AN5275

アプリケーション・ノート

STM32MP1 シリーズのブートローダで使用される USB DFU/USART プロトコル

概要

このアプリケーション・ノートでは、STM32MP1 シリーズ・マイクロプロセッサ用のブートローダ・プログラミング・ツールで使用されるプロトコルについて説明します。組込みソフトウェアでサポートされている各 USB DFU または USART コマンドと、STM32CubeProgrammer ツールで求められるシーケンスの詳細を示します。

1 組込みプログラミング·サービス

1.1 概要

本書は STM32MP1 シリーズの Arm®ベースのマイクロプロセッサに適用されます。

注 Arm は、米国内およびその他の地域にある Arm Limited（またはその子会社）の登録商標です。

ここでは、プログラミング·サービスを提供するために STM32MP1 シリーズのブートローダによって使用されるプロトコル、つまり STM32CubeProgrammer に必要な組込み部品を定義します。

このサービスは、不揮発性メモリ（NVM）、つまり外部 Flash メモリ·デバイスまたはオンチップ不揮発性メモリ（OTP）をプログラムする方法を備えています。

このドキュメントでは、STM32MP1 組込みソフトウェアと STM32CubeProgrammer の両方で使用される USB または USART プロトコルと、想定されるシーケンスについて説明します。

1.2 参照

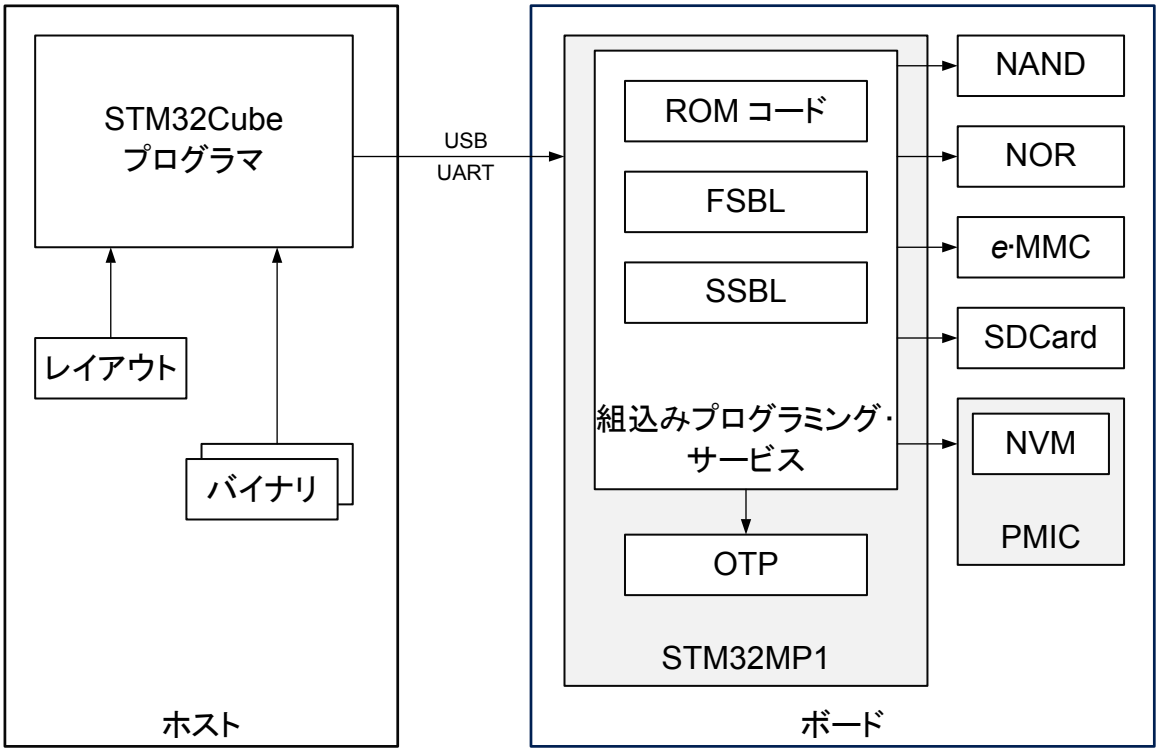
[1]	AN3155、アプリケーション·ノート、USART protocol used in the STM32 bootloader（STM32 ブートローダで使用される USART プロトコル）
[2]	AN3156、アプリケーション·ノート、USB DFU protocol used in the STM32 bootloader（STM32 ブートローダで使用される USB DFU プロトコル）
[3]	UM0412、ユーザ·マニュアル、Getting started with DfuSe USB device firmware upgrade STMicroelectronics extension（DfuSe USB デバイス·ファームウェアのアップグレード STMicroelectronics 拡張の開始）
[4]	UM0424、ユーザマニュアル、STM32 USB-FS-Device development kit（STM32 USB-FS-Device 開発キット）
[5]	USB、デバイス·ファームウェア·アップグレード用のデバイス·クラス仕様、バージョン 1.1 https://usb.org/(1)
[6]	GUID パーティション·テーブル（GPT）は、パーティション·テーブルをレイアウトするための標準で、Unified Extensible Firmware Interface（UEFI）標準の一部です（https://uefi.org/）(1)
[7]	署名ツール wiki.st.com/stm32mpu/wiki/Signing_tool
[8]	ROM コードの概要 wiki.st.com/stm32mpu/wiki/Category:ROM_code
[9]	FlashLayout wiki.st.com/stm32mpu/wiki/STM32CubeProgrammer_flashlayout
[10]	OTPwiki.st.com/stm32mpu/wiki/STM32CubeProgrammer_OTP_management
[11]	STPMIC1 NVMwiki.st.com/stm32mpu/wiki/STM32CubeProgrammer_STPMIC1_NVM_management
[12]	バイナリファイルの STM32 ヘッダ wiki.st.com/stm32mpu/wiki/STM32_header_for_binary_files
[13]	ROM コードのシリアル·ブートの概要の章 wiki.st.com/stm32mpu/wiki/Category:ROM_code
[14]	TF-A で使用されるファームウェア·イメージ·パッケージ wiki.st.com/stm32mpu/wiki/TF-A_overview#FIP

1. この URL は 3rd パーティに属します。ドキュメント発行時に有効です。ただし、STMicroelectronics は、URL または参照されている資料の変更、移動、または無効化に対して責任を負わないものとします。

1.3 概要

プログラミング操作では、ホスト·コンピュータに最初に格納されたバイナリを、インタフェースを介してプラットフォーム上の不揮発性メモリ（NVM）に書き込みます。このプロセスには、組込みプログラミング·サービスと通信する STM32CubeProgrammer ツールが含まれます。これは、ROM コード、第 1 ステージ·ブートローダ (FSBL)、および第 2 ステージ·ブートローダ (SSBL) で構成されています。

図 1. プログラミング動作



通信プロトコルはシリアル·インタフェース（USART、USB）ごとに定義され、ここには既存の STM32 マイクロコントローラ·デバイスにできる限り対応した一連のコマンドおよびいくつかのシーケンスが含まれます（[1] と [2] を参照）。

考えられる NVM は次のとおりです。

- 外部 Flash メモリ·デバイス:
  - NAND Flash メモリ
  - e•MMC
  - SD カード
  - NOR Flash メモリ
- オンチップ不揮発性メモリ:
  - STM32MP1 OTP
  - PMIC の NVM（STPMIC1 など）

レイアウト·ファイルには、プログラムするバイナリのタイプ（バイナリやファイル·システムなど）のリスト、ターゲット NVM、また NVM 内での位置が記載されています。

STM32 ヘッダを持つバイナリ·ファイルの最終的な署名手順は、SigningTool を使用して事前に実行されます（[7] を参照）。

組込みプログラミング·サービスは ROM コードで、FSBL = Arm Trusted Firmware（TF-A）、SSBL = U-Boot です。同じプロトコルが、RAM への FSBL および SSBL のダウンロードと、デバイス NVM へのプログラム対象パーティションのロードの両方で使用されます。

ROM コードは STM32MP デバイスに組み込まれています。その主なタスクは、使用可能なシリアル·ペリフェラルの 1 つを通じて内部 RAM にある第 1 ステージ·ブートローダ (FSBL) をロード、検証、および実行することです。次に、FSBL によりいくつかの初期化（クロックおよび DDR）が行われた後、2 番目のローダ（SSBL）が DDR にロードされ、署名が確認されて実行されます。

1.4 レイアウト·ファイル·フォーマット

Flash メモリ·レイアウトファイルは、デバイスに送信するバイナリまたはパーティションごとに 1 つの行を持つタブ区切り値（tsv）のテキスト·ファイルです。詳しい説明は、ST wiki [9] を参照してください。

1.5 フェーズ ID

フェーズ ID の詳しい説明は、ST wiki [9] を参照してください。0x0 および 0xF1 ~ 0xFD の ID は予約済みです。その他の値については、STM32CubeProgrammer に対して行われたダウンロード·フェーズリクエストごとにレイアウト·ファイル内に存在する固有の識別子です。このリクエストは、ROM コード、FSBL=TF-A、または SSBL = U-Boot によって実行されます。

これは、組込みプログラミング·サービスによって次のパーティションを識別するために使用されます。たとえば、Get フェーズ·コマンドの応答でダウンロードされる TF-A および U-Boot です。

表 1. フェーズ ID

エンコード	パーティション
0x00	レイアウト·ファイル
0X01 から 0x0F	FIP ヘッダ [14] または STM32 ヘッダ付きのブート·パーティション[12]:SSBL、FSBL、その他（信頼できる実行環境 -TEE、Cortex-M4 ファームウェア）
0x01（予約済み）	FSBL を含むイメージ:ROM コードによって使用されます
0x03（予約済み）	FSBL=TF-A で使用される FIP イメージ（組込み SSBL、OP-TEE など）
0x10 から 0xF0	ヘッダなしでプログラムされたユーザ·パーティション（uimage、dtb、rootfs、userfs など）
0xF1 から 0xFD	内部用に予約済み
0xF1	GetPhase コマンド
0xF2	OTP
0xF3	予約済みです。
0xF4	PMIC NVM（オプション）
0xFE	操作終了
0xFF	リセット

0x01 および 0x03 の ID は、FSBL=TF-A または SSBL=U-Boot/FIP を含むイメージ用に予約されています。FSBL は ROM コードによって RAM にロードされ、SSBL は FSBL によって RAM にロードされます。

0x0 および 0xF1 から 0xFD までの ID は内部用に予約されています。

OTP（0xF2）および PMIC NVM（0xF4）の特殊動作の詳しい説明は、ST wiki [10] および [11]を参照してください。

1.6 STM32 イメージ·ヘッダ

STM32 イメージ·ヘッダの詳細については、STM32 MPU wiki [12] を参照してください。

このヘッダは、ROM コードによってロードされる FSBL パーティション = TF-A で使用されます。

- STM32MP15x デバイスの場合はバージョン 1（0x100 バイトのサイズ = 256、符号付きまたはなし）
- STM32MP13x デバイスの場合はバージョン 2（0x200 バイトのサイズ = 512、符号付きまたはなし）

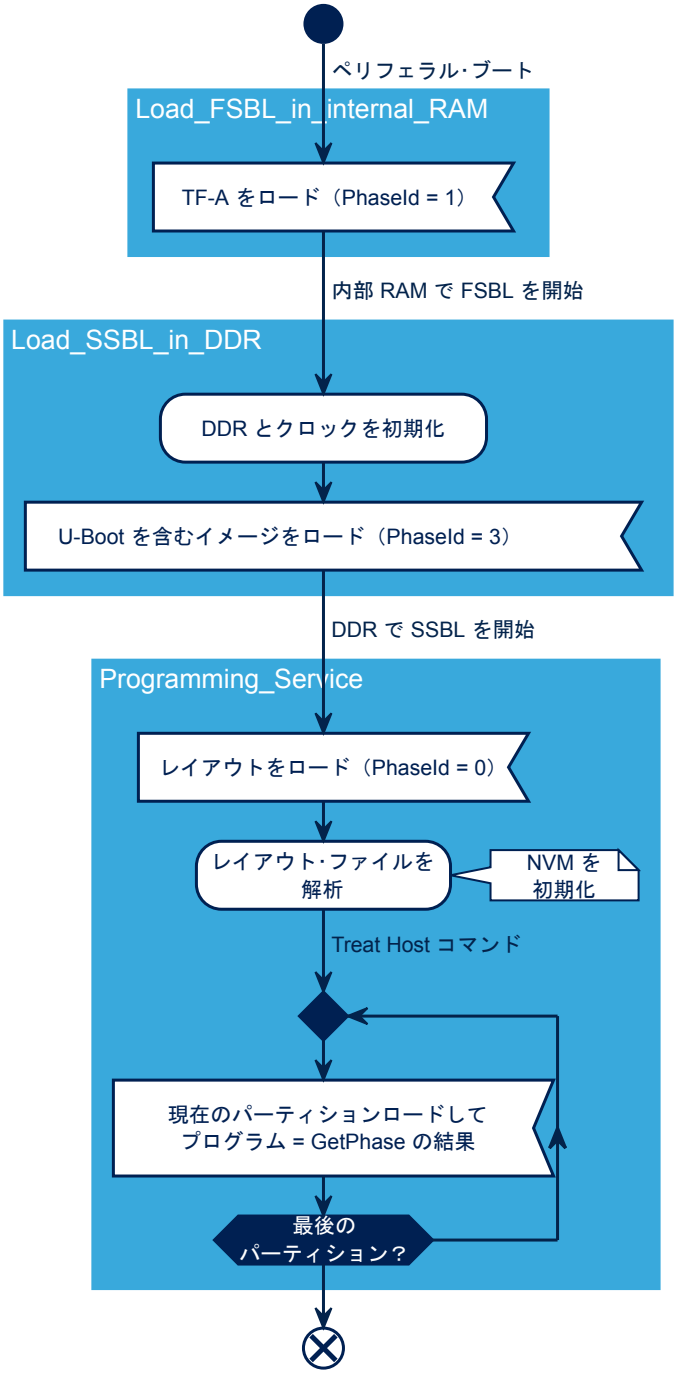
注 このヘッダは、SSBL をロードするために STM32MP15x シリーズの TF-A で使用することもできますが、FIP ヘッダの使用を推奨します。
署名は、閉じたデバイスでのみ必須です。署名プロセスの詳細については、署名ツール [7] を参照してください。閉じたデバイスの定義については、[8] を参照してください。

1.7 プログラミング·シーケンス

1.7.1 ケース 1 – リセットからのプログラミング

STM32CubeProgrammer のこの通常のユースケースでは、ブート·ピンの電圧レベルに基づいて、ブートに使用されるペリフェラルが決定されます（USB または UART、[13]を参照）。ブートローダは STM32CubeProgrammer によって RAM にロードされ、SSBL U-Boot によって DDR で動作するプログラミングサービスが提供されます。

図 2. プログラミング·チャート



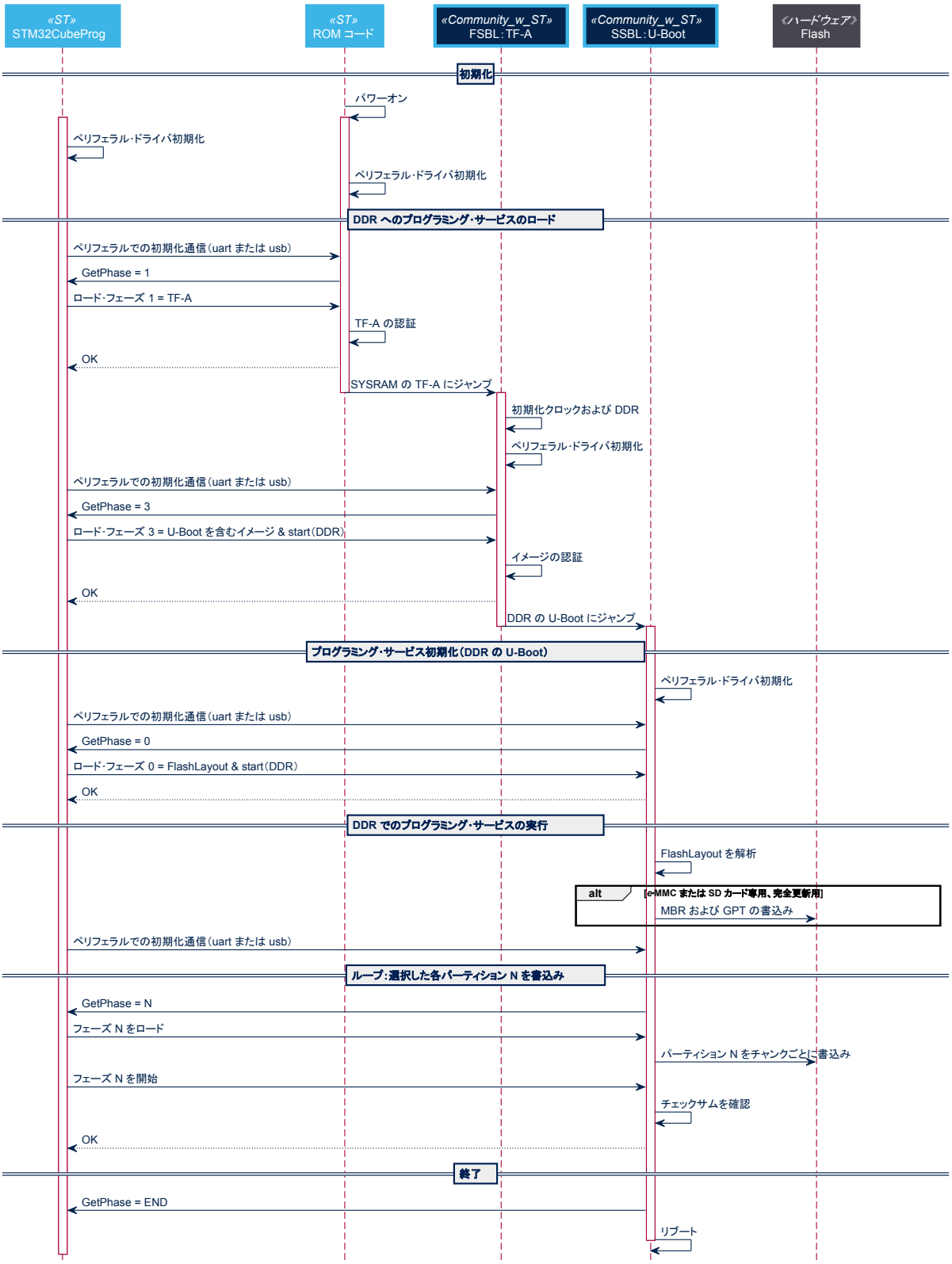
フル·アップデートの場合、同じブート·ステージ·パーティション（TF-A および U-Boot を含むイメージ）が 2 回リクエストされます。1 回は RAM でのロードと実行用、1 回は NVM 更新用です。ただし、提供されるバイナリが同じであるとは限りません。

U-Boot では、UART と USB で受信されるデータ·チャンクは DDR でバッファされます。このバッファのサイズはデバイスによって異なります。

バッファはいっぱいになると、NVM にフラッシュされます。そのため、転送された各チャンクの ACK は、この NVM 更新が実行されるまで遅延されます。

下の図に、標準的なプログラミング·シーケンスの概要を示します。

図 3. プログラミング·シーケンスの概要



1.7.2 ケース 2 – プログラムされたデバイスのための U-Boot からのプログラミング

このコード更新のユースケースでは、NVM は有効なブート·チェーンですでにプログラムされ、ブート·ピンで選択されています（[13] を参照）。

予期されるシーケンスは次のとおりです。

1. ユーザがボードのスイッチをオンにします。
2. ブートローダが NVM からロードされ、RAM で実行されます。
3. ユーザがプラットフォームのブートを中断し、プログラマ·モード（UART または USB）で U-Boot を開始します。これには、次のような方法があります。
   – プログラミングモードに入るためのキー押下（STMicroelectronics ボード上のユーザ·ボタンなど）を検出
   – ユーザが U-Boot コンソールでダウンロード·コマンドを起動（コマンド stm32prog）
   – Linux からリブート·モード経由でプログラマ·モードをリクエスト
4. ユーザが PC で STM32CubeProgrammer を起動します。

その後、前のケースと同様に プログラミング·サービスが DDR で実行されます。

2 UART/USART

2.1 UART/USART プロトコル

UART/USART プロトコルは、[1] で説明されている STM32 マイクロコントローラの動作にできる限り従います。下の図では、違いを強調し、想定されるシーケンスの詳細を示します。

図 4. USART プログラミング·シーケンスの説明 - 1

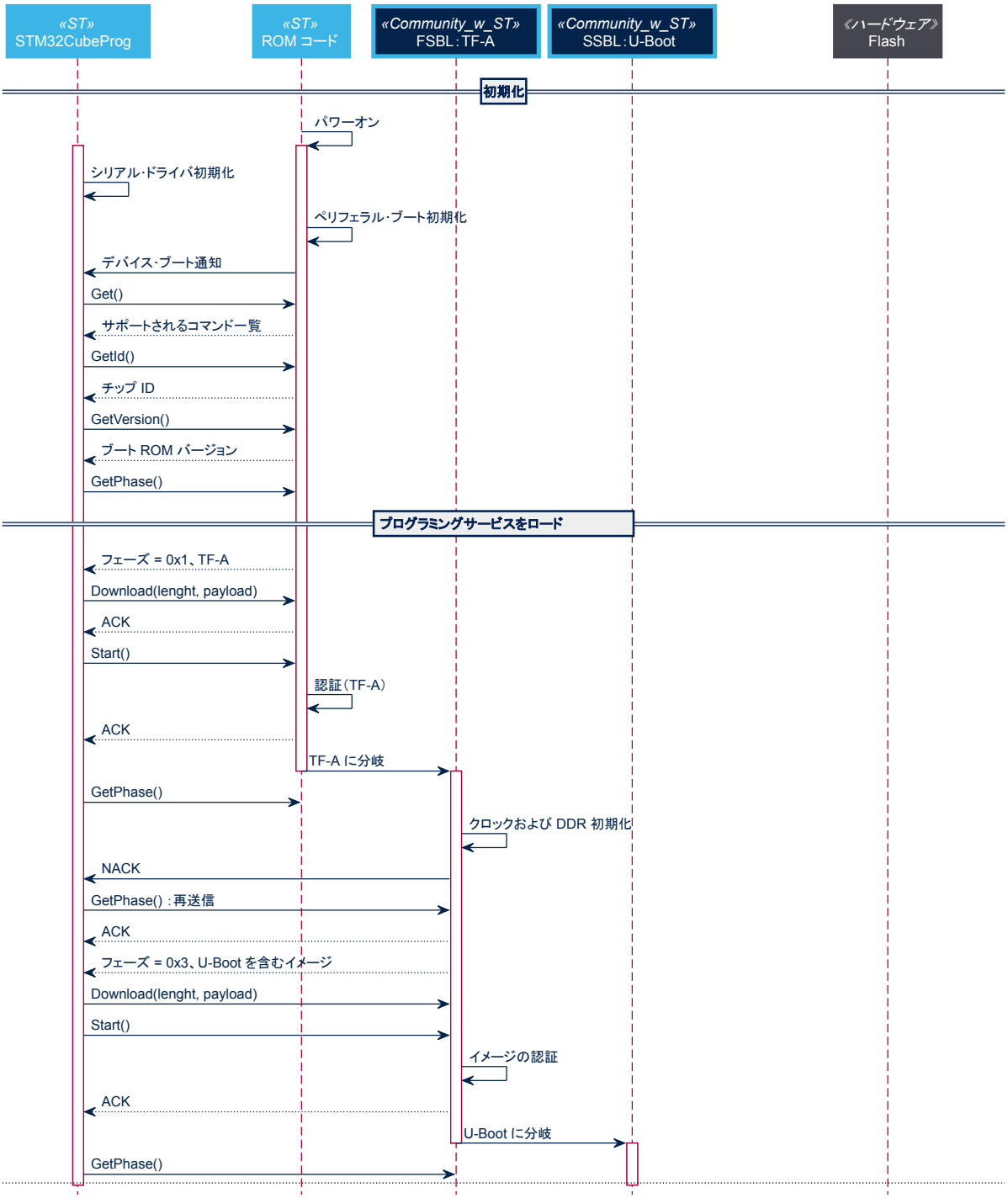
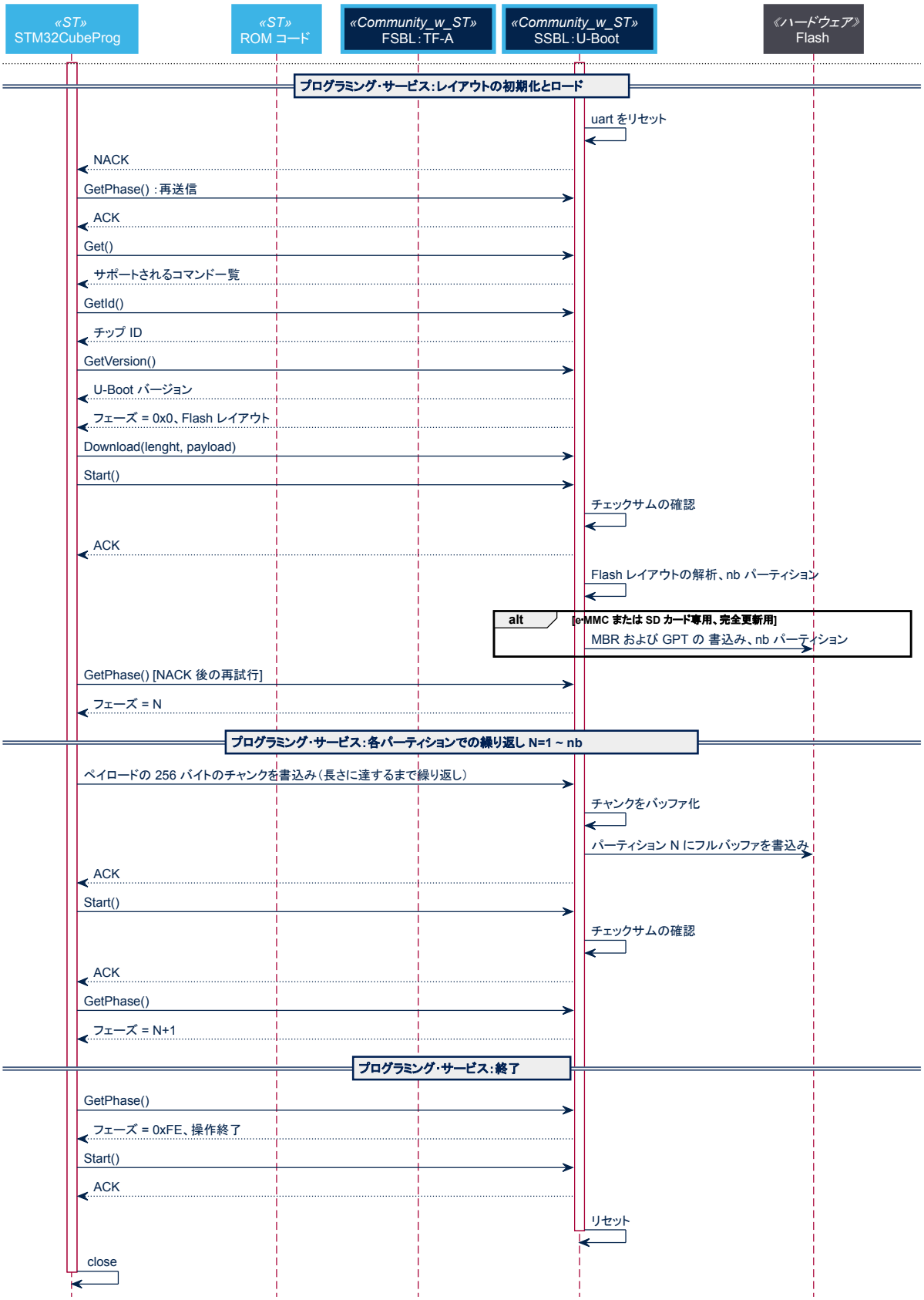


図 5. USART プログラミング・シーケンスの説明 - 2



2.2 UART/USART 設定

設定は、次の設定項目により、ROM コード（[13]）でセットされます。

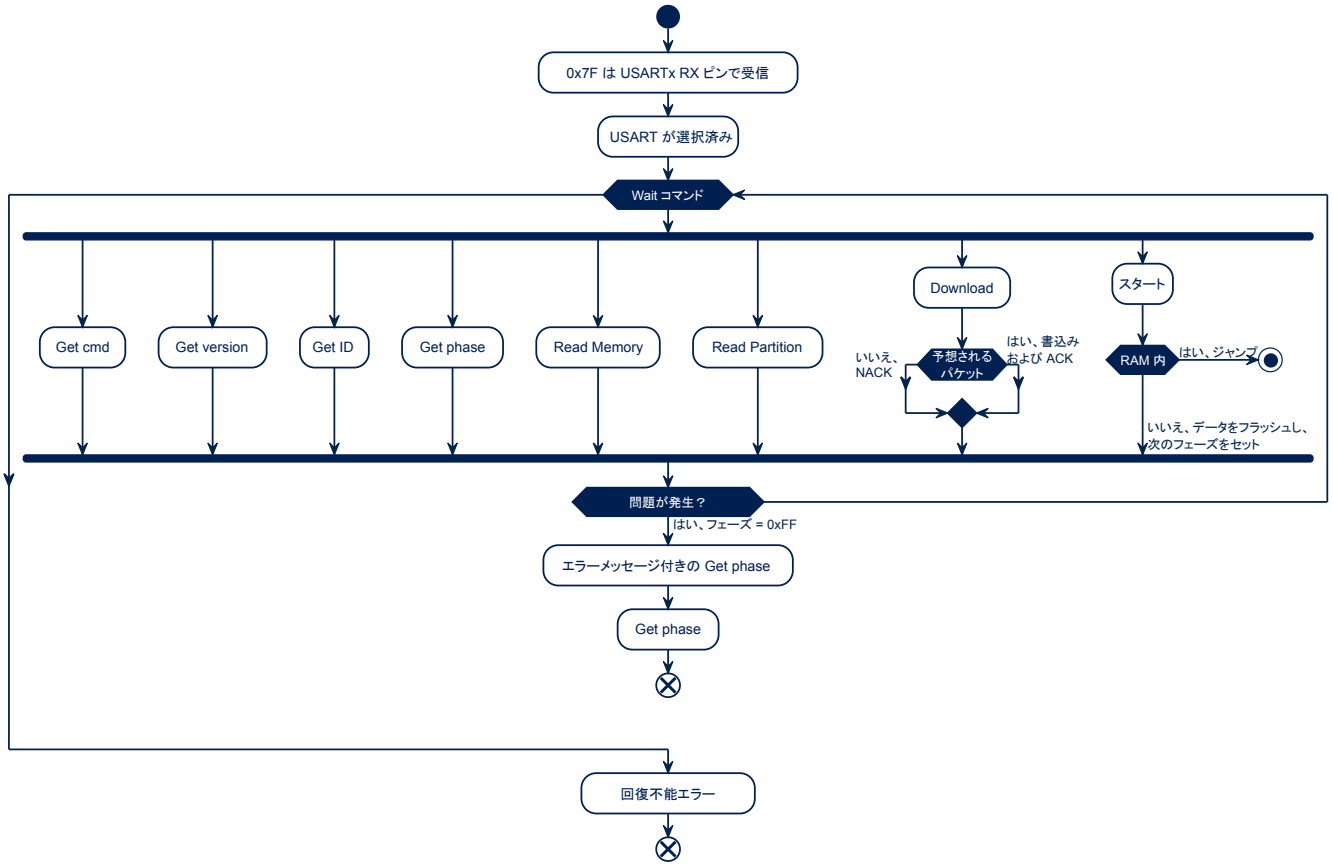
- ボーレート = 115200 ボー
- 8 ビットデータ
- 偶数パリティ
- 1 個のスタートビット
- 1 個のストップビット

2.3 UART/USART 接続

シリアル･ブート･モードに入ると（[13] のブート･ピンを参照）、すべての UART/USART インスタンスが ROM コードによってスキャンされ、各インスタンスについて USARTx_RX ライン･ピンを監視しながら、0x7F データ･フレーム（1 個のスタートビット、0x7F データビット、パリティなしビット、1 個のストップビット）の受信を待ちます。

2.4 UART/USART メイン･ループ

図 6. UART/USART のメイン･シーケンス



デバイスは、次のコマンド･セットに対応しています。

- Get cmd（ブートローダのバージョンと、ブートローダの現在のバージョンでサポートされている使用可能なコマンドを取得します）
- Get version（ソフトウェアのバージョンを取得します）
- Get ID（チップ ID を取得します）
- Get phase(*)（フェーズ ID、つまりダウンロードされるパーティションを取得します）
- Download（イメージをデバイスにダウンロードします）
- Read Memory（SRAM メモリを読み出します）

- Read Partition(*)
- Start（ユーザ・コードに移動します）

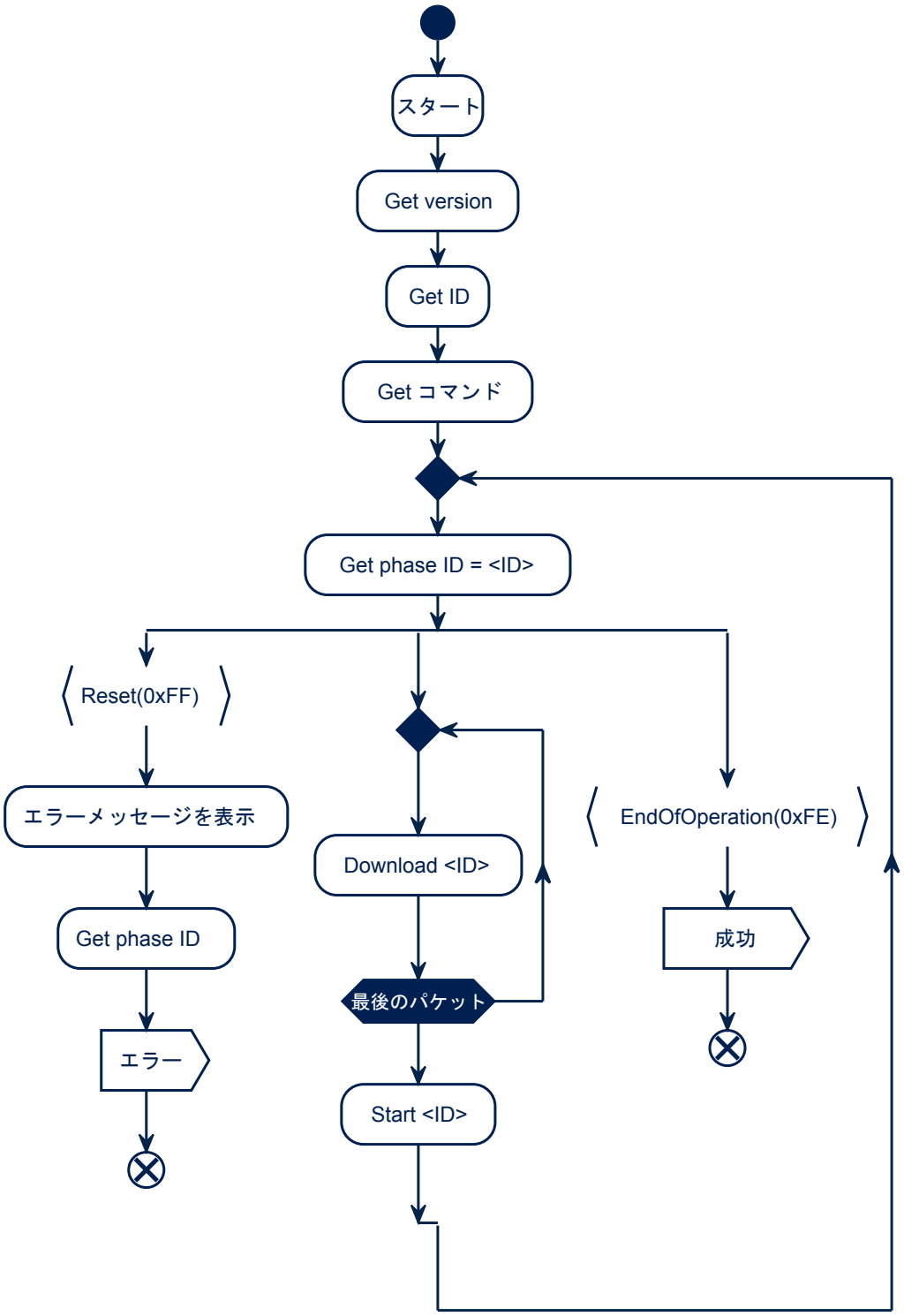
(*) STM32MP1 固有のコマンド

図 6 に、さまざまなダウンロード・フェーズでのデバイス・コード実行を示します。ホストでは最初に Get cmd コマンド、次に Get phase コマンドを使用し、フェーズ ID に応じて、適切なデータを送信します。

- 0：ホストは、レイアウト・ファイルをダウンロードするためのデータを送信します。
- 1：ホストは、プログラマ・サービスで使用される FSBL（=TF-A）を含むファイルをダウンロードするためのデータを送信します。
- 3：ホストは、プログラマ・サービスに使用される SSBL（=U-Boot）を含むイメージをダウンロードするためのデータを送信します。
- [4, F]：その他のブート・パーティション
- [10, F0]：ユーザ・パーティション
- F1：UART/USART コマンド用に予約済みです。
- [F2, FD]：仮想パーティション。OTP として不揮発性メモリにアクセスするために使用されます。
- FE：操作終了
- FF：リセット

データが破損している場合、デバイスはエラーを返します。NACK エラーの場合、システムで動作が再開されます。ABORT エラーの場合、システムは次の GetPhaseId の後に、応答の値 RESET（0xFF）でリブートされます。

図 7. STM32CubeProgrammer ダウンロード・シーケンス



アボート応答

アボート応答は、重大な問題（署名チェック中のエラーまたは書込み中のエラー）が発生した場合に、ホストに通知するために使用されます。回復させるには、システムをリセットして通信を再開する必要があります。アボート応答が発生すると、ダウンロードしたイメージはただちに破棄されます。次の GetPhaseId の後に、デバイスでは応答の値 RESET（0xFF）でシステムが強制的にリセットされます。

2.5 UART/USART コマンド·セット

サポートされるコマンドを次の表に示します。各コマンドの詳細については、次のセクションで説明します。

表 2. USART コマンド

コマンド	コマンド·コード	コマンドの説明
Get	0x00	実行中のエレメントのバージョンと、サポートされている使用可能なコマンドを取得します。
Get Version	0x01	バージョンを取得します。
Get ID	0x02	デバイス ID を取得します。
Get phase	0x03	フェーズ ID、つまりダウンロードされるレイアウト·ファイル内のパーティションの識別子を取得します。
Read Memory	0x11	アプリケーションで指定されたアドレスを始点として、最大 256 バイトのメモリを読み出します。
Read Partition	0x12	アプリケーションによって指定されたオフセットの位置から、パーティションの最大 256 バイトを読み出します(UART/USART ブートローダ·プロトコル v4.0 の新機能)。
Start(移動)	0x21	RAM に配置されたユーザ·アプリケーションにジャンプするか、受信データを不揮発性メモリ内にフラッシュします。
Download(メモリへの書込み)	0x31	イメージをダウンロードします。
消去	0x43	USART プロトコル v3 に存在し、STM32MP1 では使用されません。
Extended Erase	0x44	USART プロトコル v3 に存在し、STM32MP1 では使用されません。
Special	0x50	USART プロトコル v3 に存在し、STM32MP1 では使用されません。
Extended Special	0x51	USART プロトコル v3 に存在し、STM32MP1 では使用されません。
Write Protect	0x63	USART プロトコル v3 に存在し、STM32MP1 では使用されません。
Write Unprotect	0x73	USART プロトコル v3 に存在し、STM32MP1 では使用されません。
Readout Protect	0x82	USART プロトコル v3 に存在し、STM32MP1 では使用されません。
Readout Unprotect	0x92	USART プロトコル v3 に存在し、STM32MP1 では使用されません。
Get Checksum	0xA1	USART プロトコル v3 に存在し、STM32MP1 では使用されません。

通信の安全性

STM32CubeProgrammer(PC)からデバイスへのすべての通信は、次のように確認されます。

- UART/USART の偶数パリティがチェックされます。
- 各コマンドに対して、ホストからバイトとその歩数(XOR = 0x00)が送信されます。
- デバイスで、送受信されたデータブロックのチェックサムが実行されます。以前のすべてのバイトの計算された XOR を含むバイトが、各通信の最後に追加されます(チェックサム·バイト)。受信したすべてのバイト(データ + チェックサム)の排他的論理和をとることによって、パケットの最後の結果が 0x00 になる必要があります。ブロック状態を回避するために、待機ループでタイムアウトを管理する必要があります。

各コマンド·パケットは、受容(ACK アンサー)、破棄(NACK アンサー)、または中断(回復不能エラー)されます。

- ACK = 0x79
- NACK = 0x1F
- ABORT = 0x5F

2.5.1 Get コマンド(0x00)

Get コマンドは、ブートローダのバージョンと、サポートされているコマンドを返します。デバイスは Get コマンドを受信すると、図 6.に示すように、バージョンとサポートされているコマンドのコードをホストに送信します。サポートされないコマンドは、リストから削除されます。

デバイスは、下の表に示すようにバイトを送信します。

表 3. Get コマンドの応答

バイト	説明
1	ACK
2	N = 後続のバイト数 – 1（現在のバイトと ACK を除く）: • N = 8（これらすべてのコマンドに対応する U-Boot の場合） • N = 7（ROM コード STM32MP15x の場合） • N = 6（ROM コード STM32MP13x の場合）
3	UART/USART ブートローダのバージョン（0 < バージョン < 255） 例：0x10 = バージョン 1.0 STM32MP1 では USART プロトコルのバージョンは V4.0 であるため、値は 0x40 です
4	0x00：Get コマンド
5	0x01：Get version
6	0x02：Get ID
7	0x03：Get phase ID
8	0x31：Download コマンドは最後のバイト：11 とする必要があります（cmd は昇順）。
9	0x11：Read memory コマンド（ROM コード STM32MP13x ではサポートされていません）
10	0x12：Read partition コマンド（ROM コードではサポートされません）
11	0x21：Start コマンド
最後	ACK

2.5.2 Get ID コマンド（0x02）

Get ID コマンドを使用すると、デバイス ID のバージョンを取得できます。デバイスはコマンドを受信すると、デバイス ID をホストに送信します。

デバイスからは、次のようにバイトが送信されます。

表 4. Get ID コマンドの応答

バイト	説明
1	ACK
2	N = 1 = 後続のバイト数 – 1（現在のバイトと ACK を除く）
3～4	デバイス ID = STM32MP15x の場合は 0x0500、STM32MP13x の場合は 0x0501
最後	ACK

2.5.3 Get version コマンド（0x01）

Get version コマンドを使用すると、実行コンポーネントのバージョンを取得できます。デバイスはコマンドを受信すると、バージョンをホストに送信します。

デバイスからは、次のようにバイトが送信されます。

表 5. Get version コマンドの応答

バイト	説明
1	ACK
2	ブートローダのバージョン => ソフトウェアのバージョン（ROM コード/TF-A/U-Boot） （0 < バージョン ≤ 255）、例：0x10 = バージョン 1.0
3	オプション・バイト 1: 0x00(1) を参照してください。
4	オプション・バイト 2: 0x00(1)
最後	ACK

1. オプション・バイトでは、汎用ブートローダのプロトコルとの互換性が保持されます。[Ref1]

2.5.4 Get phase コマンド（0x03）

注 このコマンドは STM32MP1 固有です。

Get phase コマンドを使用すると、ホストではフェーズ ID を取得し、ダウンロードされる次のパーティションを識別することができます。

デバイスは Get phase コマンドを受信すると、次のようにパーティション ID をホストに送信します。

表 6. Get phase コマンドの応答

バイト	説明
1	ACK
2	N = 後続のバイト数 -1（現在のバイトと ACK を除く、0 ≤ N ≤ 255）
3	フェーズ ID
4～7	ダウンロード・アドレス
8	X = 追加情報のバイト数（X = N-5）
X	X バイトの追加情報
最後	ACK

ダウンロード・アドレスが存在する場合、メモリ内の転送先アドレスとなります。値が 0xFFFFFFFF の場合、パーティションが NVM に書き込まれます。

フェーズ ID = 0xFF は応答値リセットに対応します。この場合、情報バイトにより、リセット実行の直前に発生したエラーの原因が文字列で示されます。

ROM コード

ROM コードにより、フェーズ = TF-A が送信されます。

バイト 1：ACK
バイト 2：N = 6
バイト 3：フェーズ ID（FSBL = TF-A , 1 を含むファイル）
バイト 4 ~ 7：STM32MP15x へ 0x2FFC2400、STM32MP13x へ 0x2FFDFE00
バイト 8：X = 1
バイト 9：0：予約済み
バイト 10：ACK

TF-A

TF-A により送信

バイト 1：ACK
バイト 2：N = 5
バイト 3：フェーズ ID（SSBL = U-Boot, 3 を含むファイル）
バイト 4 ~ 7：FIP のアドレスを DDR にロード（例 0xC8000000）
バイト 8：X = 0
バイト 9：ACK

U-Boot

U-Boot から追加情報が提供されない場合（N = 5、X = 0）、U-Boot から次のようにバイトが送信されます。
バイト 1：ACK
バイト 2：N = 5
バイト 3：フェーズ ID（プログラムする次のパーティション）
バイト 4 ~ 7：0xFFFFFFFF または DDR のロード・アドレス
バイト 8：X = 0
バイト 9：ACK バイト 8 の場合：X = 0
バイト 9：ACK

たとえば、レイアウト（フェーズ = 0）の場合
バイト 1：ACK
バイト 2：N = 5
バイト 3：フェーズ ID = レイアウト（0）
バイト 4 ~ 7：0xC0000000（DDR ベースアドレス = レイアウト・ファイルのロード・アドレス）
バイト 8：X = 0
バイト 9：ACK

たとえば、TF-A（フェーズ = 1）の場合
バイト 1：ACK
バイト 2：N = 5
バイト 3：フェーズ ID（FSBL1：TF-A = 1）
バイト 4 ~ 7：0xFFFFFFFF
バイト 8：X = 0
バイト 9：ACK

エラーの場合、U-Boot から次のようにバイトが送信されます。
バイト 1：ACK
バイト 2：N = X-5
バイト 3：フェーズ ID = リセット（0xFF）
バイト 4 ~ 7：0xFFFFFFFF
バイト 8 X= 文字列サイズ
X バイト：ASCII の文字列（最大 250 バイト）：エラーの原因
最後のバイト：ACK

2.5.5 Download コマンド（0x31）

Download コマンドは、バイナリ・コード（イメージ）を SRAM メモリにダウンロードしたり、NVM にパーティションを書き込んだりするために使用します。

次の 2 種類の操作が可能です。

- 通常の操作：現在のパーティションのバイナリをデバイスにダウンロードします。初期化フェーズでは、パーティションは SRAM にロードされます。それ例外の場合、書込みフェーズでは、パーティションは NVM に書き込まれます。
- 特殊な操作：符号なしデータを非実行可能メモリ空間にダウンロードします。

Download コマンド後にこれらの操作を完了するには、Start コマンドが必要です。
パケット番号は、操作のタイプと現在のパケットの番号を指定するために使用されます。次の表に、パケット番号の説明を示します。

表 7. パケット番号

バイト	値	説明
3	0x00	通常の操作：現在のフェーズでの書込み
	0xF2	特殊な操作：OTP 書込み
	0xF3	特殊な操作：予約済みです。
	0xF4	特殊な操作 PMIC：NVM 書込み
	その他	予約済みです。
0～2	-	パケット番号：0 から 0xFFFFFF まで増加します（*）。

注 パケット番号は、メモリ・マップされた Flash のみを備えた STM32 マイクロコントローラのようなアドレスではなく、受信したパケットのインデックスです。パケット N のオフセット：現在のパーティション/フェーズでのオフセットは、フルパケットのみが使用される場合、N* 256 バイトです。

- 例：
  - パケット番号 = 0x00603102
    操作 = 通常の操作、パケット番号 = 0x603102
  - パケット番号 = 0xF200000N：N 番目の OTP 部分を送信
    操作 = OTP 書込み、OTP 品名

ホストは、下の表に示すように、デバイスにバイトを送信します。

表 8. Download コマンド

バイト	説明
1	0x31 = ダウンロード（メモリ書込み）
2	0xCE = バイト 1 の XOR
-	ACK または NACK を待つ
3～6	パケット番号（表 7 を参照）
7	チェックサム・バイト：XOR（バイト 3 からバイト 6）
-	ACK または NACK を待つ
8	パケット・サイズ（0 < N < 255）
9 ～（最後-1）	N+1 データ・バイト（最大 256 バイト）
最後	チェックサム・バイト：XOR（バイト 8 から 最後-1）
-	ACK または NACK を待つ

2.5.6 Read memory コマンド（0x11）

Read memory コマンドを使用すると、システム・メモリの任意の有効なメモリ・アドレスからデータを読み出すことができます。
デバイスは Read memory コマンドを受信すると、アプリケーションに ACK バイトを送信します。ACK バイトの送信後、デバイスはアドレス（4 バイト）とチェックサム・バイトを待ってから、受信したアドレスをチェックします。アドレスが有効であり、チェックサムが正しい場合、デバイスは ACK バイトを送信します。そうしない場合、NACK バイトを送信して、コマンドを中断します。
アドレスが有効で、チェックサムが正しい場合、デバイスは N（N = 受信するバイト数 -1）とその相補バイト（チェックサム）を待ちます。チェックサムが正しい場合、デバイスは受信したアドレスを始点として、必要なデータ（N+1 バイト）をアプリケーションに送信します。チェックサムが正しくない場合、NACK を送信してからコマンドを中断します。
メモリにマップされた内容を読み出すために、ホストは次のようにデバイスにバイトを送信します。

表 9. Read memory コマンド

バイト	説明
1	0x11 = メモリ読出し
2	0xEE = バイト 1 の XOR
-	ACK または NACK を待つ
3～6	開始アドレス
7	チェックサム・バイト：XOR（バイト 3 からバイト 6）
-	ACK または NACK を待つ
8	受信するバイト数 – 1（N = [0, 255]）
9	チェックサム・バイト：XOR（バイト 8）
-	ACK または NACK を待つ

2.5.7 Read partition コマンド（0x12）

注 このコマンドは STM32MP1 固有です。

フェーズに関連付けられたデバイスはメモリにマップされないため、Read コマンドを使用することにより、1 つのパーティション内の任意の有効なオフセットからデータを読み出すことができます。

デバイスは Read memory コマンドを受信すると、アプリケーションに ACK バイトを送信します。ACK バイトの送信後、デバイスはパーティション ID、オフセット（4 バイト）、およびチェックサム・バイトを待ってから、受信したアドレスをチェックします。アドレスが有効であり、チェックサムが正しい場合、デバイスは ACK バイトを送信します。そうしない場合、NACK バイトを送信して、コマンドを中断します。

アドレスが有効であり、チェックサムが正しい場合、デバイスは、送信されるバイト数 - 1（N バイト）とその相補バイト（チェックサム）を待ちます。チェックサムが正しい場合、デバイスは受信したアドレスを始点として、必要なデータ（N バイト）をアプリケーションに送信します。チェックサムが正しくない場合、NACK を送信してからコマンドを中断します。

NVM のパーティションを読み出すために、ホストは下の表に示すようにバイトをデバイスに送信します。

表 10. Read Partition コマンド

バイト	説明
1	0x12 = パーティション読出し
2	0xED = バイト 1 の XOR
-	ACK または NACK を待つ
3	パーティション ID = 値
4～7	オフセット・アドレス
8	チェックサム・バイト：XOR（バイト 3 からバイト 7）
-	ACK または NACK を待つ
9	受信するバイト数 – 1 (N = [0,255])
10	チェックサム・バイト：XOR（バイト 9）
-	ACK または NACK を待つ

2.5.8 Start コマンド（0x21）

Start コマンドは次のように使用します。

- メモリにダウンロードしたばかりのコードまたは他のコードを、アプリケーションによって指定されたアドレスに分岐して実行します。デバイスは Start コマンドを受信すると、アプリケーションに ACK バイトを送信します。アドレスが有効である場合、デバイスは ACK バイトを送信してこのアドレスにジャンプします。そうしないと、NACK バイトを送信してコマンドを中断します。
- ホストがアドレス 0xFFFFFFFF を示した場合に、最後のダウンロード・コマンドを完了します。ホストは、下の表に示すように、デバイスにバイトを送信します。

表 11. Start コマンド

バイト	説明
1	0x21 = 開始
2	0xDE = バイト 1 の XOR
-	ACK または NACK を待つ
3～6	開始アドレス または 0xFFFFFFFF
7	チェックサム・バイト：XOR（バイト 3 からバイト 6）
-	ACK または NACK を待つ

3 USB

3.1 DFU プロトコル

組込みプログラミング･サービスでは、DFU プロトコル v1.1 を使用しています（詳細は [5] を参照してください）。

DFU v1.1 プロトコルとの唯一の違いは、次のとおりです。DFU_DETACH は dfuIDLE 状態で許容され、ランタイム･モード appIDLE に戻ります。

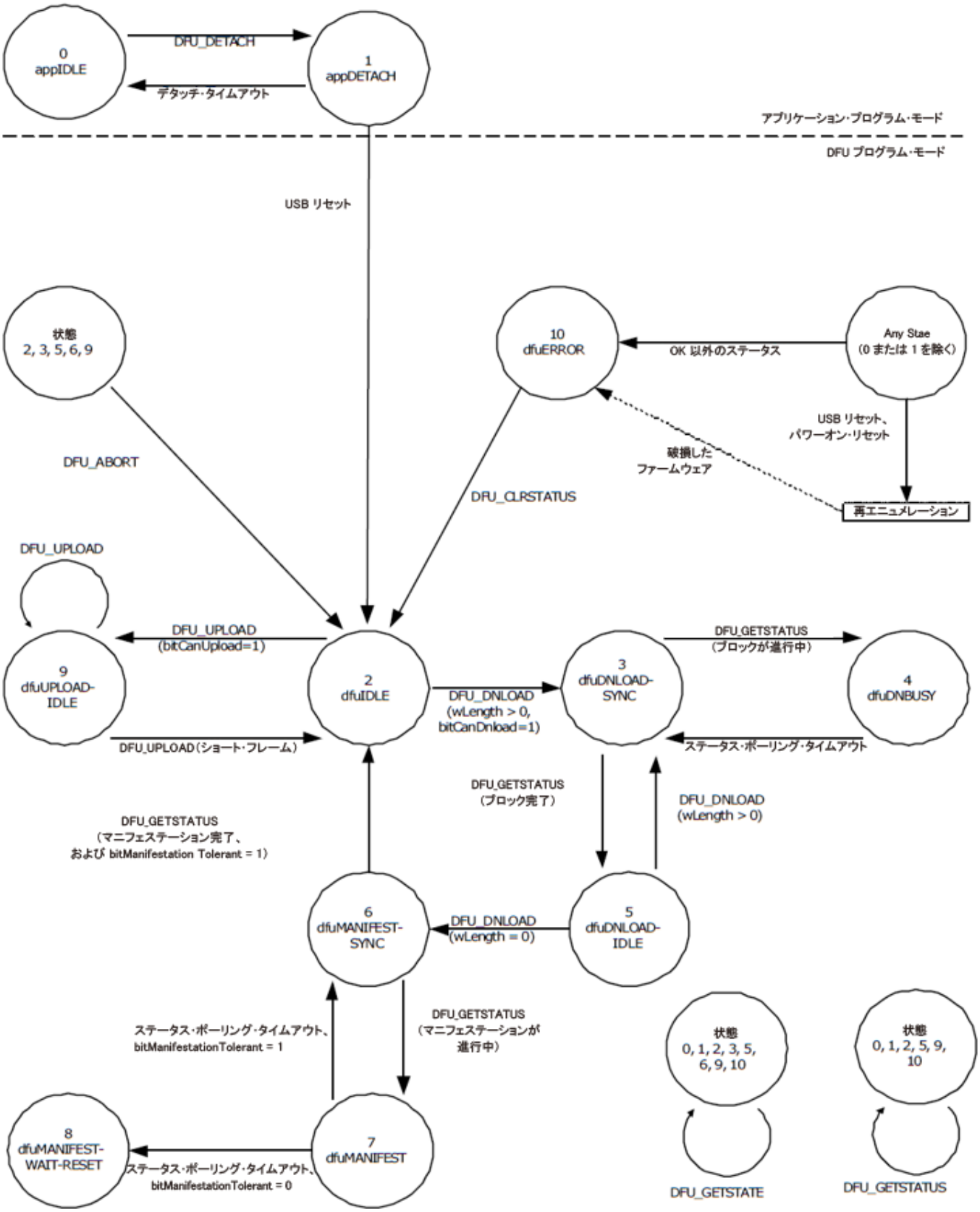
注 この動作は、U-Boot DFU スタック、STM32Programmer（「-detach」オプション付き）、および dfu-utils（-e オプション付き）ですでにサポートされています。

dfuIDLE 状態では、実行を継続するためにデバイスでの USB リセットが必要になります。次のセクションでは、STM32MP1 の仕様のみを示します。

注意 STM32 マイクロコントローラで使用されている DFU 標準の STMicroelectronics 拡張機能は、STM32MP1 シリーズではサポートされていません。これらの拡張機能は DFUSE とも呼ばれます。これらの拡張機能を使用する場合は、DFU のバージョンを v1.1a にする必要があります（詳細は [2] を参照してください）。

DFU の状態については、インタフェースのステート遷移図で説明しています（下の図を参照してください。典拠：[5] の A.1）。

図 8. インタフェースのステート遷移図



3.2 USB シーケンス

USB 記述は TF-A と STM32MP15x ROM コードで共有されるため、STM32MP15x のための TF-A のマニフェステーション後に新しいエニュメレーションは必要ありません。そのため、STM32MP15 の ROM コードでは、代替設定が ROM コードでサポートされていない場合でも、TF-A で使用される代替設定を提示する必要があります。

STM32MP13x の場合、USB コントローラは FSBL フェーズでのマニフェステーションの前に ROM コードで切断されます。USB PHY がリセットされない場合でも、FSBL=TF-A で新しいエニュメレーションが必要です。

DFU アプリケーションである dfu-util の STM32CubeProgrammer は、代替設定による複数のパーティションのダウンロードに対応しています。したがって、ROM コード TF-A および U-Boot のすべての DFU スタックは、それがマニフェステーション･トレラントであることを示す必要があります（DFU 属性で bitMainfestationTolerant = 1）。

STM32MP13x での ROM コードと TF-A の間、または TF-A と U-Boot の間で再エニュメレーションが必要な場合、次のシーケンスで DFU のデタッチがリクエストされます。

1. ホストのリクエスト状態、デバイスのアンサーは dfuIdle（状態 2）です。
2. ホスト･リクエスト DFU_UPLOAD（GetPhase）
3. デバイスは同じフェーズ 0x0 を示しますが、追加情報バイトに「デタッチが必要」フラグ付きで示します。
4. ホストがコマンド DFU_DETACH を送信し、デバイス上で DFU は dfuIdle モードになっています。
5. ホストで USB がリセットされます。

図 9 に すべてそろった USB シーケンスの詳細を示します。

図 9. USB シーケンス - 1

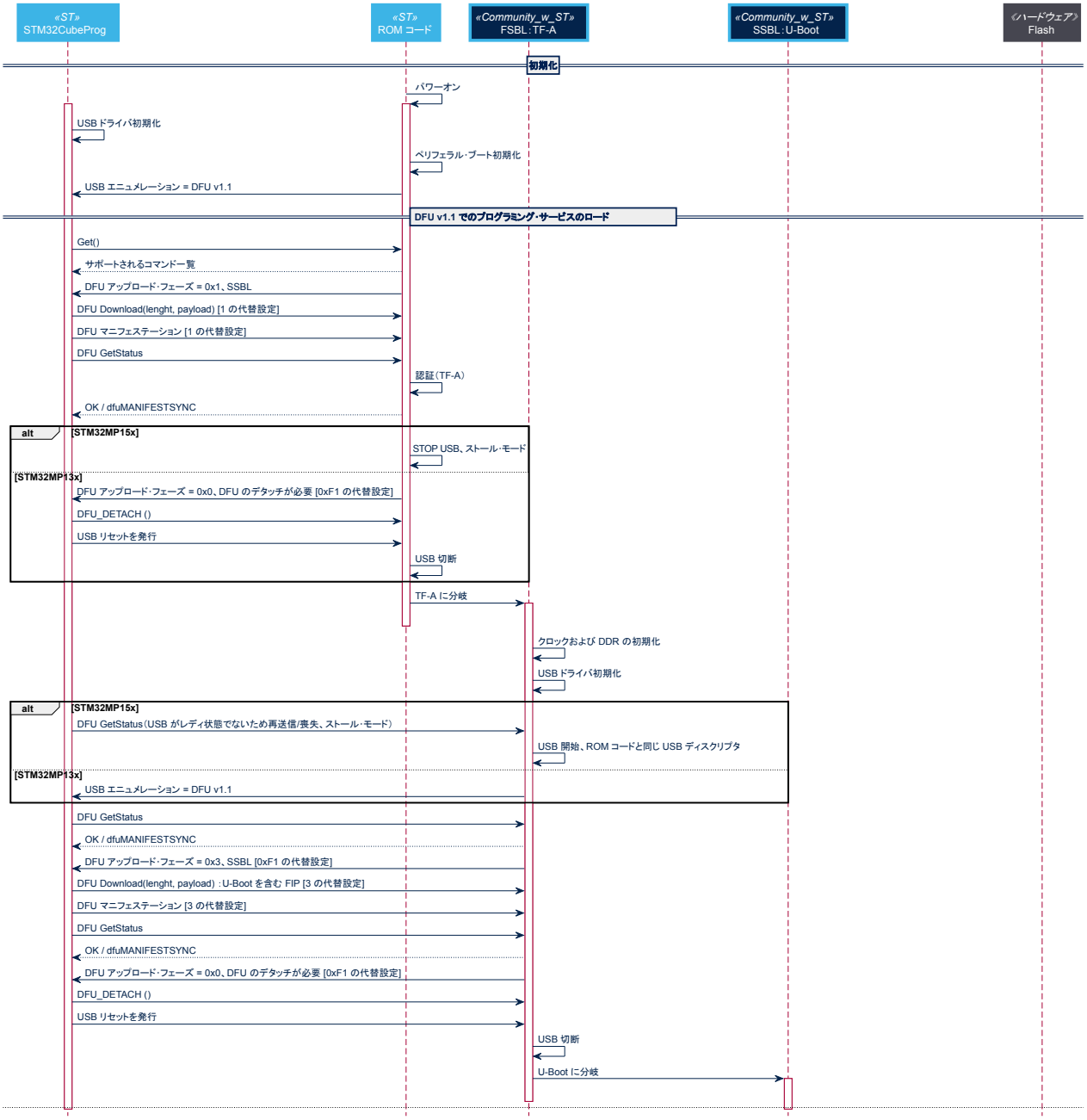
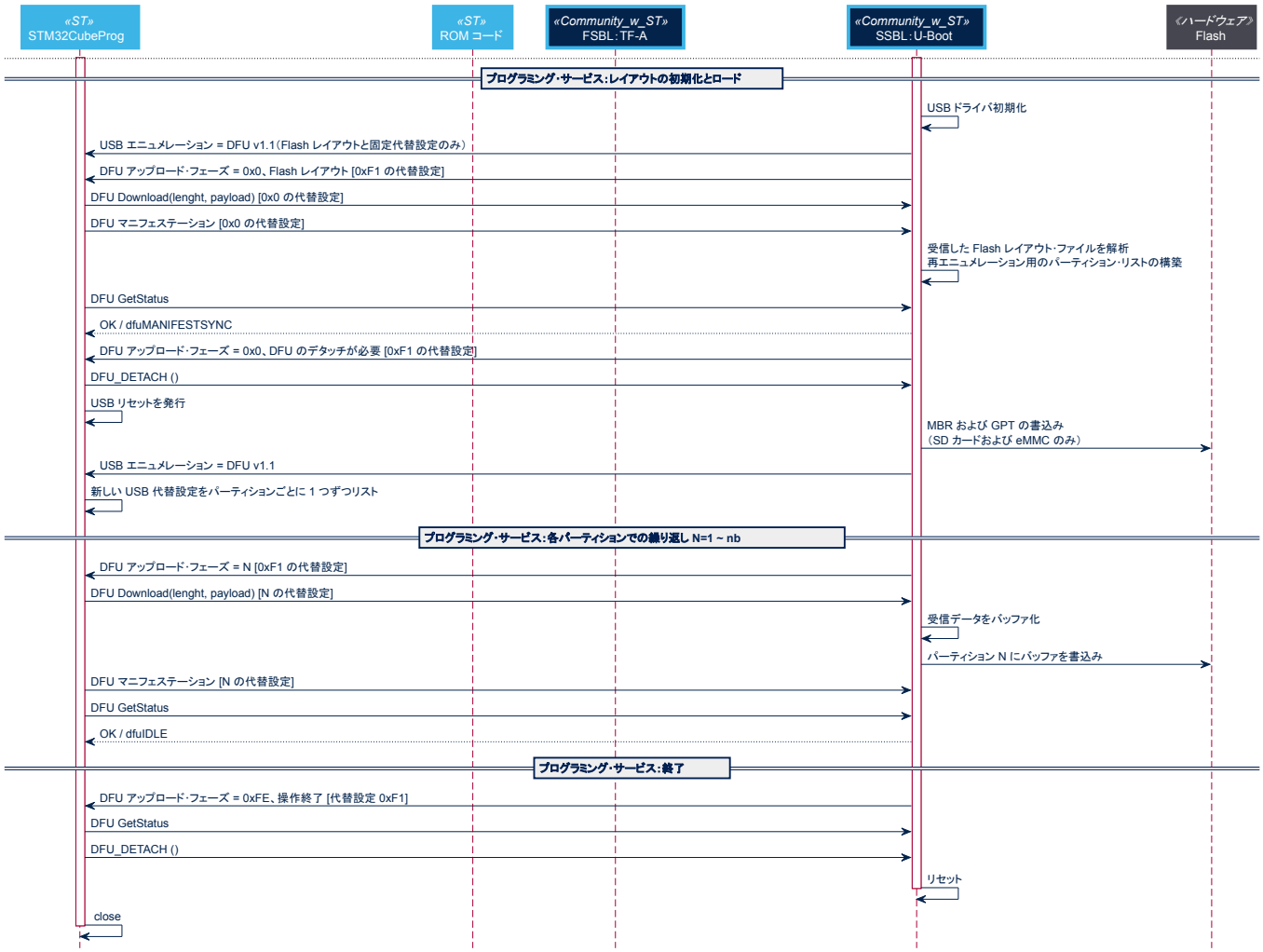


図 10. USB シーケンス - 2



3.3 DFU エニュメレーションと代替設定

USB エニュメレーション中、STM32 デバイス情報が DFU モードのデバイス・ディスクリプタに提示されます。

- idVendor =「STMicroelectronics」の場合は 0x0483
- idProduct =「DFU モードの STM デバイス」の場合は 0xdf11
- iSerial = 固有のデバイス ID から構築される文字列
- iProduct = Product 文字列で、残りのすべての STM32 情報を提供するために STMicroelectronics がエンコードします。

```
<Info>@Device ID /<DevId>, @Revision ID /<RevId> [, @Name /<DevName>]
```

このとき

  - <Info> = 使用される USB スタックに関する情報、例：ROM コード内の HS モードの DFU、または U-Boot 内の TFA USB ダウンロード・ガジェット
  - <DevId> = デバイス ID 0x500（STM32MP15x の場合）0x501（STM32MP13x の場合）
  - <RevId>= シリコン・リビジョン
  - <DevName> = ROM コードで提供されるのではなく、デバイス品名（RPN）で構築されるデバイス名

ROM コードと TF-A 用に RAM にロードされたり、U-Boot 用に NVM でプログラムされたりする各パーティションは、DFU プロファイルの代替設定で使用できます。

- フェーズ ID は、代替設定識別子と整合します。
- デバイスは、特定の代替設定（最後の設定）で UPLOAD により GetPhase に対応します。

- 代替設定文字列ディスクリプタの名前は、[4] 10 章の説明に従います。

```
@Target Memory Name/Start Address/Sector(1)_Count*Sector(1)_Size
Sector(1)_Type,Sector(2)_Count*Sector(2)_SizeSector(2)_Type,...
...,Sector(n)_Count*Sector(n)_SizeSector(n)_Type
```

STM32MP1 のパーティションはメモリマップされないため、「開始アドレス」はパーティション ID（フェーズ ID の場合は 0x + 2 桁）に置き換えられます。

パーティションには 1 つのセクタのみがあり、サイズはレイアウト·ファイルから計算されます。STM32MP1 は、2 種類のパーティションにのみ対応しています。

- a（0x41）：レイアウト·ファイルで選択されていないパーティション用に読出し可能。
- e（0x45）：更新が想定されるパーティション用に読出しおよび書込み可能。

一部の仮想パーティション（NVM 上のメモリに直接リンクされない）が追加され、予約済みのフェーズ ID を使用します。

- 0xF1 = コマンド GetPhase 用に予約されたパーティション ID
- 0xF2 = OTP 用に予約されたパーティション ID
- 0xF4 = PMIC NVM 用に予約されたパーティション ID

3.3.1 仮想コマンド·パーティション 0xF1（GetPhase/SetOffset/Start）

仮想パーティション 0xF1 は、次のコマンドに使用されます。

- DFU_UPLOAD：GetPhase コマンド
- DFU_DOWNLOAD：SetOffset および Start コマンド（メモリ内のアドレスにジャンプ）

3.3.1.1 DFU ダウンロード

これは U-Boot でのみサポートされます。SetOffset コマンドのフォーマットを以下に示します。

表 12. DFU SetOffset コマンド

バイト	説明
1	フェーズ ID：0x00 ~ 0xF0 の通常のパーティション
2～5	パーティション内のオフセット

SetOffset を受信すると、現在のフェーズ ID が更新され（GetPhase の場合）、選択されたパーティションでの次のアップロード/ダウンロードにオフセットが使用されます。

メモリにロードされたコードを実行するための Start コマンドのフォーマットを以下に示します。

表 13. DFU Start コマンド

バイト	説明
1	0xFF
2～5	ジャンプ先のメモリ·アドレス

3.3.1.2 DFU アップロード

この代替設定に基づいてアップロードされる内容は、GetPhase コマンドの実行結果です。

表 14. DFU GetPhase コマンドの応答

バイト	説明
1	フェーズ ID
2～5	ダウンロード·アドレス
6～9	オフセット
10 ~ 最後	追加情報オプション、サイズ：0 ~ 250 バイト

ダウンロード·アドレスが存在する場合、メモリへのロード·アドレスとなります。値が 0xFFFFFFFF の場合、フェーズ ID によって識別されるパーティションは NVM に書き込まれます。

オフセット·フィールドは、パーティション操作で使用される現在のオフセットとなり、デフォルトでは 0 です。

付加情報の内容は、フェーズ ID 値によって決定されます。

- フェーズ ID = 0xFF、文字列でリセット·リクエストを引き起こすエラーの原因
- フェーズ ID = 0x0、レイアウト·ファイルでは情報バイトは次のようになります。
  - バイト 1 リセット必要表示:
    DFU_DETACH がリクエストされた場合は 1
    DFU_DETACH がリクエストされていない場合は 0、またはなし

3.4 ROM コードおよび TF-A の DFU スタック

TF-A および ROM コードでは、STMicroelectronics DFU v1.1 スタックを使用します。

3.4.1 STM32MP15x デバイス用の ROM コードの最初の USB エニュメレーション

STM32MP15x の ROM コードにより、USB と DFU スタックが初期化され、内部 RAM の TF-A のロードと実行が可能になります。TF-A の実行前、TF-A でのマニフェステーションが dfuMANIFESTSYNC 状態になると、USB スタックの割込みが発生します。

TF-A では、USB デバイスが復元され、DFU スタックが再開されます(USB リセット、エニュメレーションなし)。TF-A で受信する最初のリクエストは、dfuIDLE を返さなければなりません。最初のエニュメレーション ROM コードで、代替設定により、ROM コードおよび FSBL TF-A によって使用されるパーティションが提示されます。

表 15. STM32MP15x ROM コード USB エニュメレーション

代替設定	PhaseID	文字列ディスクリプタ	ROM コードのサポート	TF-A のサポート
0	0	@Partition0 /0x00/1*256Ke	なし	いいえ
1	1	@FSBL /0x01/1*1Me	あり	不可
2	2	@Partition2 /0x02/1*1Me	なし	いいえ
3	3	@Partition3 /0x03/1*16Me	なし	SSBL を含むファイル(FIP)
4	0xF1	@virtual /0xF1/1*512Ba	GetPhase	GetPhase

サイズは、ROM コードで固定されている場合でも、柔軟に選択されます。

- Partititon0 = レイアウト·ファイルは 256 KB 未満である必要があります
- FSBL = TF-A のサイズは内部 RAM によって制限されます(最大 1 MB に設定)
- Partition2 = 将来の使用のために予約済み(1 MB)
- Partition3 = SSBL:カーネルを直接ロードできるように、16 MB が使用されます
- Partition4 = 将来の使用のために予約済み(16 MB)
- 0xF1 = GetPhase:読出し専用、通常は 256 バイトに制限

3.4.2 STM32MP13x デバイス用の ROM コードの最初の USB エニュメレーション

STM32MP13x の ROM コードは USB と DFU スタックを初期化し、内部 RAM での TF-A のロードと実行を可能にします。TF-A の実行前に、DFU DETACH リクエスト時に USB が切断され、USB 接続後、TF-A で新しい USB エニュメレーションが行われます。

STM32MP13x での最初のエニュメレーションでは、ROM コードの代替設定には、ROM コードによってのみ使用されるパーティションが表示されます。

表 16. STM32MP13x ROM コード USB エニュメレーション

代替設定	PhaseID	文字列ディスクリプタ
0	1	@FSBL /0x01/1*128Ke
1	0xF1	@virtual /0xF1/1*512Ba

3.4.3 TF-A USB エニュメレーション

STM32MP15x TF-A では、ROM コード USB 設定は新しい USB エニュメレーションなしで再利用されるので、代替設定は ROM コードで STM32MP15x によって実行される最初のエニュメレーションによって固定されます（セクション 3.4.1 を参照）。

STM32MP13x TF-A では、2 番目の USB エニュメレーションが示すのは、SSBL が含まれるファイルのロードに使用されるパーティションのための代替設定です。

パーティションの数とサイズには柔軟性があり、TF-A コードで変更できます。次の表に例を示します。

表 17. STM32MP13x TF-A USB エニュメレーション

代替設定	PhaseID	文字列ディスクリプタ
0	3	@SSBL /0x03/1*16Me
1	0xF1	@virtual /0xF1/1*512Ba

注　上の表に示すように、最大 16 MB の Partition3 はカーネルを含めて FIP に使用できます。

3.5 U-Boot の DFU スタック

U-Boot の既存の DFU スタックが使用されます。

ROM コードまたは TF-A で USB ダウンロード·モードが示されている場合、USB スタック、コントローラ、および USB PHY は DFU モードで初期化する必要があります。

GetPhase コマンドは、パーティション 0xF1 のダウンロード·コマンドで使用できます。

STM32CubeProgrammer は、フェーズが 0xFE または 0xFF になるまでループする必要があります。

- 現在のフェーズを取得：DFU_UPLOAD（GetPhase = 0xF1）
- レイアウト·ファイルのフィールド·バイナリで関連ファイルを検出
- 関連する代替設定でファイルをダウンロード
- マニフェステーションとプール·マニフェステーションの終了

NVM パーティションの場合、アップロード/ダウンロード操作により NVM にアクセスし、マニフェステーションによりデバイスでの最後の操作をフラッシュします。

「メモリ」パーティションの場合、マニフェステーションにより現在の操作とキャッシュのみがフラッシュされます。イン·メモリ開始操作は、仮想パーティションで DFU_DOWNLOAD によってのみ実行されます（3.3.1 を参照）。

3.5.1 U-Boot の最初の USB エニュメレーション

U-Boot は、レイアウトをロードするために必要な代替設定を提示します（最小設定は、フェーズ ID = 0x0 および 0xF1 です）。その他のメモリまたは特殊領域（フェーズ ID > 0xF0）は、OTP などを使用して、その他の代替設定の U-Boot により追加することもできます。

表 18. U-Boot の最初の USB エニュメレーション

代替設定	PhaseID	文字列ディスクリプタ
0	0	@Flashlayout /0x00/1*256Ke
1	0xF1	@virtual /0xF1/1*512Be
2	0xF2	@OTP /0xF2/1*776Be

3.5.2 U-Boot の 2 番目の USB エニュメレーション

2 番目のエニュメレーションでは、U-Boot により、必要なすべての代替設定が提示されます。

- レイアウト·ファイルで定義された、アドレス = フェーズ ID となる NVM または各デバイスの各パーティション（フェーズ、サイズ、タイプは選択した場合のみ書込み可能）
- メモリ·マッピング領域（U-Boot で追加される場合）
- 特殊領域（アドレス = 予約済み、フェーズ ID > 0xF0）

U-Boot 設定およびレイアウト·ファイルの内容に依存します。例を下の表に示します。

表 19. U-Boot の 2 番目の USB エニュメレーション

代替設定	PhaseID	文字列ディスクリプタ
0	0	@Flashlayout /0x00/1*4Ke
1	1	@fsbl1/0x01/1*256Ke
2	2	@fsbl2 /0x02/1*256Ke
3	3	@ssbl /0x03/1*512Ke
4	0x10	@bootfs /0x10/1*64Me
5	0x11	@rootfs /0x11/1*512Me
N	M	レイアウトの最終ユーザ·パーティション（N<0xF0）
N+1	-	@sysram /0x2FFF000/1*256Ke
N+2	-	@mcuram /0x30000000/1*284Ke
N+3	-	@ddr /0xC0000000/1*1Ge
最後-1	0xF1	@virtual /0xF1/1*512Ba
最後	0xF2	@OTP /0xF2/1*512Be

改版履歴

表 20. 文書改版履歴

日付	版	変更内容
2019 年 10 月 2 日	1	初版発行
2022 年 3 月 15 日	2	更新: • セクション 1.4 レイアウト·ファイル·フォーマット • セクション 1.6 STM32 イメージ·ヘッダ • 図 4、図 5 および 図 6 • セクション 2.5 UART/USART コマンド·セット セクション 3.4.2 STM32MP13x デバイス用の ROM コードの最初の USB エニュメレーションを追加
2022 年 3 月 22 日	3	ドキュメント分類を更新

目次

表一覧

図一覧