRC32			
0x80	L1	L2	0x30	D1...D16	C1	C2	C3	C4

説明

工場出荷時リセット コマンドは、メインのフラッシュ (アプリケーション) メモリとメイン以外のフラッシュ (構成) メモリをすべて消去します。

このコマンドの処理は、BCR の構成メモリの工場出荷時リセット構成によって変わります。

工場出荷時リセットの許可は、次のように決定されます。

- 「イネーブル」なら、パスワードなしで許可
- 「イネーブルでパスワードが必要」なら、パスワードを指定すれば許可
- 「ディセーブル」なら不許可

BCR の構成メモリでフラッシュ領域が静的書き込み保護されている場合は、その領域を消去できません。

保護

あり

データ

16 バイトの工場出荷時リセット パスワードが、BCR の構成メモリに保存されます。デフォルトのパスワードは、全て 0xFF です。パスワードは、BCR の構成で工場出荷時リセットが「イネーブルでパスワードが必要」になっている場合のみ必要です。

コマンドの戻り値

BSL アクノリッジ、および動作ステータスについてのメッセージを含む BSL コアの応答。詳細については、セクション 4.4.1 を参照してください。

> **注意**
>
> 工場出荷時リセットを行った後で、メイン以外の構成が復元されるまで、システムは潜在的なロックアウト状況に対して非常に脆弱です。これが発生すると、デバイスに再度アクセスできなくなります。

例

シナリオ 1：パスワードを使用しない工場出荷時リセット

ホスト：80 01 00 30 DE 20 24 0B

BSL：00 08 02 00 3B 00 38 02 94 82

シナリオ 2：パスワードを使用した工場出荷時リセット

ホスト： 80 11 00 30 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 8A 28 EA DC

BSL： 00 08 02 00 3b 00 38 02 94 82

4.3.10 スタンドアロン検証

構造

ヘッダー	長さ		CMD	アドレス	データ	CRC32			
0x80	0x09	0x00	0x26	A1...A4	D1...D4	C1	C2	C3	C4

説明

このコマンドは、指定されたメモリ範囲に格納されているデータの CRC を検証するために使用されます。これにより、プログラムされたデータをより迅速に検証できます。データ サイズは最小 1kB にする必要があります。

メイン フラッシュ (アプリケーション メモリ)、メイン以外のフラッシュ (構成メモリ)、および SRAM メモリで CRC 検証を行えます。

注

ホストから SRAM メモリへのアクセスには制限があります。詳細については、セクション 3.3.1 を参照してください。

保護

あり

アドレス

検証対象のメモリ領域の開始アドレス。A1...A4 で、A1 は 32 ビット アドレスの最下位バイトです。

データ

検証されるデータのサイズをバイト単位で、L1...L4 により表します。ここで、L1 は最下位バイトです。1KB ≦ サイズ ≦ 64KB。

コマンドの戻り値

BSL アクノリッジと、要求されたメモリ領域について計算された CRC 値を含む BSL コア応答。詳細については、セクション 4.4.5 を参照してください。

検証コマンドのアドレスまたは長さが無効な場合、BSL アクノリッジに続いて、対応するエラーがメッセージ応答として送信されます。セクション 4.4.1 を参照してください。

例

ホスト: 80 09 00 26 00 00 00 00 00 0C 00 00 1C E9 07 E1

BSL: 00 08 05 00 32 A0 45 71 82 91 1F 94 EC

4.3.11 アプリケーションの開始

構造

ヘッダー	長さ		CMD	CRC32			
0x80	0x01	0x00	0x40	C1	C2	C3	C4

説明

アプリケーションの開始コマンドは、システム・リセットを発行します。これによりブートローダは終了し、ブートコードが再実行されて、アプリケーションが開始されます。

保護

なし

コマンドの戻り値

BSL アクノリッジ

例

ホスト:80 01 00 40 E2 51 21 5B

BSL:00

4.3.12 ボーレートの変更

構造

ヘッダー	長さ		CMD	データ	CRC32			
0x80	0x02	0x00	0x52	D1	C1	C2	C3	C4

説明

このコマンドを使用して、UART インターフェイスのボーレートを変更できます。新しいボーレートは、このパケットについて BSL アクノリッジを送信した後に有効になります。

BSL UART のデフォルトのボーレートは 9600bps です。

注

ボーレートを更新した後で、ブートローダーのロック解除コマンドで誤った BSL パスワードが送信されると、ボーレートの設定はデフォルト値に戻ります。それ以後の通信はデフォルトのボーレートで行われます。

保護

なし

データ

表に記載されている D1 ボーレート。

ID	ボーレート (bps)
1	4800
2	9600
3	19200
4	38400
5	57600
6	115200
7	1000000
8	2000000
9	3000000

コマンドの戻り値

BSL アクノリッジ

例

ホスト: 80 02 00 52 03 6C 83 A2 AF

BSL:00

4.4 ブートローダーのコア応答

BSL の応答	RSP	データ
メモリの読み戻し	0x30	D1…Dn
デバイス情報の取得	0x31	D1…D24
スタンドアロン検証	0x32	D1…D4
メッセージ	0x3B	MSG

BSL の応答	RSP	データ
詳細なエラー	0x3A	D1..D3

略語:

MSG

BSL コアからの応答を含む 1 バイトで、要求されたアクションの結果を表しています。エラー コード、または動作が成功したことを示すアクノリッジです。BSL がデータ (メモリ、バージョン、CRC、バッファ サイズなど) を返す必要がある場合、動作が成功したという応答は返されず、BSL コアは直接データを送信します。

D1..Dn

データ バイト。「n」は BSL の最大バッファ サイズによって制限されます

4.4.1 ブートローダーのコア メッセージ

構造

ヘッダー	長さ		RSP	データ	CRC32			
0x08	0x02	0x00	0x3B	MSG	C1	C2	C3	C4

説明

BSL は一部のコマンドについて、処理したコマンドのステータスを示すメッセージ応答をホストに送信します。BSL から返される可能性があるすべてのメッセージを、次の表に示します。

MSG	意味	考えられる原因(1)
0x00	動作の成功	
0x01	BSL ロック中のエラー	BSL は、ブートローダのロック解除パスワード コマンドでロック解除されていない。または、BSL のロック解除後に、コマンド受信フェーズでタイムアウトが発生した。
0x02	BSL のパスワード エラー	ブートローダのロックを解除するために送信されたパスワードが正しくない。
0x03	複数回の BSL パスワード エラー。セキュリティ アラートのアクションが実行される。	ブートローダーのロック解除で、正しくないパスワードが 3 回送信された。
0x04	未知のコマンド	BSL に与えられたコマンドが有効なコマンドとして認識されなかった
0x05	無効なメモリ範囲	指定されたメモリ範囲は無効。
0x06	無効なコマンド	BSL に与えられたコマンドは既知のコマンドであるが、その時点では無効なもので、処理できない。
0x07	工場出荷時リセットが無効	BCR の構成で、工場出荷時のリセットが無効になっている
0x08	工場出荷時リセットのパスワード エラー	BCR の構成で工場出荷時リセットが「有効でパスワードが必要」に設定されているときに、工場出荷時リセットのコマンドでパスワードが送信されていないか、正しくない
0x09	読み出しエラー	メモリの読み出しが、BCR の構成で無効になっている
0x0A	アドレスまたは長さのアライメントが無効	フラッシュへのプログラムで、開始アドレスまたはデータの長さが 8 バイト境界になっていない
0x0B	スタンドアロン検証の長さが無効	スタンドアロン検証用に送信されたデータのサイズが 1KB 未満

(1) この表には、ステータスやエラーが発生した原因として考えられるものが記載されていますが、これ以外の原因がないとは限りません。また、ここに記載されているのはエラーの原因のうち、ソフトウェアに関する、ホストによって訂正可能なものだけです。

4.4.2 詳細なエラー

構造

ヘッダー	長さ		RSP	データ	CRC32			
0x08	0x02	0x00	0x3A	D1..D3	C1	C2	C3	C4

説明

D1 - エラー タイプ

D3、D2 - エラーの詳細

可能な値

エラーのタイプ		エラーの詳細	
値	説明	値	説明
0xF0	フラッシュ エラー	0xXX	FLASHCTL.STATCMD レジスタの値が含まれています

4.4.3 メモリの読み戻し

構造

ヘッダー	長さ		RSP	データ	CRC32			
0x08	L1	L2	0x30	D1...Dn	C1	C2	C3	C4

説明

このコマンドは、読み戻しコマンドへの応答として、要求されたデータを返します

データ

データ D1..DN。「n」は BSL の最大バッファ・サイズによって制限されます。

4.4.4 デバイス情報

構造

ヘッダー	長さ		RSP	データ	CRC32			
0x08	0x19	0x00	0x31	D1...D24	C1	C2	C3	C4

説明

このコマンドは、ID の取得コマンドへの応答として、バージョン情報と BSL バッファ・サイズを返します

データ

ID バイト	データ・バイト
コマンド・インタープリタのバージョン	[D02-D01]
ビルド ID	[D04-D03]
アプリケーションのバージョン	[D08-D05]
アクティブ・プラグイン・インターフェイスのバージョン	[D10-D09]
BSL の最大バッファ・サイズ	[D12-D11]
BSL バッファの開始アドレス	[D16-D13]
BCR の構成 ID	[D20-D17]
BSL の構成 ID	[D24-D21]

アプリケーションのバージョン:

32 ビットのアプリケーション・バージョンは、BSL の構成で指定されているアドレスから取得されます

BSL バッファ・サイズ:

送受信される BSL データ・パケットを格納するために使用できる RAM データ・バッファのサイズ。

例

ホスト:80 01 00 19 B2 B8 96 49

BSL:00 08 19 00 31 00 01 00 01 00 00 00 00 01 00 C0 06 60 01 00 20 01 00 00 00 01 00 00 00 49 61 57 8C

上の応答は次のように解釈されます。

コマンド・インタープリタのバージョン - 0x0100

ビルド ID - 0x0100

アプリケーションのバージョン - 0x00000000

アクティブ・プラグイン・インターフェイスのバージョン - 0x0001

BSL の最大バッファ・サイズ - 0x06C0

BSL バッファの開始アドレス - 0x20000160

BCR の構成 ID - 0x00000001

BSL の構成 ID - 0x00000001

4.4.5 スタンドアロン検証

構造

ヘッダー	長さ		RSP	データ	CRC32			
0x08	0x05	0x00	0x32	D1...D4	C1	C2	C3	C4

説明

このコマンドは、スタンドアロンの検証コマンドへの応答として CRC 値を返します

データ

要求されたメモリ領域について計算された 32 ビットの CRC 値。D1...D4 で、D1 は CRC32 の最下位バイトです。

4.5 ブートローダのセキュリティ

4.5.1 パスワードで保護されているコマンド

メモリのデータに直接または間接的にアクセスできるすべてのコマンドは、パスワードで保護されています。このパスワードは、メイン以外のメモリの BSL 構成で設定できます。

誤ったパスワードが送信されると、デバイスは次の 2 秒間スリープ状態になり、ブルートフォース攻撃を困難にするため、この期間中はいかなるコマンドも受け付けません。誤ったパスワードが 3 回送信されると、BSL によってセキュリティ・アラート・アクションが実行されます。

4.5.1.1 セキュリティ アラート

セキュリティ アラートは、BSL 構成で、次に示す 3 つのモードのいずれかに構成できます。

1. **工場出荷時リセット** - 工場出荷時リセットは、メインおよびメイン以外のフラッシュ メモリ全体を消去します。この消去は、BCR の構成の工場出荷時リセット モードに関係なく行われます。フラッシュの特定の部分が静的書き込み保護されている場合、BSL はフラッシュ メモリ全体を消去できません。この場合、メイン以外が消去されると、ブートローダーのパスワードはデフォルトに戻ります。その後、BSL のロックを解除し、新しいパスワードを使用してプログラムできます。
2. **ブートローダーのディセーブル** - BCR の構成でブートローダーをディセーブルし、BSL を終了します。BCR のメイン以外の構成を更新してブートローダをイネーブルにしない限り、それ以後は BSL を使用できません。メイン以外について静的書き込み保護がイネーブルされている場合、BSL はディセーブルされません。

   また、この機能は 2KB を超える SRAM メモリを搭載したデバイスでサポートされています。
3. **何もしない** - 何の動作も行いません。

注

「工場出荷時リセット」のセキュリティ アラートが構成されており、メイン以外が消去済み状態のままの場合、デバイスはロックされ、以後デバイスにアクセスできなくなります。

4.5.2 BSL エントリ

ブートローダへのエントリは、ブートコード経由でのみ許可されます。

ブートローダは、BCR の構成で無効にできます。

5 ブートローダによるプログラムのフローのサンプル

このセクションでは、BSL ホストがブートローダでイメージをロードするときの代表的なシーケンスについて説明します。このサンプル・シーケンスは、フラッシュ・メモリを消去し、新しいファームウェアをプログラムします。

- ブートローダは、ブランク・デバイスの検出、ピン・ベースの起動、またはアプリケーション要求によって開始されます。
- 起動すると、目的のインターフェイスを経由して接続コマンドを送信し、BSL との接続を確立します。
- UART インターフェイスを使用する場合は、ボーレートをより高い値に変更して、通信をさらに高速化することができます。これはオプションです。
- フラッシュ・メモリを完全に消去するには、一括消去コマンドを使用します。メイン以外のフラッシュを更新する必要がある場合のみ、工場出荷時リセット・コマンドを使用します。これは、メイン以外のフラッシュが消去され、プログラムされていない状態のままだと、デバイスがロックされるためです。
- ファームウェア・イメージをプログラムします
- プログラムされたメモリ領域の CRC 検証を行い、プログラムされたデータの正確性をチェックします。この手順はオプションです。
- アプリケーションは、「アプリケーションの開始」コマンドで開始できます。

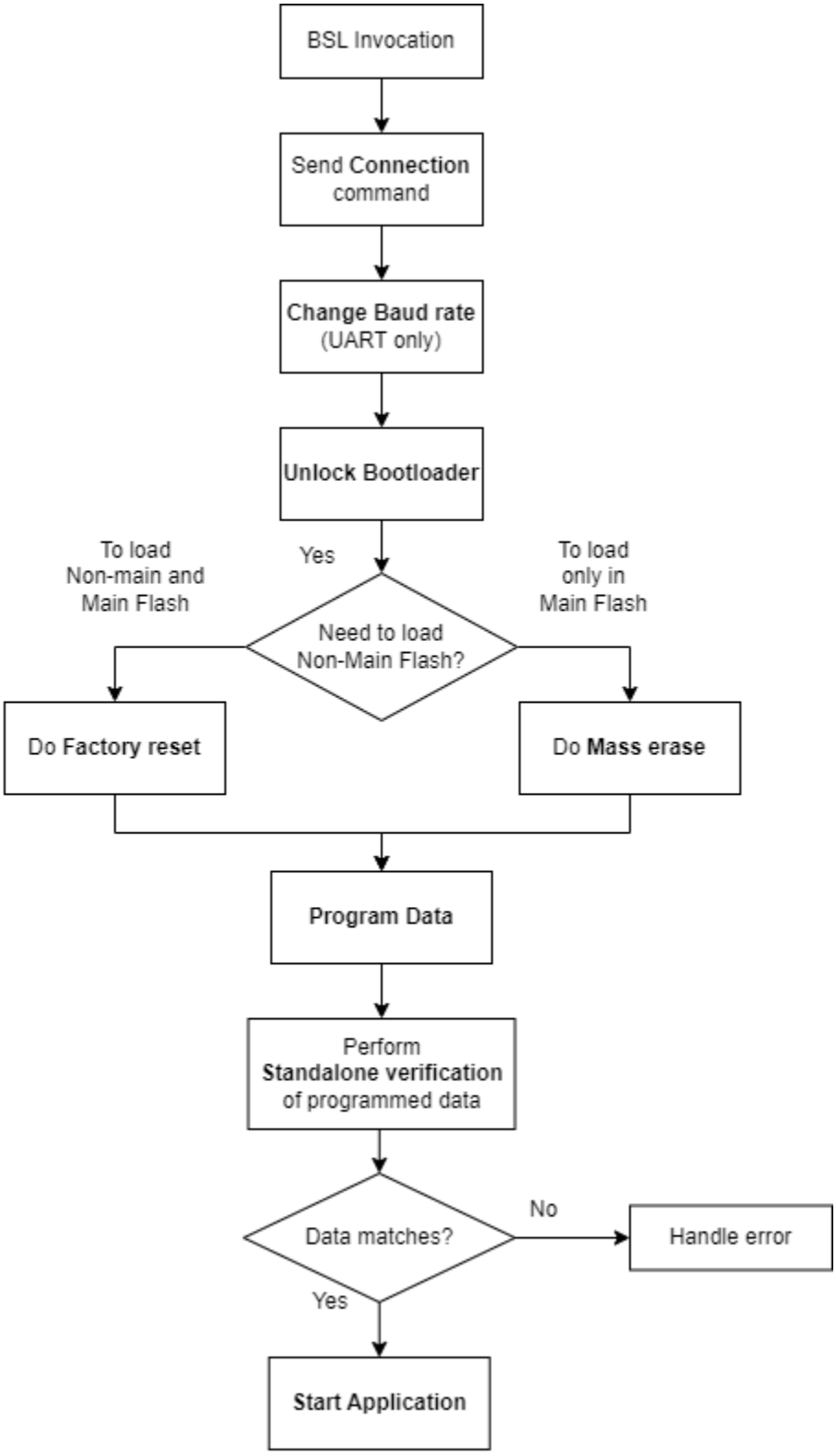

図 5-1. BSL ホストのシーケンス

6 フラッシュ ベースの BSL

フラッシュ ブートローダーは、フラッシュ メモリに配置されているユーザー実装のブートローダー ルーチンです。これにより、カスタマイズと更新が可能になります。フラッシュ BSL の実装は、デバイスの BSL タイプによって異なります。

- ROM BSL がサポートされたタイプ 1 デバイスは、フラッシュ メモリ内にカスタム ブートローダーを実装するオプションを提供します。このオプションを、セカンダリ / 代替ブートローダーと呼びます。このタイプのデバイスでは、ROM BCR が起動を処理し、フラッシュの実装では、デバイスで利用可能ないずれかのシリアル インターフェイスを介して BSL 動作 (プログラム、メモリの読み戻しなど) を処理することが期待されています。
- ROM BSL のサポートがないタイプ 2 デバイスは、セカンダリ BSL と同じ実装を使用します。

タイプ 1 とタイプ 2 のフラッシュ BSL の実装は、どちらも同じ起動フローと構成手順に従います。タイプ 1 および 2 デバイスについては、セクション 6.1 を参照してください。

- タイプ 3 デバイスでは、BSL 動作に加えて BSL 起動処理が必要です。

タイプ 3 デバイスについては、セクション 6.2 を参照してください。

注

フラッシュ BSL メモリが書き込み保護されていない場合、ブートローディング処理によってフラッシュ BSL コードが消去され、デバイスがロックアップする可能性があります。デバッグ構成によっては、リカバリが可能な場合と不可能な場合があります。

6.1 セカンダリ BSL / ROM で起動されるフラッシュ BSL

ROM BCR が BSL 起動条件を検出すると、このフラッシュ ベースの BSL を起動しますが、シリアル インターフェイスを使用して BSL 動作を処理することが想定されています。

これを実現するには、メイン フラッシュ メモリの BSL を 0x0000 以外の任意のアドレスにロードします。アプリケーションはフラッシュ メモリの先頭に配置することが予期されています。そして、それをメイン以外のフラッシュの BSL 構成に登録します。次にいずれかの呼び出し方法で BSL を起動すると、デバイスはフラッシュ BSL 構成フィールドをチェックしてそこに分岐し、制御が ROM に戻ることは予期されません。

BCR の構成の BSL モード構成は、フラッシュ BSL にも適用されます。この設定がディセーブルなら、フラッシュ BSL は起動されません。

ブートローディング処理のとき意図せず消去されないよう、BSL がロードされるフラッシュ メモリ領域は、BCR の構成で書き込み保護する必要があります。メイン以外の書き込み保護はオプションです。ただし、メイン以外のメモリの BSL 構成でフラッシュ BSL ポインタが保持されるように注意する必要があります。

6.1.1 セカンダリ BSL の例

SDK サンプルの一部として、参照用にタイプ 1 デバイスのセカンダリ ブートローダーの例が用意されています。このセクションでは、その例について詳しく説明します。

説明

このサンプルのセカンダリ ブートローダは、メモリ内のデータのプログラミングと検証をサポートしており、デバイスのプライマリ BSL (ROM BSL) と同じ BSL プロトコル フォーマットを持ち、ROM BSL と同じ方法で起動できます。

次に示す主要な機能がサポートされています

- データのプログラム
- フラッシュ メモリの消去
- データの読み戻し
- CRC 検証
- アプリケーションの開始

このブートローダは、UART インターフェイスを使用してホストと通信します。

この例では、セカンダリ ブートローダの実装と登録を行います。したがって、このイメージがデバイスにロードされると、デバイスのプライマリ ブートローダは使用できなくなります。セカンダリ ブートローダのみがアクティブになります。プライマリ ブートローダを使用するようにデバイスを復元するには、SWD_Factory_Reset コマンドを使用する必要があります (デバッグ サブシステム メールボックスを使用して工場出荷時にリセット) .

使用例

- UART_RX と UART_TX を BSL ホスト (UART 搭載の任意のマイクロコントローラ) に接続します。
- サンプルをコンパイルしてロードします。
- BSL の起動ピン、または他の起動方法を使用して、BSL 起動条件を作り出します。
- ホストから**デバイス情報の取得**コマンドを送信します。
- デバイスから、バージョン情報と、使用可能な SRAM バッファ容量が返されます。
- 同様に、消去、プログラム、検証コマンドを送信して、メモリ内にデータをプログラムします。

ソフトウェア ファイルの詳細

ファイル名	詳細
secondary_bsl.c	BSL の動作に必要なペリフェラルを初期化。通信インターフェイスからコマンド パケットを受信し、コマンド処理レイヤに渡す。 また、BSL 構成メモリにセカンダリ ブートローダを登録する。
bsl_ci.c	コマンド パケットを解釈し、コマンドを処理して、応答をホストに返す
bsl_ci.h	BSL のコマンドと応答の定義が記載されている。また、bsl_ci.c の関数宣言も含まれている
bsl_uart.c	ホストと BSL コアとの間の通信を処理する
bsl_uart.h	BSL アクノリッジの定義と、bsl_uart.c の関数宣言が含まれている
ti_msp_dl_config.h	UART ピン、使用されるペリフェラルのベース アドレスなど、デバイス固有の構成が含まれている
boot_config.h	BCR と BSL の構成構造が含まれている
factory_config.c	SRAM メモリ サイズなど、工場出荷時に構成されているデバイス固有のデータを読み取る関数が実装されている。
factory_config.h	工場出荷時の構成構造と、factory_config.c の関数宣言が含まれている
startup_mspm0x_ticlang	スタートアップ ファイルで、ベクタ テーブル、リセット ハンドラ、および他のハンドラが含まれている
mspm0x.cmd	リンカ コマンド ファイルで、セカンダリ ブートローダ イメージを格納するメモリ領域と、そのイメージがどの SRAM 領域で動作するかを指定する。

カスタマイズ

この例では、セカンダリ ブートローダのリファレンス実装を示します。必要に応じて、この実装をカスタマイズできます。カスタマイズは主に、BSL のコア レイヤ (secondary_bsl.c、bsl_ci.c) またはインターフェイス レイヤ (bsl_uart.c) で行われます。

カスタマイズは、次の手順に従います。

- 必要に応じてコードを変更します
- 変更が完了したら、コードをコンパイルします
- BCR 構成で、フラッシュの書き込み保護設定を適切に変更します
- BCR 構成の CRC を計算し、新しい CRC 値を保存します
- コードを再度コンパイルします
- カスタマイズされた BSL イメージをロードします

6.2 スタンドアロン フラッシュ BSL

このフラッシュ BSL の実装では、BSL の動作とともに起動メカニズムを処理することが期待されています。デバイスがリセットから復帰すると、常にフラッシュ メモリの先頭に配置されたコードを実行します。

フラッシュ BSL サポートを追加するには、起動と BSL の実装をフラッシュにロードする必要があります。また、ブートローディング処理のとき意図せず消去されないよう、BSL がロードされるフラッシュ メモリ領域は、BCR の構成で書き込み保護する必要があります。

6.2.1 スタンドアロン フラッシュ BSL の例

MSPM0 SDK には、タイプ 3 デバイス用のフラッシュ BSL 実装のサンプルが付属しています。このセクションでは、それについて詳しく説明します。

説明

フラッシュ BSL のサンプルでは、メモリ内のデータのプログラミングと検証をサポートしています。ホストとの通信に、UART/I2C インターフェイスの 1 つを使用できます。

次の 3 つの方法のいずれかで呼び出すことができます。

- アプリの起動時にアプリケーションなし
- BSL ピンのイノベーション
- アプリケーション ベースの呼び出し

次に示す機能がサポートされています

- データのプログラム
- デバイス情報の取得
- フラッシュ メモリの消去
- データの読み戻し
- CRC 検証
- アプリケーションの開始

フラッシュ BSL 領域は、ブートコードのユーザー構成を使用して静的書き込み保護されています。このイメージがデバイスにロードされると、フラッシュ ブートローダーはアクティブになり、リセットのたびに起動条件を探します。BSL 要求が見つからない場合、アプリケーションに分岐します。その後、フラッシュ BSL の更新が必要な場合や、削除が必要な場合は、SWD_Factory_Reset コマンドを使用してフラッシュ BSL をフラッシュ メモリから消去する必要があります。

ユーザーは、インターフェイス ピンやその他の設定を構成できます。また、メモリ フットプリントを最適化するために必要な機能のみを選択的に有効化することもできます。

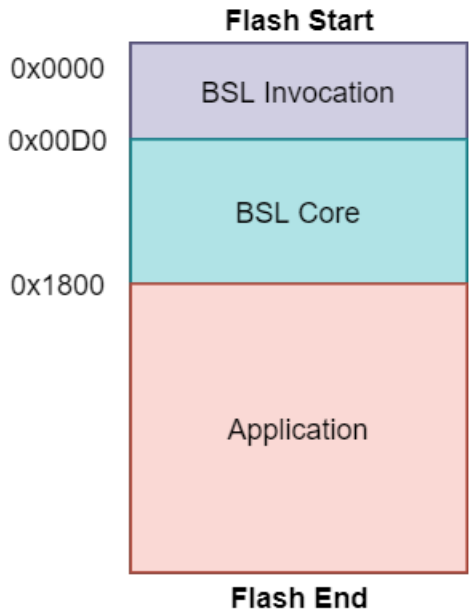

図 6-1. スタンドアロン フラッシュ BSL メモリのレイアウト

フラッシュ メモリの起動ロジックは開始アドレス 0x0000 に配置されています。次に、BSL が配置されています。これら 2 つの組み合わせは、デフォルトで最初の 6KB のフラッシュを占有します。このアプリケーションは、0x1800 から開始することを想定しています。これらのアドレスは、BSL で追加または削除された機能によって異なる場合があります。

使用例

- UART_RX と UART_TX、または I2C_SDA と I2C SCL を BSL ホストに接続します
- **flash_bsl_modules.h** で必要な BSL 機能のみをイネーブルにします
- サンプルをコンパイルしてロードします。
- BSL の起動ピン、または他の起動方法を使用して、BSL 起動条件を作り出します。
- ホストからデバイス情報の取得コマンドを送信します。デバイスから、バージョン情報と、使用可能な SRAM バッファ容量が返されます。
- 同様に、消去、プログラム、検証コマンドを送信して、メモリ内にデータをプログラムします。

ソフトウェア ファイルの詳細

ファイル名	詳細
flashBSL.c	BSL の動作に必要なペリフェラルを初期化。通信インターフェイスからコマンド パケットを受信し、コマンド処理レイヤに渡す。 BSL 起動が要求されると、この動作が実行される。
flashBSL_defines.h	flashBSL.c に関する構成とその他の定義が含まれている
flashBSL_modules.h	フラッシュ BSL のさまざまな機能が含まれており、必要に応じて構成できる
flashBSL_invocation .c	結果に基づいて、さまざまな起動条件と、アプリケーションまたは BSL に分岐する
flashBSL_invocation .h	呼び出しに関連する構成とマクロ定義が含まれている
flashBSL_uart .c	ホストと BSL コアとの間の通信を処理する
flashBSL_uart .h	BSL アクノリッジの定義が含まれている
flashBSL_i2c .c	ホストと BSL コアとの間の通信を処理する
flashBSL_i2c .h	BSL アクノリッジの定義が含まれている
flashBSL_ci.c	コマンド パケットを解釈し、コマンドを処理して、応答をホストに返す
flashBSL_ci.h	BSL のコマンドと応答の定義が記載されている。また、flashBSL_ci.c の関数宣言も含まれている
boot_config .c	ブート構成の詳細が含まれている。さらに、BSL 構成セクションも含まれており、フラッシュ BSL の実装に基づいてカスタマイズできる。 注:SYSCFG ツールから派生したものではありません。
boot_config .h	boot_config .c の定義が含まれている
startup_mspm0c1104_ticlang.c	BSL のスタートアップ ファイル。BSL と起動を 2 つの独立したルーチンとして処理する。
mspm0c1104.cmd	前述のメモリ レイアウトを処理する。

7 インターフェイス プラグイン

ROM ブートローダは、カスタムのインターフェイス実装をフラッシュ プラグインとして ROM BSL コアに追加できます。これによって、BSL コア全体を再実装しなくてもインターフェイスをカスタマイズできます。

この機能を使用して、次の操作を行えます。

- ROM BSL に存在しない新しいインターフェイスを自動検出用のインターフェイス リストに追加できます。例：SPI、CAN など (または)
- ROM インターフェイスの実装 (UART/I2C) をオーバーライドできます

このオプションを使用するには、フラッシュ プラグインのイメージをメイン フラッシュにロードし、メイン以外のフラッシュ メモリの BSL 構成に登録する必要があります。MSPM0xx テクニカル リファレンス マニュアルの「構成メモリ (メイン以外)」セクションを参照してください。

また、プラグインがロードされるフラッシュ メモリ領域と、メイン以外の構成メモリは、フラッシュ プラグインと、メイン以外のメモリに書き込まれている登録が、ブートローディング処理で消去されないよう、BCR の構成で書き込み保護します。

注

メイン フラッシュのフラッシュ プラグイン領域を消去すると、デバイスがロックされる可能性があります。このため、書き込み保護が必要です。

7.1 実装

このプラグインは、データ処理、および BSL ホストとの通信を行う必要があります。このインターフェイスのフラッシュ・プラグインは、次の 4 つの API を使用して ROM BSL コアと結合されます。

- init
- receive
- transmit
- deinit

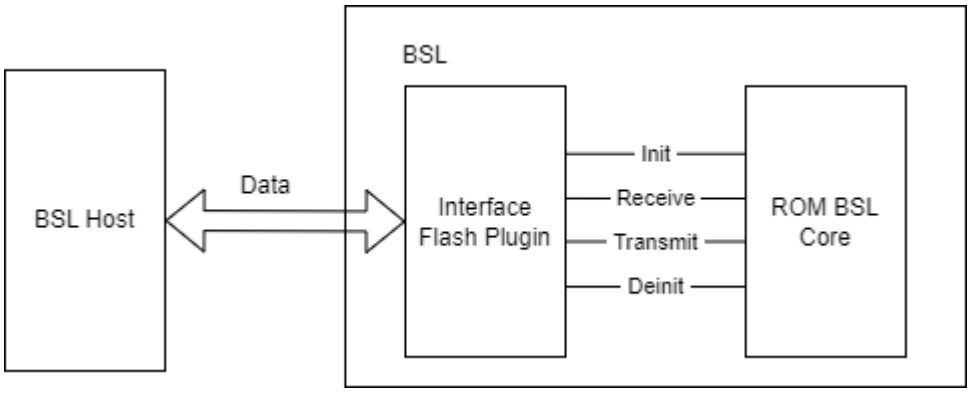

図 7-1. プラグインの実装

フラッシュ・プラグインのイメージは、他のアプリケーションと同様に構築し、メイン・フラッシュにロードする必要があります。ただし、アプリケーションとは異なり、スタートアップ・コードや main 関数は実行されません。上記の 4 つの API のみが、メイン以外のメモリの BSL 構成に登録されているフックを通して ROM BSL によって呼び出されます。

7.1.1 init

プロトタイプ

uint16_t init(uint8_t* buffer, uint16_t bufferSize);

buffer - BSL コアから送信される SRAM データ・バッファへのポインタ。

bufferSize - データ・バッファとして使用できる SRAM メモリ・サイズの 1/2。2 つのバッファを、1 つは送信、もう 1 つは受信に使用する場合、1 つのバッファのアドレスは **buffer** で、もう 1 つのバッファのアドレスは **buffer + bufferSize** です

return - 16 ビットのプラグインのバージョン情報を返します

説明

init 関数は、必要なインターフェイスを構成し、データ処理用のパラメータを初期化します。また、使用される他のグローバル変数も初期化します。これらの変数を初期化するスタートアップ・コードは実行されません。

7.1.2 receive

プロトタイプ

uint32_t receive(void);

return - 受信されたデータ・パケットの 32 ビットの開始アドレスを返します。データ・パケットは、セクション 4 の ROM BSL プロトコルで説明されているものと同じ形式です。

説明

receive 関数は、BSL ホストからのデータ・パケットの読み取りを処理します。パケット全体が受信され、データの正確性がチェックされた場合 (データの CRC 検証) のみ、パケットを BSL コアと共有します。また、アドレスはデータ・パケットごとに 1 回だけ共有されます。この関数が BSL コアによって呼び出された、受信したデータ・パケットがない、または進行中のときは、「0」を返します。

データ・パケットを問題なく正常に受信すると、ROM BSL プラグインと同様に、ホストにアクノリッジします (ROM BSL アクノリッジを参照)。問題が発生した場合、ホストに NACK を通知し、そのパケットは ROM BSL コアと共有されません。

7.1.3 transmit

プロトタイプ

uint8_t send(uint8_t* data, uint16_t length);

data - ホストに送信される BSL コア応答パケットへのポインタ。形式は、セクション 4 で説明したものと同じです。

length - CRC 4 バイトの長さを除く、BSL コア応答パケットの長さ。

return - 送信が成功したなら、「1」を返します。送信が失敗したなら、「0」を返します。

説明

transmit は、BSL コア応答パケットをホストに送信します。パケットの CRC の計算と追加も行います。transmission API は、データの送信が完全に終了した後のみ戻ります。

7.1.4 deinit

プロトタイプ

bool deinit(void);

戻り値

deinit が完了すると、「True」が戻されます。

説明

この関数は、インターフェイス構成をリセットし、割り込みハンドラが登録されている場合は登録解除します。

7.1.5 重要な注意事項

フラッシュ プラグインを開発するときは、次の重要な点に留意してください。

1. フラッシュ プラグインがロードされるメイン フラッシュ メモリ領域は、静的書き込み保護します
2. すべてのグローバル変数は、「init」関数で初期化します
3. 4 つのプラグイン API の関数プロトタイプは、BSL のユーザーガイドの記載に従います

4. SRAM メモリの使用法
   a. VTOR - SRAM の開始 (0x20000000)。割り込みを使用する場合は、ROM BSL がそのアドレス領域を使用するため、SRAM の先頭に VTOR を配置する必要があります
   b. スタック開始アドレス - デバイスで使用可能な SRAM メモリの末尾
   c. スタック サイズ - ROM BSL のスタック サイズを超えないこと
   d. データ セクション (.data、.bss) - 「デバイス情報の取得」コマンドによって返される「BSL バッファの開始アドレス」は、デバイスにフラッシュ プラグインが登録されていない場合、データ セクションの開始アドレスです。
   e. データ セクションのサイズ - データ セクションで消費されるサイズ (.data、.bss) は、BSL のメイン以外の構成メモリに設定する必要があります。

7.2 フラッシュ プラグインのタイプ

フラッシュ プラグインで使用される通信インターフェイスのタイプは、BSL の構成メモリの BSLPLUGINCFG.PLUGINTYPE フィールドで設定されます (テクニカル リファレンス マニュアルを参照)。

プラグインのタイプに ROM インターフェイス タイプ (UART または I2C) が含まれている場合、フラッシュ プラグインは ROM 実装より優先されます。このタイプが、ROM では利用できない新しいタイプ (SPI など) なら、プラグインは新しいインターフェイスとして追加されます。

表 7-1. プラグインのタイプの値

番号	インターフェイス	プラグインのタイプ
1	UART	0x1000
2	I2C	0x2000
3	その他のインターフェイス	0xXXXX

図 7-2 に示すように、フラッシュ プラグインのタイプを確認してから、更新されたインターフェイス リストを使用して自動検出が行われます。フラッシュ プラグインの登録後も、オーバーライドされていない ROM インターフェイスはそのまま使用できます。

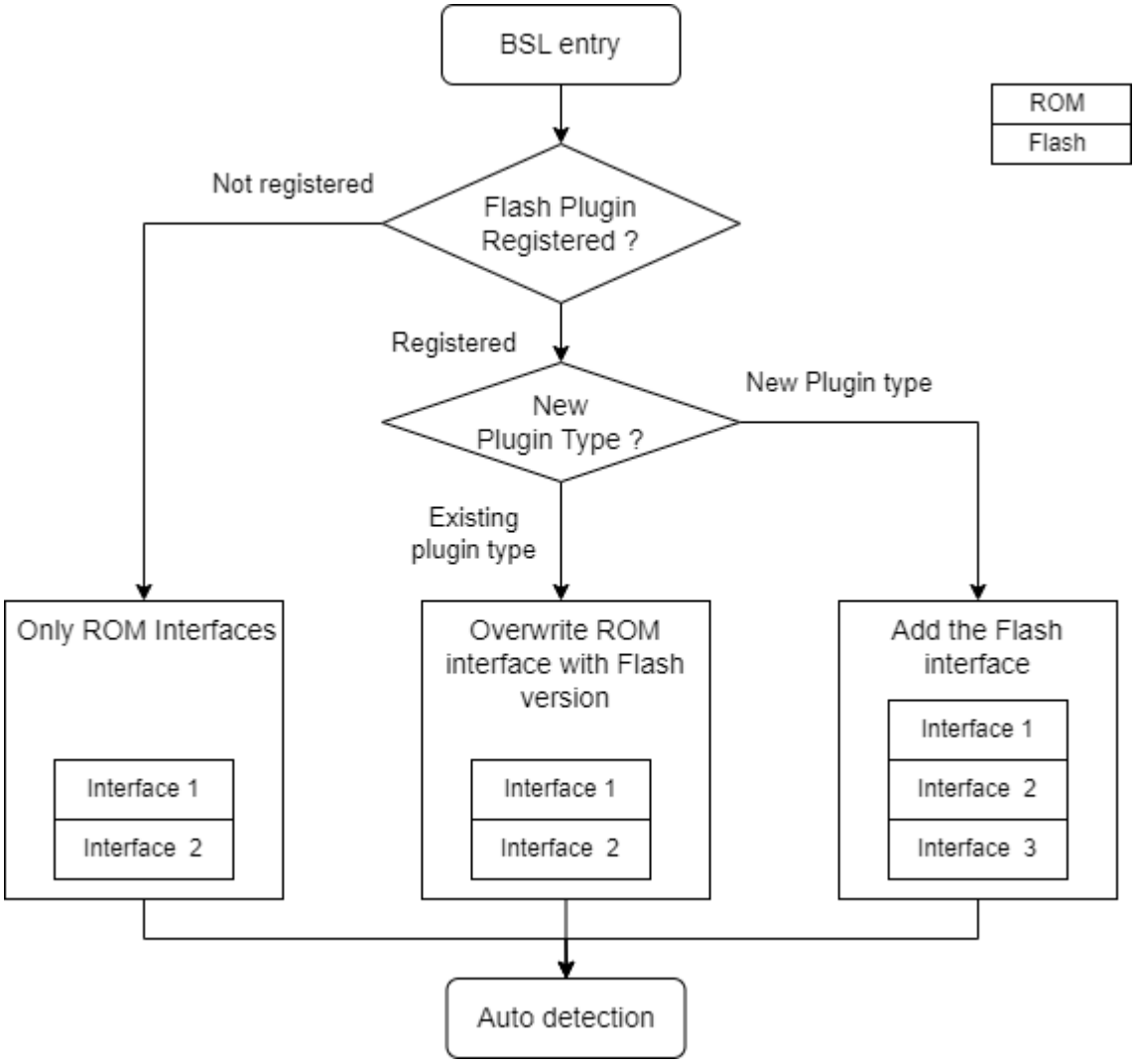

図 7-2. フラッシュ プラグイン

7.3 既存のインターフェイスのオーバーライド

この機能により、既存の ROM BSL インターフェイスである UART と I2C を、フラッシュのバージョンでオーバーライドできます。この場合、フラッシュ・プラグインがロードされると、ROM のバージョンは非アクティブになり、使用できません。ROM のインターフェイスを使用するようにデバイスを復元するには、SWD_Factory_Reset (デバッグ・サブシステム・メールボックスによる工場出荷時リセット) を使用します。

7.3.1 UART インターフェイスのフラッシュ プラグインの例

UART 通信を使用するサンプル フラッシュ プラグインは、SDK サンプルの一部として参照用に提供されています。このセクションでは、その例について詳しく説明します。

説明

UART インターフェイスのフラッシュ プラグインは、次の 4 つの API フックを使用して、BSL ホストと ROM BSL との間のデータ トランザクションを処理します。

- BSL_PI_UART_init
- BSL_PI_UART_receive
- BSL_PI_UART_send
- BSL_PI_UART_deinit

UART フラッシュ プラグインは、必要なときカスタム実装で ROM BSL UART インターフェイスをオーバーライドするために主に使用されます。

使用例

- UART_RX と UART_TX を BSL ホスト (UART インターフェイスを搭載した任意のマイコン) に接続します。
- サンプルをコンパイルしてロードします。
- BSL の起動ピン、または他の起動方法を使用して、BSL 起動条件を作り出します。
- ホストから**接続**コマンドを送信します。成功すると、BSL アクノリッジが受信されます。
- ホストから**デバイス情報の取得**コマンドを送信します。
- BSL は応答として、UART インターフェイスのフラッシュ プラグインのバージョン情報を返します。
- 同様に、消去、プログラム、検証コマンドを送信して、メモリ内にデータをプログラムします。

ソフトウェア ファイルの詳細

ファイル名	詳細
bsl_uart.c	ホストと BSL コアとの間の通信を処理する。init、receive、send、deinit の 4 つのインターフェイス API を定義する。
bsl_uart.h	BSL アクノリッジの定義と、bsl_uart.c の関数宣言が含まれている
ti_msp_dl_config.h	UART ピン、クロック構成などのデバイス固有の構成が含まれている。
boot_config.h	BCR と BSL の構成構造が含まれている
startup_mspm0x_ticlang	デフォルトのハンドラ関数定義のみを含むスタートアップ ファイル。標準的なスタートアップ ファイルとは異なり、割り込みベクタ テーブルやリセット ハンドラはありません。これらの機能はフラッシュ プラグインでは使用されないため、メモリ消費量を減らすため削除されている。
mspm0x.cmd	リンカ コマンド ファイルで、フラッシュ プラグインのイメージを格納するメモリ領域と、そのイメージがどの SRAM 領域で動作するかを指定する。

カスタマイズ

この例では、フラッシュ プラグインのリファレンス実装を示します。この実装は、必要に応じてカスタマイズできます。主に変更されるのは、インターフェイスのフラッシュ プラグインの API です。

カスタマイズは、次の手順に従います。

- 必要に応じてフラッシュ プラグインの API を変更します
- 変更が完了したら、コードをコンパイルします
- API ポインタが更新されたなら、BSL の構成の CRC を計算します
- BCR 構成で、フラッシュの書き込み保護設定を適切に変更します
- BCR 構成の CRC を計算します
- 新しい CRC を BCR および BSL の構成に保存します。
- コードを再度コンパイルします。
- フラッシュ プラグインのイメージをロードします

8 改訂履歴

資料番号末尾の英字は改訂を表しています。その改訂履歴は英語版に準じています。

Changes from FEBRUARY 28, 2023 to JANUARY 31, 2026 (from Revision * (February 2023) to Revision A (January 2026)) **Page**

- MSPM0 デバイス全体にさまざまな BSL タイプを追加。 26

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