参考資料

life.augmented

UM2073
ユーザ・マニュアル

STM32Cube 向け STM32 LoRa® ソフトウェア拡張

はじめに

本書では、I-CUBE-LRWAN と呼ばれる、STM32Lx シリーズ向けの LoRa® 拡張パッケージの実装について説明します。また、LoRaWAN™ と接続して LoRa® 無線リンクを管理する方法についても説明します。

LoRa® は、超低ビットレートの長距離通信を可能とするように設計され、センサの長寿命バッテリ動作を実現する無線通信ネットワークです。LoRaWAN™ は、LoRa® ネットワークとの相互運用を保証する通信およびセキュリティのプロトコルを定義します。

LoRa® 拡張パッケージは、LoRa Alliance™ が策定した LoRaWAN™ プロトコルに準拠しています。

I-CUBE-LRWAN の主な機能は以下のとおりです。

- アプリケーションと統合可能
- 低電力 LoRa® ソリューションを簡単にアドオン可能
- 極めて低い CPU 負荷
- 遅延要件なし
- 小さな STM32 メモリ・フットプリント
- 低電力タイミング・サービスを提供

I-CUBE-LRWAN 拡張パッケージは STM32Cube HAL ドライバに基づいています（セクション 2 を参照）。

本書では、NUCLEO-L053R8、NUCLEO-L152RE、および NUCLEOL476RG 上で動作し、Semtech 社製拡張ボード、SX1276MB1MAS、SX1276MB1LAS、SX1272MB2DAS、SX1262DVK1DAS、SX1262DVK1CAS、および SX1262DVK1BAS を使用する、ユーザ・アプリケーションの例について説明します。

本書は、次のツールを対象としています。

- P-NUCLEO-LRWAN1（LoRa® テクノロジー向けの STM32 Nucleo パック）
- B-L072Z-LRWAN1（村田製作所製 CMWX1ZZABZ-091 LoRa® モジュール内蔵の STM32 Discovery キット）
- I-NUCLEO-LRWAN1（STM32 Nucleo 向けの LoRa® 拡張ボード、USI® 社製 WM-SG-SM-42 LPWAN モジュールベース）
- LRWAN-NS1（P-NUCLEO-LRWAN3 に含まれる RiSiNGHF® 社製モデム RHF0M003 搭載の拡張ボード）

参考資料

目次

1 概要 7

1.1 略記と略語 7

1.2 参照 8

2 LoRa 規格の概要 9

2.1 概要 9

2.2 ネットワーク･アーキテクチャ 9

2.2.1 エンドデバイスのアーキテクチャ 10

2.2.2 エンドデバイスのクラス 10

2.2.3 エンドデバイスのアクティベーション（Join） 11

2.2.4 地域ごとの周波数割当て 12

2.3 ネットワーク層 12

2.3.1 物理層（PHY） 13

2.3.2 MAC 副層 13

2.4 メッセージ･フロー 13

2.4.1 エンドデバイスのアクティベーションの詳細（Join） 13

2.4.2 エンドデバイスのデータ通信（クラス A） 14

2.5 データ･フロー 16

3 I-CUBE-LRWAN ミドルウェアの説明 17

3.1 概要 17

3.2 機能 19

3.3 アーキテクチャ 20

3.4 ハードウェア関連コンポーネント 21

3.4.1 無線リセット 21

3.4.2 SPI 21

3.4.3 RTC 21

3.4.4 入力ライン 22

4 I-CUBE-LRWAN ミドルウェア･プログラミングのガイドライン 23

4.1 ミドルウェアの初期化 23

4.2 ミドルウェア MAC 層の関数 23

4.2.1 MCPS サービス 23

参考資料

4.2.2 MLME サービス 24
4.2.3 MIB サービス 24
4.3 ミドルウェア MAC 層のコールバック 24
4.3.1 MCPS 24
4.3.2 MLME 25
4.3.3 MIB 25
4.3.4 バッテリ・レベル 25
4.4 ミドルウェア MAC 層のタイマ 25
4.4.1 遅延 Rx ウィンドウ 25
4.4.2 Tx フレーム送信の遅延 25
4.4.3 Rx フレームの遅延 26
4.5 ミドルウェアユーティリティの関数 26
4.5.1 タイマ・サーバの API の説明 26
4.5.2 省電力関数 26
4.5.3 システム時間関数 27
4.5.4 トレース関数 28
4.5.5 キュー関数 28
4.6 セキュア・エレメントのエミュレート 29
4.7 ミドルウェア End_Node アプリケーション関数 30
4.7.1 LoRa End_Node 初期化 34
4.7.2 LoRa End_Node Join 要求エントリ・ポイント 34
4.7.3 LoRa End_Node Tx 開始 34
4.7.4 End_Node の Join ステータス要求 34
4.7.5 アップリンク・フレーム送信 34
4.7.6 現在のネットワーク時間の要求 35
4.7.7 4.7.7 次のビーコン・タイミングの要求 35
4.7.8 クラス切替え要求 35
4.7.9 エンドデバイスの現在のクラスの取得 35
4.7.10 ビーコン取得要求 35
4.7.11 ユニキャスト ping スロット情報周期の送信 36
4.8 LIB End_Node アプリケーション・コールバック 36
4.8.1 現在のバッテリ・レベル 36
4.8.2 現在の温度レベル 36
4.8.3 ボード固有 ID 36
4.8.4 ボード・ランダム・シード 36
4.8.5 Rx フレーム作成 37

参考資料

4.8.6 LoRa Joined 通知 37
4.8.7 End_Node クラス･モード変更確認 37
4.8.8 ダミー･アップリンク･フレーム送信 37

5 サンプルプログラムの説明 38
5.1 エンドデバイス･ハードウェアの説明 38
5.2 分割エンドデバイス･ハードウェアの説明（2 チップ･ソリューション） 39
5.3 パッケージの説明 40
5.4 End_Node アプリケーション 41
5.4.1 アクティベーション方法とキー 41
5.4.2 デバッグ･スイッチ 41
5.4.3 センサ･スイッチ 41
5.5 PingPong アプリケーションの説明 42
5.6 AT_Slave アプリケーションの説明 43
5.7 AT_Master アプリケーションの説明 43

6 システム性能 45
6.1 メモリ･フットプリント 45
6.2 リアルタイム制約 45
6.3 消費電力 46

7 改版履歴 48

参考資料

表の一覧

表 1. 略記と略語のリスト . . . 7
表 2. LoRa クラスの用途 . . . 9
表 3. LoraWAN の地域ごとの周波数割当て . . . 12
表 4. ミドルウェア初期化関数 . . . 23
表 5. MCPS サービス関数 . . . 23
表 6. MLME サービス関数 . . . 24
表 7. MIB サービス関数 . . . 24
表 8. MCPS プリミティブ . . . 24
表 9. MLME プリミティブ . . . 25
表 10. バッテリ・レベル関数 . . . 25
表 11. 遅延 Rx 関数 . . . 25
表 12. Tx フレーム送信の遅延の関数 . . . 25
表 13. Rx フレームの遅延の関数 . . . 26
表 14. タイマ・サーバ関数 . . . 26
表 15. 省電力関数 . . . 26
表 16. システム時間関数 . . . 27
表 17. トレース関数 . . . 28
表 18. キュー関数 . . . 28
表 19. セキュア・エレメント関数 . . . 29
表 20. LoRa クラス A 初期化関数 . . . 34
表 21. LoRa End_Node Join 要求エントリ・ポイント . . . 34
表 22. LoRa End_Node Tx 開始 . . . 34
表 23. End_Node の Join ステータス . . . 34
表 24. アップリンク・フレーム送信 . . . 34
表 25. 現在のネットワーク時間 . . . 35
表 26. 次のビーコン・タイミング . . . 35
表 27. クラス切替え要求 . . . 35
表 28. エンドデバイスの現在のクラスの取得 . . . 35
表 29. ビーコン取得要求 . . . 35
表 30. ユニキャスト ping スロット周期 . . . 36
表 31. 現在のバッテリ・レベル関数 . . . 36
表 32. 現在の温度レベル関数 . . . 36
表 33. ボード固有 ID 関数 . . . 36
表 34. ボード・ランダム・シード関数 . . . 36
表 35. Rx フレーム作成関数 . . . 37
表 36. LoRa Joined 通知関数 . . . 37
表 37. End_Node クラス・モード変更確認関数 . . . 37
表 38. ダミー・アップリンク・フレーム送信関数 . . . 37
表 39. サポートされている Nucleo ベースのハードウェア . . . 38
表 40. LoRa 無線拡張ボードの特性 . . . 38
表 41. STM32L0xx の IRQ 優先度 . . . 39
表 42. アプリケーション設定のスイッチ・オプション . . . 42
表 43. BSP プログラミングのガイドライン . . . 44
表 44. End_Node アプリケーションのメモリ・フットプリント値 . . . 45
表 45. 文書改版履歴 . . . 48
表 46. 日本語版文書改版履歴 . . . 49

参考資料

図の一覧

図 1. ネットワーク図 . . . 10
図 2. Tx/Rx タイミング図（クラス A） . . . 10
図 3. Tx/Rx タイミング図（クラス B） . . . 11
図 4. Tx/Rx タイミング図（クラス C） . . . 11
図 5. LoRaWAN の各層 . . . 13
図 6. Join のメッセージ･シーケンス･チャート（MLME プリミティブ） . . . 14
図 7. confirmed-data メッセージ･シーケンス･チャート（MCPS プリミティブ） . . . 15
図 8. unconfirmed-data のメッセージ･シーケンス･チャート（MCPS プリミティブ） . . . 15
図 9. データ･フロー . . . 16
図 10. プロジェクト･ファイルの構造 . . . 18
図 11. ファームウェアの構成 . . . 20
図 12. LoRaMacCrypto モジュールの構成 . . . 29
図 13. 動作モデル . . . 31
図 14. LoRa の状態遷移 . . . 32
図 15. LoRa クラス B システムの状態遷移 . . . 33
図 16. 分割エンドデバイス･ソリューションの概念 . . . 39
図 17. I-CUBE-LRWAN の構造 . . . 40
図 18. PingPong セットアップ . . . 43
図 19. Rx/Tx タイミング図 . . . 45
図 20. STM32L0 の時間に対する消費電流 . . . 47

参考資料

1 概要

STM32Cube 向けの I-CUBE-LRWAN 拡張パッケージは、Arm®(a) Cortex®-M プロセッサをベースとする STM32 32bit マイクロコントローラ上で動作します。

1.1 略記と略語

表 1. 略記と略語のリスト

用語	定義
ABP	Activation by personalization
APP	Application（アプリケーション）
API	Application programming interface（アプリケーション･プログラミング･インタフェース）
BSP	Board support package（ボード･サポート･パッケージ）
FSM	Finite state machine（有限ステートマシン）
HAL	Hardware abstraction layer（ハードウェア抽象化レイヤ）
IOT	Internet of things（モノのインターネット）
LoRa	Long range radio technology（長距離無線技術）
LoRaWan	LoRa wide-area network（LoRa 広域ネットワーク）
LPWAN	Low-power, wide-area network（省電力広域ネットワーク）
MAC	Media access control（メディア･アクセス制御）
MCPS	MAC common part sublayer
MIB	MAC information base
MLME	MAC sublayer management entity
MPDU	MAC protocol data unit（MAC プロトコル･データ･ユニット）
OTAA	Over-this-air activation
PLME	Physical sublayer management entity
PPDU	Physical protocol data unit
SAP	Service access point（サービス･アクセス･ポイント）

a. Arm は Arm Limited （またはその子会社）の米国およびその他の国々における登録商標です。

参考資料

1.2 参照

- LoRa Alliance specification protocol named LoRaWAN version V1.0.3 - 2018 年 3 月 - 最終版 - リリース済み
- IEEE Std 802.15.4TM - 2011. Low-Rate Wireless Personal Area Networks (LR-WPANs)
- LoRaWAN version 1.1 Regional Parameters - RevB - 2018 年 1 月 - リリース済み

2 LoRa 規格の概要

2.1 概要

このセクションでは、LoRa および LoRaWAN に関する推奨事項について、特に本書の主要テーマである LoRa エンドデバイスに注目して全般的な概要を示します。

LoRa は、超低ビットレートの長距離通信を可能とするように設計され、長寿命バッテリ駆動のセンサを実現する無線通信ネットワークです。LoRaWAN は、LoRa ネットワークとの相互運用を確保する通信およびセキュリティのプロトコルを定義します。

LoRa 拡張パッケージは、LoRa Alliance が策定した LoRaWAN プロトコルに準拠しています。

表 2 に、LoRa クラスの用途の定義を示します。これらのクラスの詳細については、セクション 2.2.2 を参照してください。

表 2. LoRa クラスの用途

クラス名	用途
A - すべて	– 遅延の制約のないバッテリ駆動のセンサまたはアクチュエータ – 最もエネルギー効率の良い通信クラス – すべてのデバイスでサポートする必要がある
B - ビーコン	– バッテリ駆動のアクチュエータ – 遅延が制御されているダウンリンク用のエネルギー効率の良い通信クラス – ネットワーク・ビーコンと同期したスロット通信に基づく
C - 連続	– 給電式のアクチュエータ – 連続して受信待機可能なデバイス – ダウンリンク通信では遅延なし

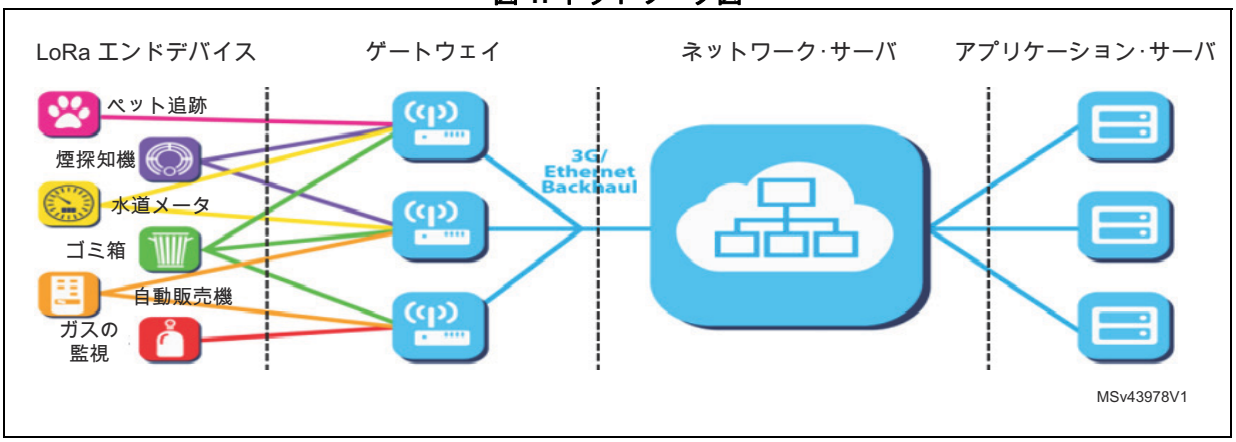
注： **LoRa の物理層は独自仕様ですが、プロトコル・スタック（LoRaWAN）の残りの部分はオープンであり、その開発は LoRa Alliance によって行われます。**

2.2 ネットワーク・アーキテクチャ

LoRaWAN ネットワークは、複数のスターで構成されたスター型トポロジで構築されており、図 1 に示すように、エンドデバイスは単一の LoRa リンク経由で 1 つのゲートウェイに接続されます。

参考資料

図 1. ネットワーク図



2.2.1 エンドデバイスのアーキテクチャ

エンドデバイスは、RF トランシーバ（無線とも呼ばれます）1 つとホスト STM32 マイクロコントローラ 1 つで構成されます。RF トランシーバは、モデム 1 つと RF アップコンバータ 1 つで構成されます。マイクロコントローラは、無線ドライバ、LoRaWAN スタック、およびセンサ·ドライバ（オプション）を実装します。

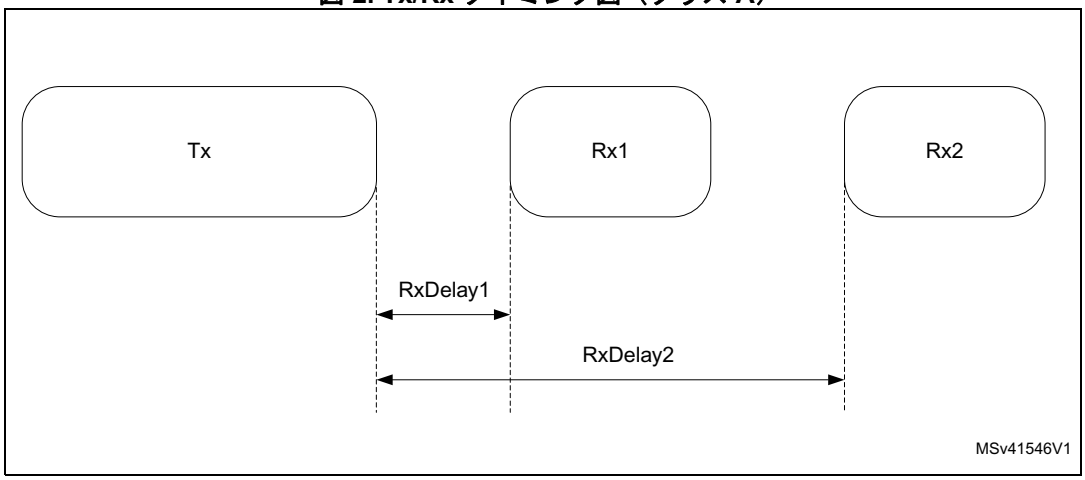
2.2.2 エンドデバイスのクラス

LoRaWAN には、エンドポイント·デバイスのクラスが複数存在し、広い範囲のアプリケーションにおけるさまざまなニーズに対応します。

双方向エンドデバイス - クラス A （すべてのデバイス）

- クラス A の動作は、最小電力のエンドデバイス·システムです。
- エンドデバイスの各アップリンク送信の後に、2 つの短いダウンリンク受信ウィンドウが続きます。
- エンドデバイスからのアップリンク送信のすぐ後にサーバからのダウンリンク通信が行われます（図 2 を参照）。
- 送信スロットは、エンドデバイス側の通信ニーズに基づきます（ALOHA タイプのプロトコル）。

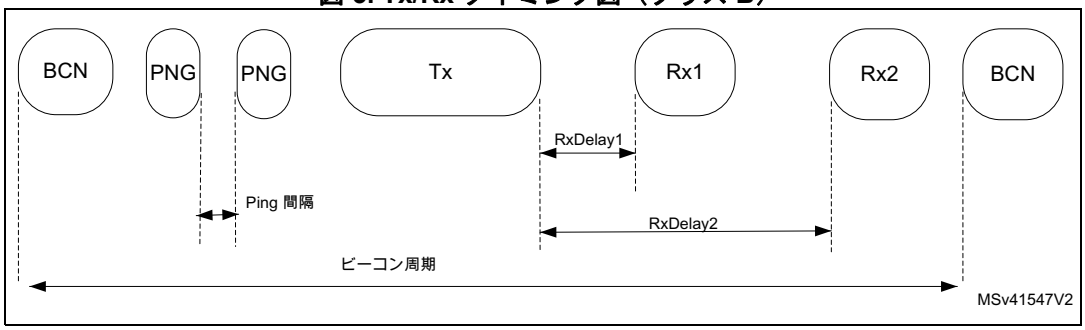
図 2. Tx/Rx タイミング図（クラス A）



受信スロットがスケジュールされた双方向エンドデバイス - クラス B （ビーコン）

- 消費電力は中程度です。
- クラス B デバイスは、スケジュールされた時間に受信ウィンドウを追加開放します（図 3 を参照）。
- スケジュールされた時間に受信ウィンドウを開放するために、エンドデバイスはゲートウェイから時間同期されたビーコンを受信します。

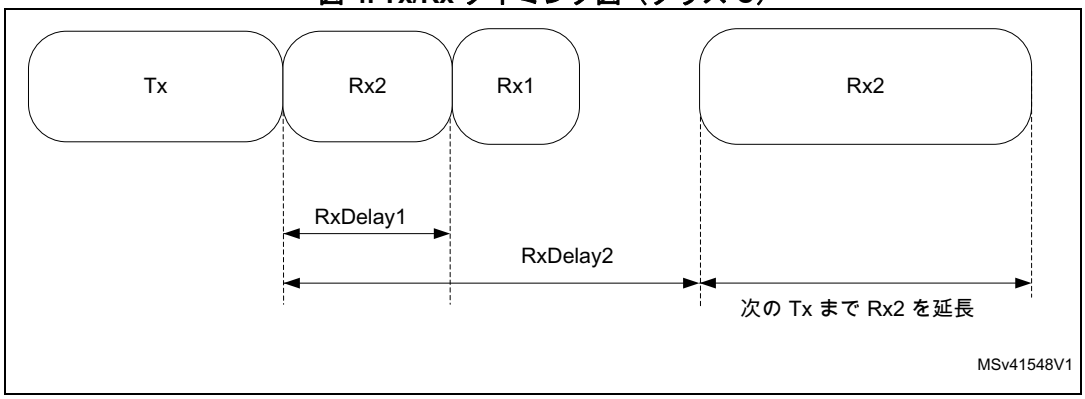
図 3. Tx/Rx タイミング図（クラス B）



最大の受信スロットを持つ双方向エンドデバイス - クラス C （連続）

- 消費電力は大きくなります。
- クラス C のエンドデバイスは、送信時を除いて、ほぼ連続的に受信ウィンドウを開放しています（図 4 を参照）。

図 4. Tx/Rx タイミング図（クラス C）



2.2.3 エンドデバイスのアクティベーション（Join）

Over-the-air activation （OTAA）

OTAA は、LoRa エンドデバイスが LoRa ネットワークに参加するための Join 手順です。LoRa エンドデバイスとアプリケーション･サーバの両方が AppKey と呼ばれる同じ秘密キーを共有します。Join 手順において、LoRa エンドデバイスとアプリケーション･サーバは情報を交換して、次の 2 つのセッション･キーを生成します。

- MAC コマンドの暗号化に使用するネットワーク･セッション･キー（NwkSKey）
- アプリケーション･データの暗号化に使用するアプリケーション･セッション･キー（AppSKey）

Activation by personalization （ABP）

ABP の場合、NwkSkey と AppSkey は、直接 LoRa ネットワークにデータを送信する LoRa エンドデバイスにすでに格納されています。

2.2.4 地域ごとの周波数割当て

LoRaWAN 仕様は、地域ごとに多少異なります。欧州、北米、およびアジアの市場は、周波数割当てと規制要件が異なっています。詳細は、表 3 を参照してください。

表 3. LoraWAN の地域ごとの周波数割当て

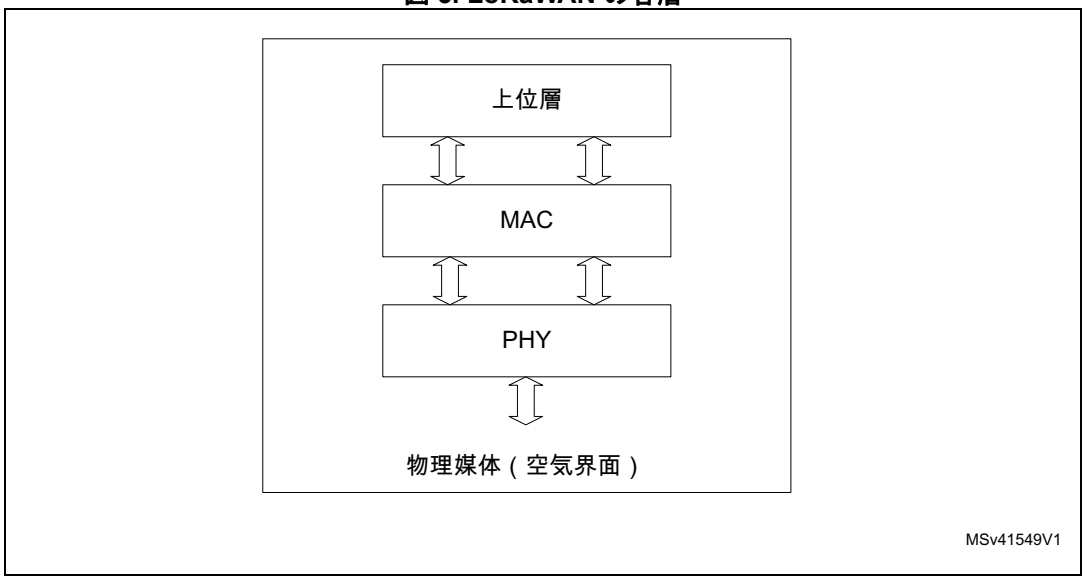
地域	サポート	帯域 [MHz]	デューティ·サイクル	出力
EU	あり	868	1 % 未満	+14 dBm
EU	あり	433	1 % 未満	+10 dBm
US	あり	915	2% 未満（帯域幅 250 kHz 未満の場合） または 4% 未満（帯域幅 250 kHz 以上の場合） 送信スロット 0.4 秒未満	+20 dBm
CN	なし	779	0.1 % 未満	+10 dBm
AS	あり	923	1 % 未満	+16 dBm
IN	あり	865	なし	+20 dBm
KR	あり	920	なし	+10 dBm
RU	あり	868	1 % 未満	+16 dBm

2.3 ネットワーク層

LoRaWAN アーキテクチャは、ブロック（「層」とも呼ばれます）で定義されています。各層は、規格の一部分を処理し、それより上位の層にサービスを提供します。

エンドデバイスは、少なくとも 1 つの無線周波数トランシーバを組み込んだ物理層（PHY）、物理チャネルへのアクセスを提供する MAC 副層、およびアプリケーション層（図 5 を参照）で構成されます。

図 5. LoRaWAN の各層



2.3.1 物理層（PHY）

物理層は、次の 2 つのサービスを提供します。

- PHY データ･サービス。PPDU（physical protocol data units）の Tx/Rx を可能にします。
- PHY 管理サービス。PIB（personal area network information base）管理を可能にします。

2.3.2 MAC 副層

MAC 副層は、次の 2 つのサービスを提供します。

- MAC データ･サービス。物理層における MAC プロトコル･データ･ユニット（MPDU）の送受信を可能にします。
- MAC 管理サービス。PIB 管理を可能にします。

2.4 メッセージ･フロー

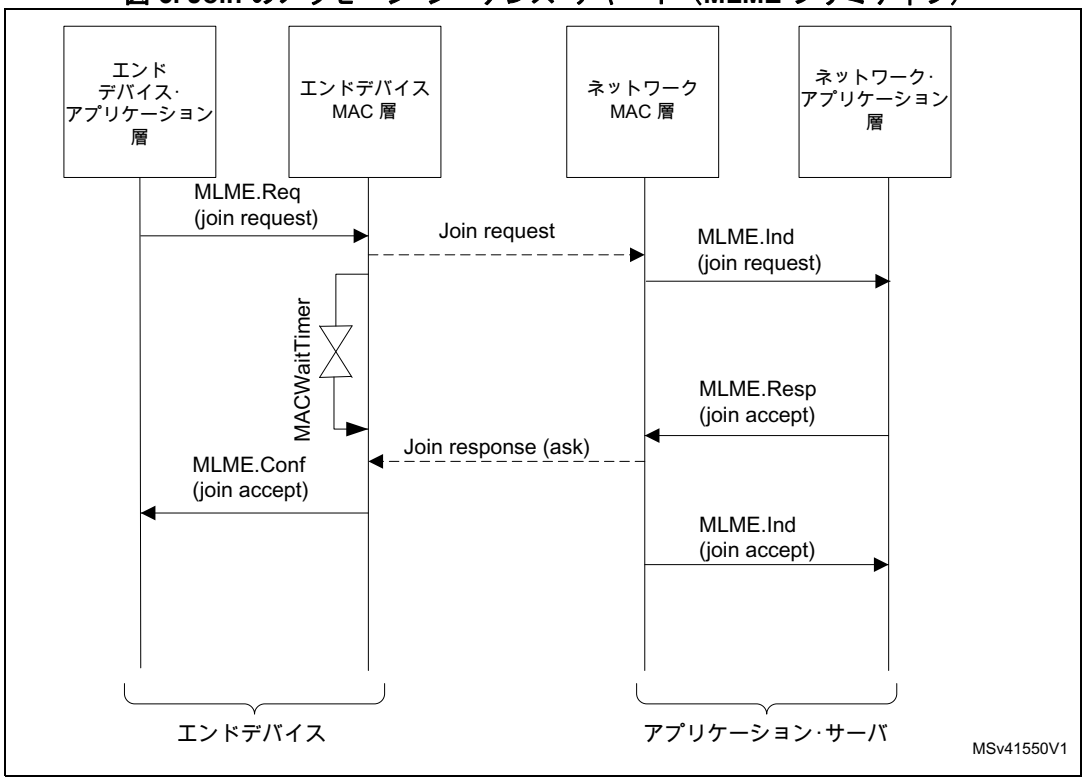
このセクションでは、N ユーザと N 層の間の情報フローについて説明します。サービスへの要求は、サービス･プリミティブにより行われます。

2.4.1 エンドデバイスのアクティベーションの詳細（Join）

LoRaWAN ネットワークで通信する前に、エンドデバイスは、セクション 2.2.3 で説明されている 2 つのアクティベーション方法のどちらかに従って関連付けまたはアクティベーションを行う必要があります。

図 6 のメッセージ･シーケンス･チャート（MSC）には、OTAA アクティベーション方法を示しています。

図 6. Join のメッセージ･シーケンス･チャート（MLME プリミティブ）



2.4.2 エンドデバイスのデータ通信（クラス A）

エンドデバイスは、confirmed-data メッセージの方法（図 7 を参照）または unconfirmed-data メッセージの方法（図 8 を参照）のいずれかの方法を使用してデータを送信します。

エンドデバイスは、前者の場合はレシーバによる ‘Ack’（確認応答）を必要とし、後者の場合は ‘Ack’ を必要としません。

‘Ackreq’ （確認応答要求）付きでデータを送信したエンドデバイスは、確認応答待機時間（‘AckWaitDuration’）が経過するまで､確認応答フレームを待機する必要があります（セクション 4.3.1: MCPSを参照）。

確認応答フレームを受信した場合は送信成功、受信しない場合は送信失敗となります。

参考資料

図 7. confirmed-data メッセージ･シーケンス･チャート（MCPS プリミティブ）

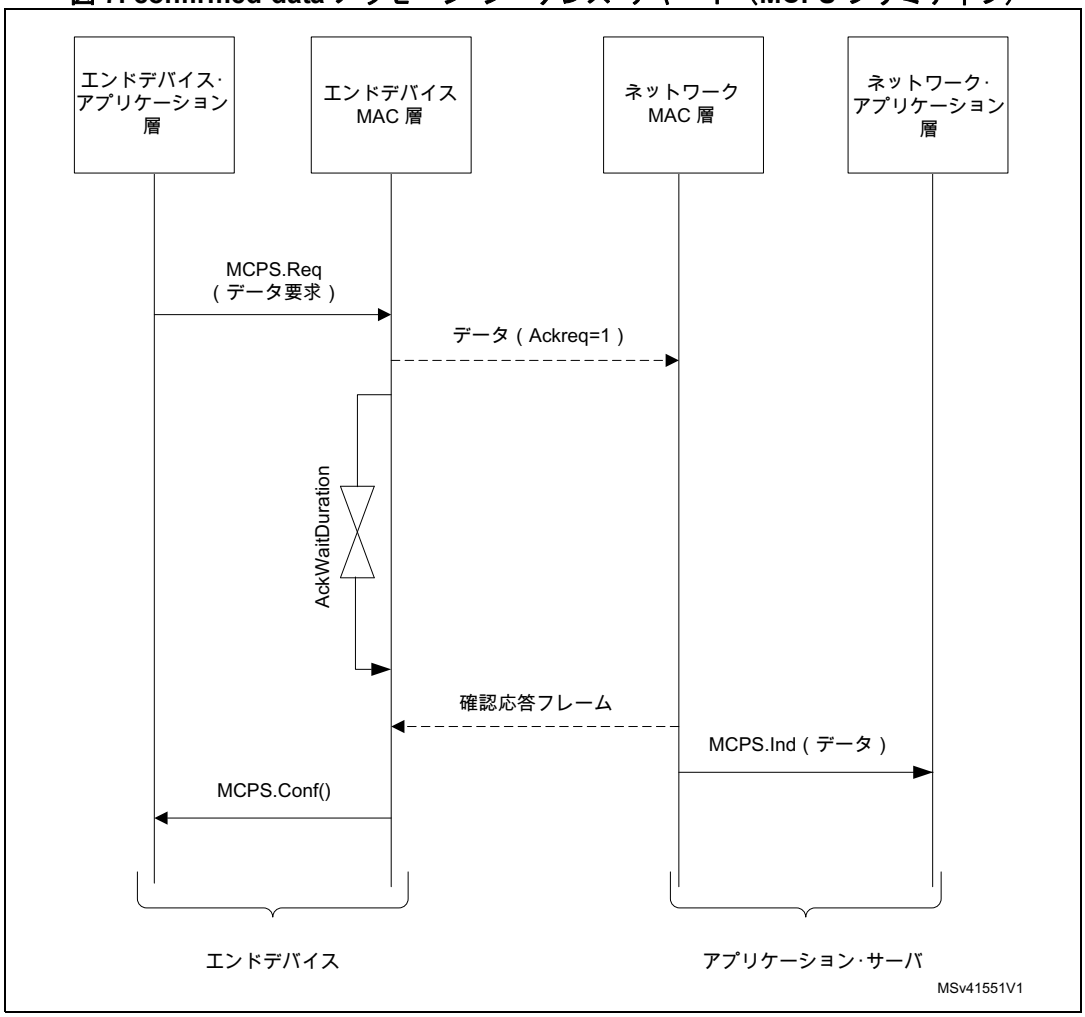
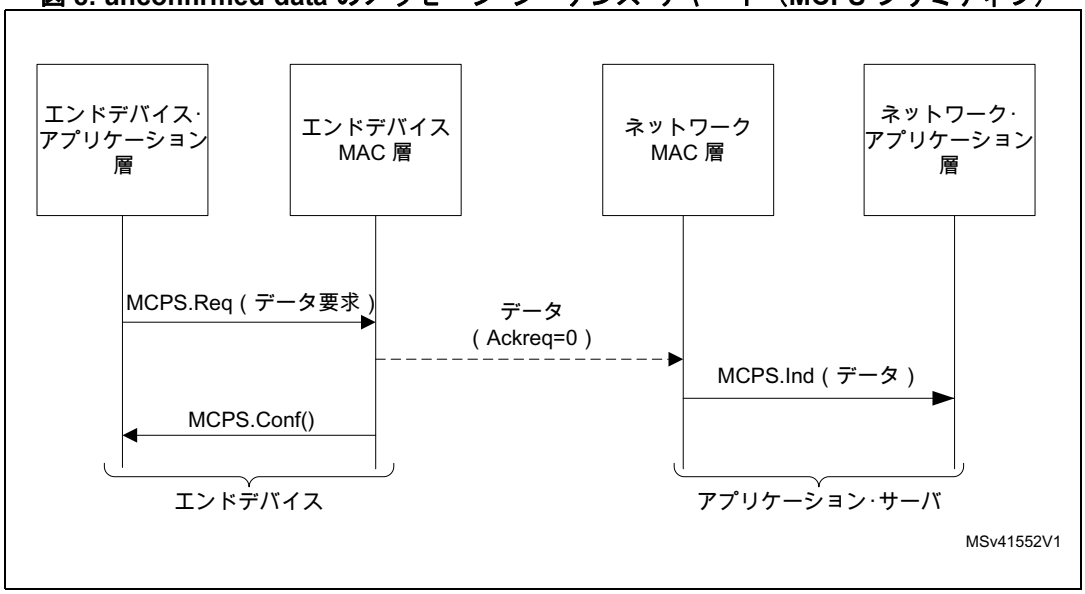


図 8. unconfirmed-data のメッセージ･シーケンス･チャート（MCPS プリミティブ）

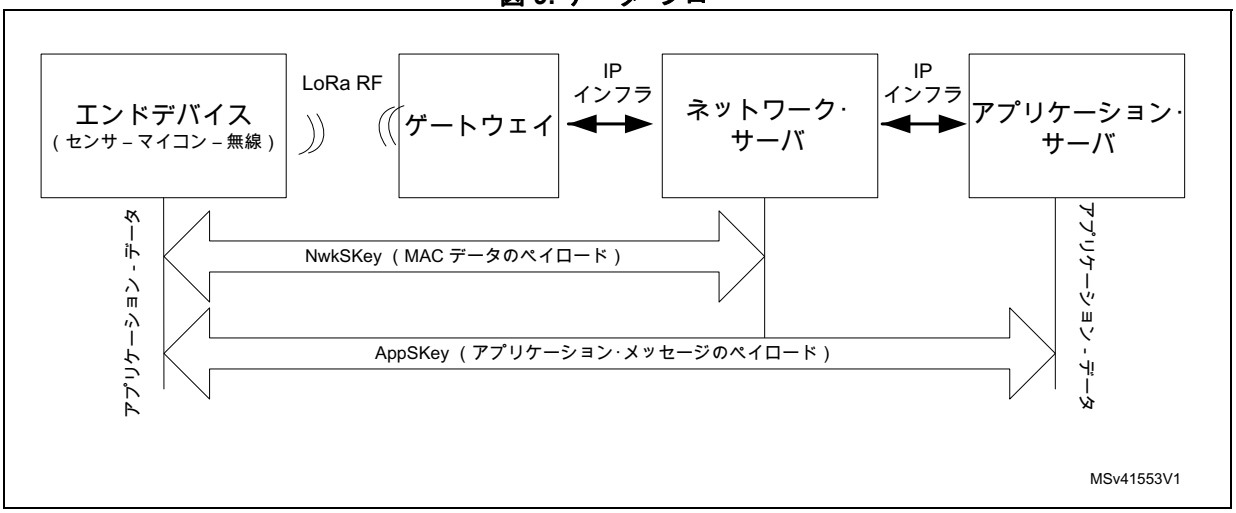


参考資料

2.5 データ·フロー

データの整合性は、ネットワーク·セッション·キー（NwkSKey）とアプリケーション·セッション·キー（AppSKey）によって確保されます。NwkSKey は MAC ペイロード·データの暗号化と復号化に使用され、AppSKey はアプリケーション·ペイロード·データの暗号化と復号化に使用されます。図 9 にデータ·フローを示します。

図 9. データ·フロー



NwkSKey は、エンドデバイスとネットワーク·サーバの間で共有されます。NwkSKey は、通信のメッセージの整合性を確保するとともに、エンドデバイス上のネットワーク·サーバ通信をセキュリティで保護します。

AppSKey は、エンドデバイスとアプリケーション·サーバの間で共有されます。AppSKey は、アプリケーション·データの暗号化と復号化に使用されます。言い換えると、AppSKey は、アプリケーションのペイロードをセキュリティで保護します。この方法により、エンドデバイスから送信されるアプリケーション·データを、ネットワーク·サーバが解釈することができなくなります。

参考資料

3 I-CUBE-LRWAN ミドルウェアの説明

3.1 概要

この I-CUBE-LRWAN パッケージは、STM32 マイクロコントローラ向けの LoRa スタック·ミドルウェアを提供します。このミドルウェアは、次の複数のモジュールに分割されます。

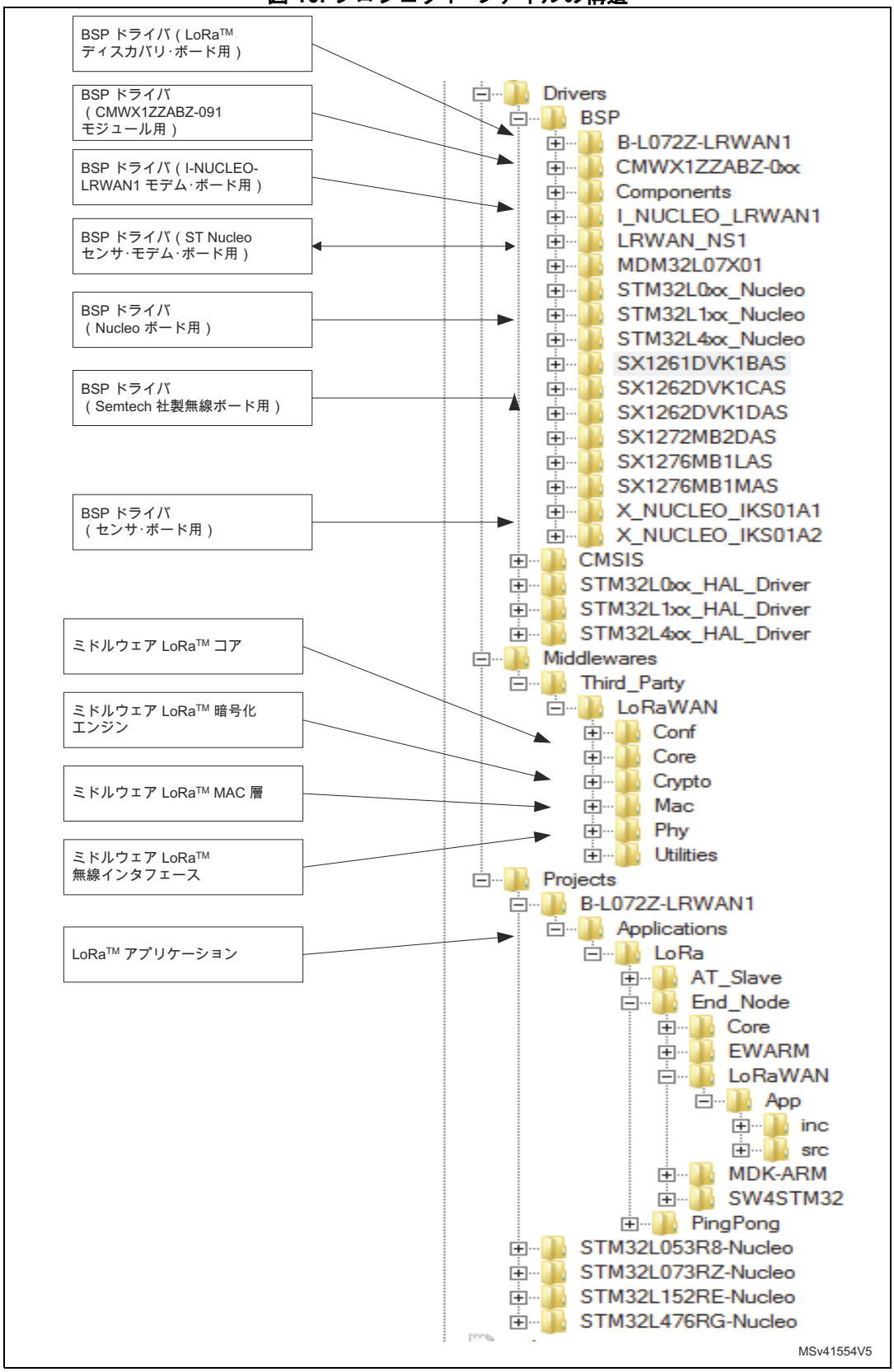
- LoRaMac 層モジュール
- LoRa ユーティリティ·モジュール
- LoRa 暗号化モジュール
- LoRa コア·モジュール

LoRa コア·モジュールは、LoRaMac 層の上に LoRa ステート·マシンを実装します。LoRa スタック·モジュールは、BSP Semtech 社製無線ドライバ·モジュールと接続します。

このミドルウェアは、ソースコード形式で提供され、STM32Cube HAL ドライバに準拠しています。

プロジェクト·ファイルの構造については、図 10 を参照してください。

図 10. プロジェクト·ファイルの構造



I-CUBE-LRWAN パッケージの内容は次のとおりです。

- LoRa スタック･ミドルウェア：
  - LoRaWAN 層
  - LoRa ユーティリティ（電源、キュー、システム時間、時間サーバ、トレース管理など）
  - LoRa ソフトウェア暗号化エンジン
  - LoRa ステート･マシン
- ボード･サポート･パッケージ：
  - Semtech 社製無線ドライバ
  - ST 製センサ･ドライバ
- STM32L0 HAL ドライバ
- LoRa 主要アプリケーション例

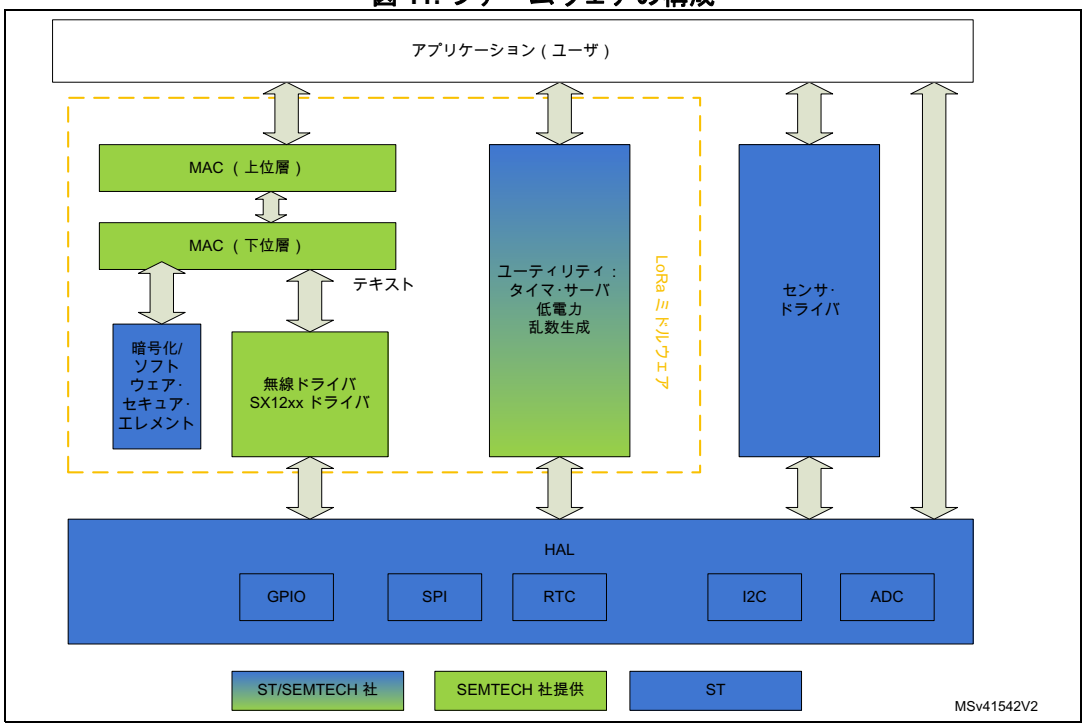
3.2 機能

- LoRaWAN と呼ばれる LoRa Alliance のプロトコル仕様に準拠
- LoRaWAN のクラス A、クラス B、およびクラス C のプロトコル･スタック
- EU 868MHz ISM バンド ETSI 準拠
- EU 433MHz ISM バンド ETSI 準拠
- US 915MHz ISM バンド FCC 準拠
- KR 920Mhz ISM バンド（韓国政府により規定）
- RU 864Mhz ISM バンド（ロシア規制により規定）
- OTAA（over-the-air activation）または ABP（activation-by-personalization）のいずれかによるエンドデバイス･アクティベーション
- アダプティブ･データ･レート（ADR）をサポート
- LoRaWAN 認定試験用アプリケーションを含む
- 省電力に最適化

3.3 アーキテクチャ

図 11 は、I-CUBE-LRWAN アプリケーションのファームウェアの構成を示します。

図 11. ファームウェアの構成



HAL は、Cube API を使用して、アプリケーションに必要なマイクロコントローラ·ハードウェアを駆動します。LoRa ミドルウェアには、LoRa アプリケーションの実行に必要な特定のハードウェアのみが含まれています。

RTC は、省電力モード（STOP モード）でも動作し続ける集中型時間ユニットを提供します。RTC アラームは、タイマ·サーバで管理されている特定のタイミングでシステムをウェイクアップするために使用します。

無線ドライバは、SPI と GPIO ハードウェアを使用して無線を制御します（図 11 を参照）。無線ドライバは、上位レベルのソフトウェアが使用する一連の API も提供します。

LoRa 無線ドライバは Semtech 社によって提供されますが、その API は STM32Cube HAL と接続するために多少変更されています。

無線ドライバは、次の 2 つの部分に分割されています。

- sx1276.c、sx1272.c、および sx126x.c には、無線のみに依存するすべての関数が含まれます。
- sx1276mb1mas.c、sx1276mb1las.c、sx1272mb2das.c、sx1262dvk1das.c、sx1262dvk1cas.c、および sx1262dvk1bas.c には無線ボードに依存するすべての関数が含まれます。

MAC は、802.15.4 モデルを使用して PHY を制御します。MAC は PHY ドライバと接続し、タイマ·サーバを使用して時間指定されたタスクの追加または削除および「無線送信時間」の管理を行います。このアクションは、ETSI で定められているデューティサイクル制限の順守を確保するとともに、MAC ヘッダとペイロードを暗号化する AES 暗号化/復号アルゴリズムを実行します。

LoRa クラス A を制御するステート･マシンは繊細なため、中間レベルのソフトウェア（lora.c）を MAC とアプリケーションの間に挿入しています（図 11 で MAC の「上位層」を参照）。現時点では、API が限られているので、ユーザはアプリケーション･レベルで自由にクラス A ステート･マシンを実装できます。

アプリケーションは、無限ループを中心として構築され、省電力の管理、割込みハンドラ（アラームまたは GPIO）の実行、および実行する必要があるタスクが存在する場合は LoRa クラス A の呼出しを実行します。また、センサの読込みアクセスも実装します。

3.4 ハードウェア関連コンポーネント

3.4.1 無線リセット

マイクロコントローラの GPIO を 1 つ使用して、無線をリセットします。このアクションは、ハードウェア初期化時に 1 回実行します（表 40: LoRa 無線拡張ボードの特性およびセクション 5.1: エンドデバイス･ハードウェアの説明を参照）。

3.4.2 SPI

sx127x または sx126x の無線コマンドとレジスタには、SPI バスを介して 1Mbit/s でアクセスします（表 40 およびセクション 5.1 を参照）。

3.4.3 RTC

RTC カレンダは、32 kHz 外部オシレータに基づいてすべての電力モードで動作するタイマ･エンジンとして使用されます。デフォルトでは、RTC は、毎秒 1024 個のティック（サブセカンド）を生成するようにプログラムされています。RTC は、マイクロコントローラの初回起動時のハードウェア初期化中に 1 回プログラムされます。RTC 出力は、約 48 日周期の 32bit タイマに制限されています。

ユーザがティック間隔を変更する必要がある場合、ティック間隔は 1 ms 未満に維持する必要があることに注意してください。

3.4.4 入力ライン

3.4.4.1 sx127x 割込みライン

sx127x の 4 本の割込みラインは、無線からの割込みの受信専用です（表 40 およびセクション 5.1 を参照）。

DIO0 は、LoRa 無線が要求されたタスクを正常に完了したことを通知するために使用します（TxDone または RxDone）。

DIO1 は、無線が要求されたタスクを正常に完了できなかったことを通知するために使用します（RxTimeout）。

FSK モードでは、FIFO レベルの割込みは、FIFO レベルが事前定義されている閾値に達して、一掃する必要があることを通知します。

DIO2 は、FSK モードで使用され、無線が正常にプリアンブルを検出したことを通知します。

DIO3 は、将来の使用のために予約されています。

注： **LoRaWAN の FSK モードの最も高いデータレートは 50kbit/s です。**

3.4.4.2 sx126x 入力ライン

sx126x インタフェースは、sx127x に比べて簡素化されています。1 つのビジー信号が、無線がビジーでコマンドを処理できないことをマイクロコントローラに通知します。マイクロコントローラは、新しいコマンドを送信する前に、レディ信号がネゲートされていることをポーリングする必要があります。

DIO1 は、1 つのライン割込みとして使用されます。

参考資料

4 I-CUBE-LRWAN ミドルウェア·プログラミングのガイドライン

このセクションでは、LoRaMac 層の API について説明します。物理層は独自仕様なので（セクション 2.1: 概要を参照）、本書の範囲外であり、ブラック·ボックスと見なす必要があります。

4.1 ミドルウェアの初期化

LoRaMac 層の初期化は、'LoRaMacInitialization' 関数を使用して実行します。この関数は、LoRaMac 層のプリアンブル·ランタイム初期化を実行し、MCPS サービスと MLME サービスのコールバック·プリミティブを初期化します（表 4 を参照）。

表 4. ミドルウェア初期化関数

関数	説明
LoRaMacStatus_t LoRaMacInitialization (LoRAMacPrimitives_t *primitives, LoRaMacCallback_t *callback, LoRaMacRegion_t region)	LoRaMac 層モジュールの初期化を実行します（セクション 4.3: ミドルウェア MAC 層のコールバックを参照）。

4.2 ミドルウェア MAC 層の関数

提供されている API は、IEEE802.15.4-2011 で規定されている「プリミティブ」の定義に従っています（セクション 1.2: 参照を参照）。

LoRaMac とのやりとりは、要求/確認と指示/応答のアーキテクチャにより行われます。アプリケーション層が要求し、LoRaMAC 層が確認プリミティブでこれを確認します。逆に、イベントが発生した場合は、LoRaMAC 層がアプリケーション層に指示プリミティブで通知します。

アプリケーション層は、指示に対して応答プリミティブで応答します。したがって、すべての確認/指示は、コールバックを使用して実装されます。

LoRaMAC 層では、MCPS サービス、MLME サービス、および MIB サービスが提供されます。

4.2.1 MCPS サービス

一般に、LoRaMAC 層では、データ送信とデータ受信に MCPS サービスを使用します（表 5 を参照）。

表 5. MCPS サービス関数

関数	説明
LoRaMacStatus_t LoRaMacMcpsRequest (McpsReq_t *mcpsRequest)	Tx データの送信を要求します。

4.2.2 MLME サービス

LoRaMAC 層では、MLME サービスを使用して、LoRaWAN ネットワークを管理します（表 6 を参照）。

表 6. MLME サービス関数

関数	説明
LoRaMacStatus_t LoRaMacMlmeRequest (MlmeReq_t *mlmeRequest)	Join 要求またはリンク·チェック要求の生成に使用します。

4.2.3 MIB サービス

MIB には、重要なランタイム情報（MIB_NETWORK_ACTIVATION、MIB_NET_ID など）が格納され、LoRaMAC 層の設定（MIB_ADR、MIB_APP_KEY など）が保持されます。表 7 に示す API が提供されています。

表 7. MIB サービス関数

関数	説明
LoRaMacStatus_t LoRaMacMibSetRequestConfirm (MibRequestConfirm_t *mibSet)	LoRaMac 層の属性を設定します。
LoRaMacStatus_t LoRaMacMibGetRequestConfirm (MibRequestConfirm_t *mibGet)	LoRaMac 層の属性を取得します。

4.3 ミドルウェア MAC 層のコールバック

LoRaMac ユーザ·イベント関数のプリミティブとコールバック関数の説明については、セクション 4.1: ミドルウェアの初期化を参照してください。

4.3.1 MCPS

一般に、LoRaMAC 層では、データ送信とデータ受信に MCPS サービスを使用します（表 8 を参照）。

表 8. MCPS プリミティブ

関数	説明
void (*MacMcpsConfirm) (McpsConfirm_t *McpsConfirm)	呼び出されるコールバックのイベント関数プリミティブ。アプリケーションによって実装されます。McpsRequest に対する応答。
Void (*MacMcpsIndication) (McpsIndication_t *McpsIndication)	呼び出されるコールバックのイベント関数プリミティブ。アプリケーションによって実装されます。受信したパケットが使用可能なことをアプリケーションに通知します。

4.3.2 MLME

LoRaMAC 層では、MLME サービスを使用して、LoRaWAN ネットワークを管理します（表 9 を参照）。

表 9. MLME プリミティブ

関数	説明
void (*MacMlmeConfirm) (MlmeConfirm_t *MlmeConfirm)	呼び出されるコールバックのイベント関数プリミティブ。アプリケーションによって実装されます。

4.3.3 MIB

N/A

4.3.4 バッテリ・レベル

LoRaMAC 層には、バッテリ・レベル測定サービスが必要です（表 10 を参照）。

表 10. バッテリ・レベル関数

関数	説明
uint8_t HW_GetBatteryLevel (void)	測定したバッテリ・レベルを取得します。

4.4 ミドルウェア MAC 層のタイマ

4.4.1 遅延 Rx ウィンドウ

セクション 2.2.2: エンドデバイスのクラスを参照してください。遅延 Rx 関数については、表 11 を参照してください。

表 11. 遅延 Rx 関数

関数	説明
void OnRxWindow1TimerEvent (void)	RxDelay1（ReceiveDelayX - RADIO_WAKEUP_TIME）を設定します。
void OnRxWindow2TimerEvent (void)	RxDelay2 を設定します。

4.4.2 Tx フレーム送信の遅延

表 12 に、Tx フレーム送信の遅延の関数を示します。

表 12. Tx フレーム送信の遅延の関数

関数	説明
void OnTxDelayedTimerEvent (void)	Tx フレーム送信のタイマを設定します。

4.4.3 Rx フレームの遅延

表 13 に、Rx フレームの遅延の関数を示します。

表 13. Rx フレームの遅延の関数

関数	説明
void OnAckTimeoutTimerEvent (void)	受信したフレーム確認応答のタイムアウトを設定します。

4.5 ミドルウェアユーティリティの関数

4.5.1 タイマ·サーバの API の説明

タイマ·サーバが提供されているため、ユーザは時間指定されたタスクの実行を要求できます。ハードウェア·タイマは RTC に基づいているため、省電力モードであっても時間は常にカウントされています。

タイマ·サーバは、ユーザと LoRa スタックに信頼できるクロックを提供します。ユーザは、アプリケーションに必要な数だけ、タイマを要求できます。

表 14 に示す 4 つの API が提供されています。

表 14. タイマ·サーバ関数

関数	説明
void TimerInit (TimerEvent_t *obj, void (*callback) (void))	タイマを初期化し、タイマ経過時に呼び出すコールバック関数を関連付けます。
void TimerSetValue (TimerEvent_t *obj, uint32_t value)	タイマのタイムアウト値をミリ秒単位で指定します。
void TimerStart (TimerEvent_t *obj)	タイマを開始します。
void TimerStop (TimerEvent_t *obj)	タイマを停止します。

タイマ·サーバは、Middlewares\Third_Party\Lora\Utilities にあります。

4.5.2 省電力関数

表 15 に示す API を使用して、マイクロコントローラの省電力モードを管理できます。

表 15. 省電力関数

関数	説明
void LPM_EnterLowPower (void)	システムを省電力モードに移行できます。
void LPM_EnterSleepMode (void)	省電力 SLEEP モードに移行します。
void LPM_ExitSleepMode (void)	省電力 SLEEPモードを終了します。
void LPM_EnterStopMode (void)	省電力 STOP モードに移行します。
void LPM_ExitStopMode (void)	省電力 STOP モードを終了します。
void LPM_EnterOffMode (void)	省電力 OFF モードに移行します。
void LPM_ExitOffMode(void)	省電力 OFF モードを終了します。

表 15. 省電力関数 （続き）

関数	説明
LPM_GetMode_t LPM_GetMode (void)	選択されている省電力モードを返します。
void LPM_SetStopMode (LPM_Id_t id, LPM_SetMode_t mode)	STOP モードを有効にするか、または SLEEP モードを要求するために STOP モードを無効にできます。
void LPM_SetOffMode (LPM_Id_t id, LPM_SetMode_t mode)	STOP モードを有効にするか、または OFF モードを有効にできます。

4.5.3 システム時間関数

マイクロコントローラの時間は、マイクロコントローラ·リセットを基準にしています。SysTime は、Unix エポック·タイムを記録できます。

表 16 に示す API を使用して、マイクロコントローラのシステム時間を管理できます。

表 16. システム時間関数

関数	説明
void SysTimeSet (SysTime_t sysTime)	入力された Unix エポック·タイム（秒およびサブセカンド）に基づいて、マイクロコントローラの時間との差が BACK_UP レジスタに格納されます（STANDBY モードでも保持されます）。 システム時間は、1970 年 1 月 1 日月曜日から始まる Unix エポック·タイムを基準としています。
SysTime_t SysTimeGet (void)	現在のシステム時間を取得します。 システム時間は、1970 年 1 月 1 日月曜日から始まる Unix エポック·タイムを基準としています。
uint32_t SysTimeMkTime (const struct tm* localtime)	ローカル時間をエポック·タイムに変換します（以下の注を参照）。
void SysTimeLocalTime (const uint32_t timestamp, struct tm *localtime)	ローカル時間をエポック·タイムに変換し、ローカル時間に変換します（以下の注を参照）。

注：

SysTimeMkTime と SysTimeLocalTime は、time.h インタフェースの指定に従ってエポック·タイムを tm 構造体に変換する目的でも提供されています。
UNIX タイムをローカル時間に変換するには、タイム·ゾーンを加算し、うるう秒を減算する必要があります。2018 年にはうるう秒として 18 秒を減算する必要があります。パリの夏時間は、グリニッジ標準時と 2 時間の時差があり、時間が設定されていると仮定すると、次のようにしてローカル時間を端末に表示できます。

```
{
SysTime_t UnixEpoch = SysTimeGet();

struct tm localtime;

UnixEpoch.Seconds-=18; /*removing leap seconds*/

UnixEpoch.Seconds+=3600*2; /*adding 2 hours*/

SysTimeLocalTime(UnixEpoch.Seconds,  & localtime);
```

```
PRINTF ("it's %02dh%02dm%02ds on %02d/%02d/%04d\n\r",
        localtime.tm_hour, localtime.tm_min, localtime.tm_sec,
        localtime.tm_mday, localtime.tm_mon+1, localtime.tm_year + 1900);
}
```

4.5.4 トレース関数

トレース·モジュールを使用すると、DMA を使用して COM ポートにデータを出力できます。

表 17 に示す API を使用して、トレース関数を管理できます。

表 17. トレース関数

関数	説明
void TraceInit (void)	TraceInit は、アプリケーションを初期化する際に呼び出す必要があります。これは、COM または VCOM ハードウェアを DMA モードで初期化し、DMA 送信完了時に処理されるようにコールバックを登録します。
int32_t TraceSend (const char *strFormat,...)	文字列形式をバッファとバッファ長に変換し、十分なスペースが残っている場合は循環キューに追加します。十分なスペースが残っていてキューに追加した場合は 0 を返し、十分なスペースが残っていなかった場合は -1 を返します。

注： **リアルタイム制約が適用されない場合、通常はアプリケーションを初期化する際に、ポーリング·モードで使用できます。**

```
#define PPRINTF(...)    do{} while (0!= TraceSend (__VA_ARGS__)) //ポーリング·モード
```

リアルタイム·モードで使用できます。この場合、循環キューに十分なスペースが残っていなければ、文字列は追加されず、COM ポートに出力されません。

```
#define PRINTF(...)    do {TraceSend (__VA_ARGS__);} while(0)
```

十分なスペースが残っていないことが多い場合、utilities_conf.h でバッファ長を増やすことができます。

```
#define DBG_TRACE_MSG_QUEUE_SIZE 256
```

4.5.5 キュー関数

キュー·モジュールは、バッファを循環キューとして管理する一連のサービスを提供します。

表 18 に示す API を使用して、循環キュー·バッファを管理できます。

表 18. キュー関数

関数	説明
int CircularQueue_Init (queue_t *q, uint8_t* queueBuffer, uint32_t queueSize, uint16_t elementSize, uint8_t optionFlags)	循環バッファを初期化します。
uint8_t* CircularQueue_Add (queue_t *q, uint8_t* x, uint16_t elementSize, uint32_t nbElements)	要素を循環バッファに追加します。
uint8_t* CircularQueue_Remove (queue_t *q, uint16_t* elementSize)	循環バッファから要素を削除します。
uint8_t* CircularQueue_Sense (queue_t *q, uint16_t* elementSize)	循環バッファが空でないかどうかを検知します。空ではない場合、バッファのアドレスおよびその長さを要素サイズを使用して返します。

注： **キューには要素が充填されます。各要素は、バッファ長フィールド（2 バイト）とバッファで構成されます。1 つの要素がキューの最後に収まらないほど大きい場合は、2 つの要素に分割されます。**

4.6 セキュア·エレメントのエミュレート

デフォルトでは、提案されているハードウェア·プラットフォームにセキュア·エレメント·デバイスは実装されていません。したがって、このセキュア·エレメント·デバイスは、ソフトウェアでエミュレートします。

図 12 に、LoRaMacCrypto モジュールの構成を示します。

図 12. LoRaMacCrypto モジュールの構成

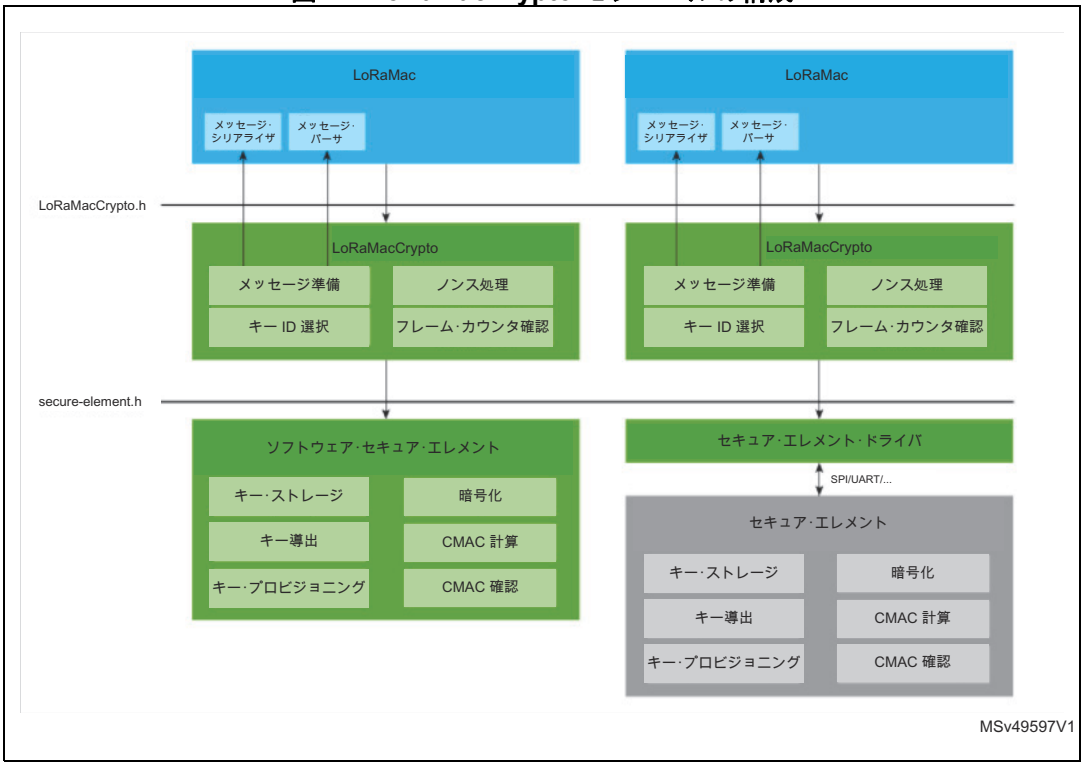


表 19 に示す API を使用して、エミュレートするセキュア·エレメントを管理できます。

表 19. セキュア·エレメント関数

関数	説明
SecureElementStatus_t SecureElementInit (EventNvmCtxChanged seNvmCtxChanged)	セキュア·エレメント·ドライバを初期化します。 不揮発性コンテキストを格納する必要がある場合に呼び出されるコールバック関数です。
SecureElementStatus_t SecureElementRestoreNvmCtx (void* seNvmCtx)	渡されたポインタで指定される内部不揮発性コンテキストを、不揮発性モジュール·コンテキストに復元します。
void* SecureElementGetNvmCtx (size_t* seNvmCtxSize)	不揮発性コンテキストが格納されるアドレスを要求します。

表 19. セキュア･エレメント関数 （続き）

関数	説明
SecureElementStatus_t SecureElementSetKey (KeyIdentifier_t keyID, uint8_t* key)	キーを設定します。
SecureElementStatus_t SecureElementComputeAesCmac (uint8_t* buffer, uint16_t size, KeyIdentifier_t keyID, uint32_t* cmac)	CMAC を計算します。 KeyID を使用して､使用する AES キーを決定できます。
SecureElementStatus_t SecureElementVerifyAesCmac (uint8_t* buffer, uint16_t size, uint32_t expectedCmac, KeyIdentifier_t keyID)	CMAC を計算して、予想される CMAC と比較します。 KeyID を使用して､使用する AES キーを決定できます。
SecureElementStatus_t SecureElementAesEncrypt (uint8_t* buffer, uint16_t size, KeyIdentifier_t keyID, uint8_t* encBuffer)	バッファを暗号化します。 KeyID を使用して､使用する AES キーを決定できます。
SecureElementStatus_t SecureElementDeriveAndStoreKey (Version_t version, uint8_t* input, KeyIdentifier_t rootKeyID, KeyIdentifier_t targetKeyID)	キーを派生させて格納します。キーの導出は、LoRaWAN version に依存します。rootKeyID を使用して、派生を実行するルート･キーを指定できます。

4.7 ミドルウェア End_Node アプリケーション関数

MAC とのやりとりには、MAC インタフェース･ファイル ‘LoRaMac.h’ を使用します。

標準モード

標準モードでは、インタフェース･ファイル（図 11 の MAC 上位層を参照）が提供されるため、ユーザは LoRa ステート･マシンを気にせずに開始できます。インタフェース･ファイルは、Middlewares\Third_Party\Lora\Core\lora.c にあります。

インタフェース･ファイルは、以下を実装します。

- LoRaMAC サービスへのアクセスを可能にする一連の API
- LoRa 認定試験ケース（アプリケーション層からは見えません）

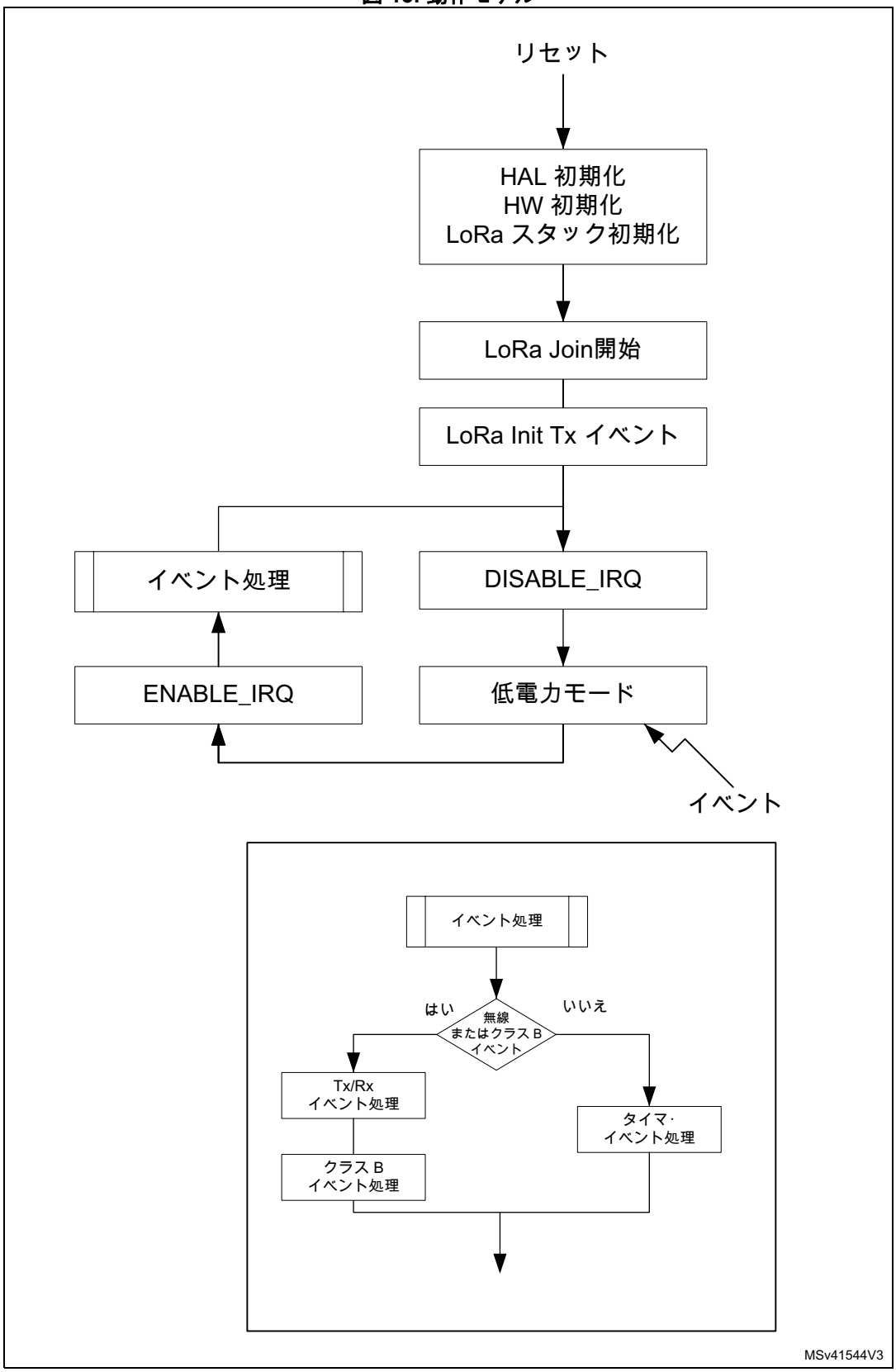
アドバンスト･モード

このモードでは、ユーザは、ユーザ･ファイルに MAC を含めることによって MAC 層に直接アクセスします。

動作モデル

この LoRa End_Node に提案されている動作モデル（図 13 を参照）は､「時間駆動」を含む「イベント駆動」のパラダイムに基づいています。LoRa システムの動作は、タイマ･イベントまたは無線イベントおよびガード･トランジションのいずれかによってトリガされます。

図 13. 動作モデル


リセット
HAL 初期化
HW 初期化
LoRa スタック初期化
LoRa Join開始
LoRa Init Tx イベント
イベント処理
DISABLE_IRQ
ENABLE_IRQ
低電力モード
イベント
イベント処理
はい
無線
またはクラス B
イベント
いいえ
Tx/Rx
イベント処理
タイマ·
イベント処理
クラス B
イベント処理
MSv41544V3

LoRa システムの状態遷移

図 14 に、Lora End_Node システムの状態遷移を示します。

リセット後のシステム初期化の完了後、LoRa End_Node システムは ‘**Init**’ 状態に移行します。

“over_the_air_activation (OTAA)” を使用する場合、LoRa End_Node システムは Join ネットワーク要求を実行して、‘**Sleep**’ 状態に移行します。

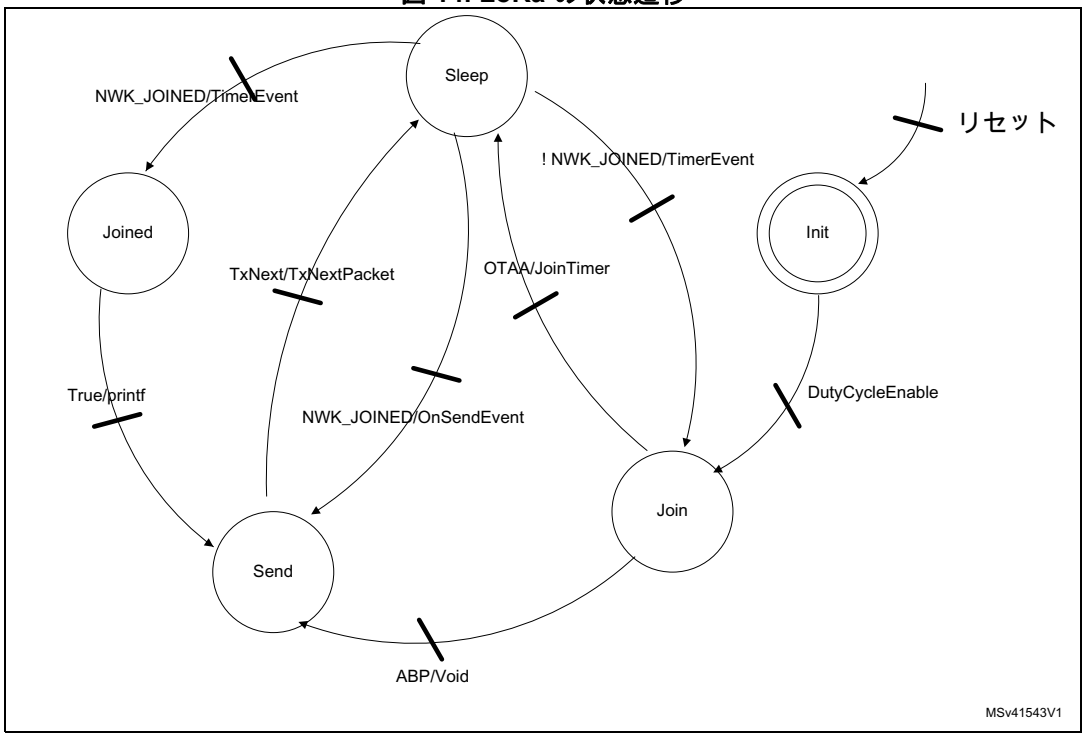
“activation by personalization (ABP)” を使用する場合、ネットワークにすでに参加しているので、LoRa End_Node システムは直接 ‘**Send**’ 状態に移行します。

‘**Sleep**’ 状態で、“TimerEvent” の発生時にエンドデバイスがすでにネットワークに参加している場合、LoRa End_Node システムは一時的に ‘**Joined**’ 状態に移行した後、‘**Send**’ 状態に移行します。

‘**Sleep**’ 状態で、“OnSendEvent” の発生時にエンドデバイスがすでにネットワークに参加している場合、LoRa End_Node システムは‘**Send**’ 状態に移行します。

‘**Send**’ 状態で、LoRa End_Node システムは、スケジュールされている次の送信パケットに対応する ‘onSendEvent’ を待機するため、‘**Sleep**’ 状態に戻ります。

図 14. LoRa の状態遷移



LoRa クラス B システムの状態遷移

図 15 に、LoRa クラス B モード·エンドデバイスのシステム状態遷移を示します。

1. エンドデバイスは、クラス B モードへの切替え要求を実行する前に、Join 状態にいる必要があります（図 13 を参照）。
2. クラス A モードからクラス B モードへの切替えの決定は、常にエンドデバイスのアプリケーション層で行われます。決定がネットワーク側で行われる場合、アプリケーション·サーバはエンドデバイスのクラス A アップリンクを使用してダウンリンク·フレームをアプリケーション層に返信する必要があります。

MLME Beacon_Acquisition_req が発生すると、エンドデバイスの LoRa クラス B システムの状態は BEACON_STATE_ACQUISITION に移行します。

LoRa エンドデバイスは、ビーコンの取得を開始します。MAC 層が RxBeacon 関数内で正常にビーコンを受信すると、BEACON_STATE_LOCKED 状態に移行します。

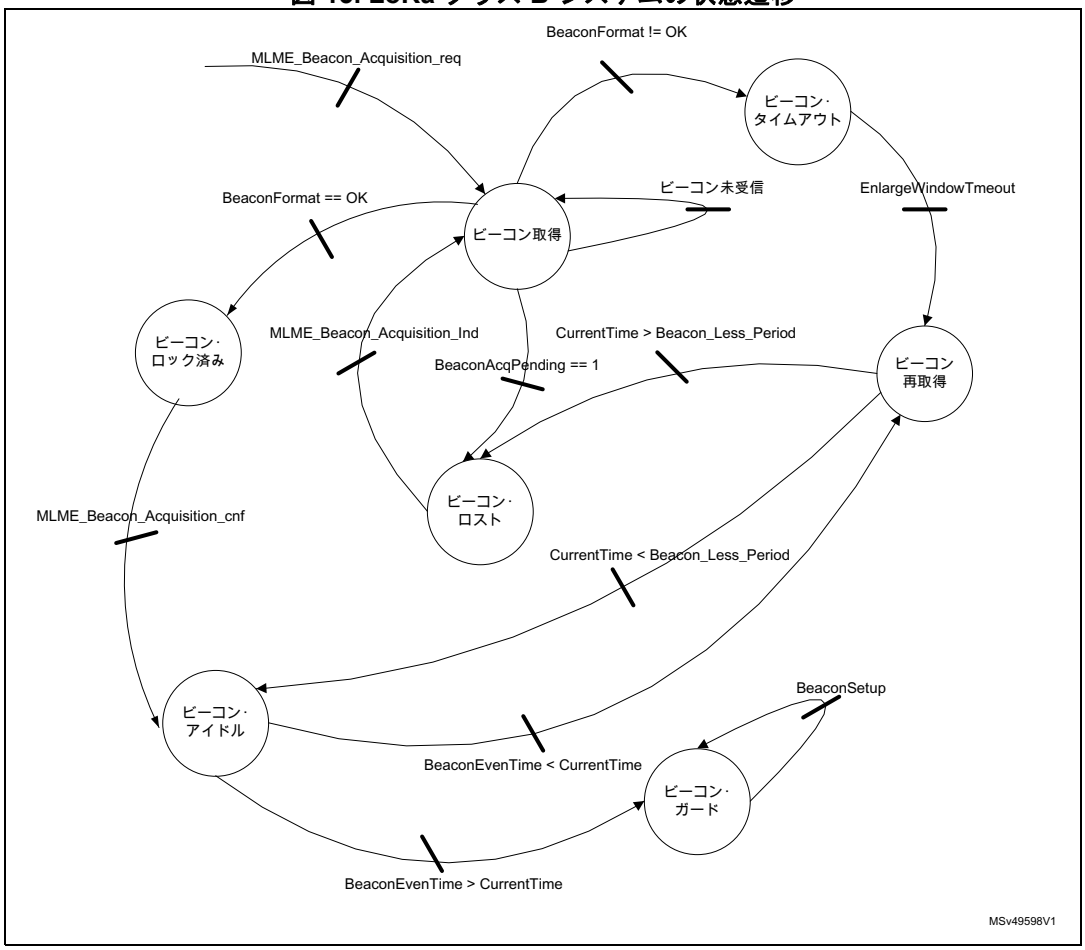
LoRa エンドデバイスがビーコンを受信しています。取得は保留中ではなくなり、MAC 層は BEACON_STATE_IDLE に移行します。

BEACON_STATE_IDLE で、MAC 層は BeaconEventTime をエンドデバイスの現在の時間と比較します。MAC 層は、beaconEventTime がエンドデバイスの現在の時間より前であれば BEACON_STATE_REACQUISITION に移行し、そうでなければ BEACON_STATE_GUARD に移行します。そして、新しいビーコン取得を実行します。

MAC 層がビーコンを見つけられない場合、ステート·マシンは BEACON_STATE_ACQUISITION 状態に留まります。この状態では、それまで取得が保留中だったことを検出して、次の BEACON_STATE_LOST 状態に移行します。

MAC 層が不正なビーコン·フォーマットを受信した場合は、BEACON_STATE_TIMEOUT に移行する必要があります。ここでウィンドウのタイムアウト時間を延ばし、次のビーコンを受信する可能性を増やして、BEACON_STATE_REACQUISITION に移行します。

図 15. LoRa クラス B システムの状態遷移



4.7.1 LoRa End_Node 初期化

表 20. LoRa クラス A 初期化関数

関数	説明
void lora_Init (LoRaMainCallback_t *callbacks, LoRaParam_t* LoRaParamInit)	LoRa クラス A 有限ステート･マシンを初期化します。

4.7.2 LoRa End_Node Join 要求エントリ･ポイント

表 21. LoRa End_Node Join 要求エントリ･ポイント

関数	説明
void lora_Join (void)	OTAA モードまたは ABP モードでネットワークに Join 要求を発行します（Join モードはコンパイル時に定義する必要があります）。

4.7.3 LoRa End_Node Tx 開始

表 22. LoRa End_Node Tx 開始

関数	説明
void loraStartTx (TxEventType_t EventType)	EventType パラメータが TX_ON_TIMER に等しい場合、OnTxTimerEvent の発生を開始します。ユーザはここに独自のコードを自由に実装できます。

4.7.4 End_Node の Join ステータス要求

表 23. End_Node の Join ステータス

関数	説明
LoraFlagStatus LORA_JoinStatus (void)	End_Node のアクティベーション･タイプ（ACTIVATION_TYPE_NONE、ACTIVATION_TYPE_ABP、ACTIVATION_TYPE_OTAA）をチェックします。

4.7.5 アップリンク･フレーム送信

表 24. アップリンク･フレーム送信

関数	説明
bool LORA_send (lora_AppData_t* AppData,LoraConfirm_t IsTxConfirmed)	アップリンク･フレームを送信します。このフレームは、unconfirmed の空のフレームまたは unconfirmed/confirmed のペイロード･フレームのいずれかのフレームになります。

4.7.6 現在のネットワーク時間の要求

表 25. 現在のネットワーク時間

関数	説明
LoraErrorStatus LORA_DeviceTimeReq (void)	エンドデバイスがネットワークに現在のネットワーク時間を要求します(クラス B モードでビーコン検出時間を短縮するのに役立ちます)。 注：LoRaWAN バージョン 1.0.3 以降で BeaconTimeReq の代わりに使用します。

4.7.7 4.7.7 次のビーコン·タイミングの要求

表 26. 次のビーコン·タイミング

関数	説明
LoraErrorStatus LORA_BeaconTimeReq (void)	エンドデバイスがネットワークに次のビーコン·タイミングを要求します(クラス B モードでビーコン検出時間を短縮するのに役立ちます)。 注：このコマンドは LoRaWAN V1.0.3 では非推奨となっています。

4.7.8 クラス切替え要求

表 27. クラス切替え要求

関数	説明
LoraErrorStatus LORA_RequestClass (DeviceClass_t newClass)	エンドデバイスに現在のクラスから新しいクラス（A、B、または C）への切替えを要求します。

4.7.9 エンドデバイスの現在のクラスの取得

表 28. エンドデバイスの現在のクラスの取得

関数	説明
void LORA_GetCurrentClass (DeviceClass_t *currentClass)	現在実行中のクラス（A、B、または C）を要求します。

4.7.10 ビーコン取得要求

表 29. ビーコン取得要求

関数	説明
LoraErrorStatus LORA_BeaconReq (void)	ビーコン·スロットの取得を要求します。

4.7.11 ユニキャスト ping スロット情報周期の送信

表 30. ユニキャスト ping スロット周期

関数	説明
LoraErrorStatus LORA_PingSlotReq (void)	サーバにユニキャスト ping スロット情報周期を送信します。

4.8 LIB End_Node アプリケーション·コールバック

4.8.1 現在のバッテリ·レベル

表 31 に現在のバッテリ·レベル関数を示します。

表 31. 現在のバッテリ·レベル関数

関数	説明
uint8_t HW_GetBatteryLevel (void)	バッテリ·レベルを取得します。

4.8.2 現在の温度レベル

表 32. 現在の温度レベル関数

関数	説明
uint16_t HW_GetTemperatureLevel (void)	チップセットの現在の温度（摂氏温度）を q7.8 フォーマットで取得します。

4.8.3 ボード固有 ID

表 33 に、ボード固有 ID 関数を示します。

表 33. ボード固有 ID 関数

関数	説明
void HW_GetUniqueId (uint8_t *id)	固有 ID を取得します。

4.8.4 ボード·ランダム·シード

表 34 に、ボード·ランダム·シード関数を示します。

表 34. ボード·ランダム·シード関数

関数	説明
uint32_t HW_GetRandomSeed (void)	ランダム·シード値を取得します。

4.8.5 Rx フレーム作成

表 35 に、Rx フレーム作成関数を示します。

表 35. Rx フレーム作成関数

関数	説明
void LoraRxData (lora_AppData_t *AppData)	受信フレーム·アプリケーションを処理します。 ユーザはここに独自のコードを自由に実装できます。

4.8.6 LoRa Joined 通知

表 36. LoRa Joined 通知関数

関数	説明
void LORA_HasJoined (void)	End_Node が Joined の状態になったことをアプリケーションに通知します。

4.8.7 End_Node クラス·モード変更確認

表 37. End_Node クラス·モード変更確認関数

関数	説明
void LORA_ConfirmClass (DeviceClass_t Class)	End_Node がデバイス·クラスを変更したことをアプリケーションに通知します。

4.8.8 ダミー·アップリンク·フレーム送信

表 38. ダミー·アップリンク·フレーム送信関数

関数	説明
void LORA_TxNeeded (void)	フレーム送信をアプリケーションに要求します。

5 サンプルプログラムの説明

5.1 エンドデバイス･ハードウェアの説明

アプリケーション層、Mac 層、および PHY ドライバは 1 つのマイクロコントローラ上に実装されます。End_Node アプリケーションは、このハードウェア･ソリューションに実装されています（セクション 5.4 を参照）。

I-CUBE-LRWAN は、次のような複数のプラットフォーム上で動作します。

- STM32 Nucleo プラットフォーム（LoRa 無線拡張ボード搭載）
- B-L072Z-LRWAN1 ディスカバリ･ボード（LoRa 拡張ボードは不要）

オプションで ST 製 X-NUCLEO-IKS01A1 センサ拡張ボードを Nucleo ボードとディスカバリ･ボードに追加できます。表 39 にサポートされている Nucleo ベースのハードウェアを示します。

表 39. サポートされている Nucleo ベースのハードウェア

LoRa 無線拡張ボード/ Nucleo ボード	SX1276MB1MAS	SX1276MB1LAS	SX1272MB2DAS
NUCLEO-L053R8	サポート	サポート	サポート
NUCLEO-L073RZ	サポート	サポート	サポート(1)
NUCLEO-L152RE	サポート	サポート	サポート
NUCLEO-L476RG	サポート	サポート	サポート

1. この組合せは P-NUCLEO-LRWAN1 キットとして市販されています。

I-CUBE-LRWAN 拡張パッケージは、サポートされている他のデバイスや開発ボードに合わせて簡単に調整できます。

LoRa 無線拡張ボードの主な特性を表 40 に示します。

表 40. LoRa 無線拡張ボードの特性

ボード	特性
SX1276MB1MAS	868 MHz （HF）/14 dBm および 433 MHz （LF）/14 dBm
SX1276MB1LAS	915 MHz （HF）/20 dBm および 433 MHz （LF）/14 dBm
SX1272MB2DAS	915 MHz および 868 MHz/14 dBm
SX1261DVK1BAS	E406V03A sx1261、14 dBm、868 MHz、XTAL
SX1262DVK1CAS	E428V03A sx1262、22 dBm、915 MHz、XTAL
SX1262DVK1DAS	E449V01A sx1262、22 dBm、860 ~ 930 MHz、TCXO

無線インタフェースは、以下のとおりです。

- 無線レジスタには SPI 経由でアクセスします。
- DIO マッピングは無線に依存します（セクション 3.4.4 を参照）。
- 無線のリセットには、マイクロコントローラの 1 つの GPIO を使用します。
- マイクロコントローラの 1 つのピンでアンテナ･スイッチを制御し、Rx モードまたは Tx モードに設定します。

ハードウェア･マッピングは、Projects\<platform>\Applications\LoRa\<App_Type>\Core\inc にあるハードウェア設定ファイルに記述されています。

ここで、<platform> には STM32L053R8-Nucleo、STM32L073RZ-Nucleo、STM32L152RE-Nucleo、STM32L476RG-Nucleo、B-L072Z-LRWAN1 （村田製作所製モデム･デバイス）が入ります。<Target> には STML0xx が入ります。<App_Type> には AT_Master、End_Node、PingPong、AT_Slave が入ります。

割込み

表 41 に､Cortex システム･プロセッサの例外および STM32L0 シリーズの LoRa アプリケーション固有の割込み（IRQ）に適用される割込み優先レベルを示します。

表 41. STM32L0xx の IRQ 優先度

割込み名	割込み優先度	サブ優先度
RTC	0	NA
EXTI2_3	0	NA
EXTI4_15	0	NA

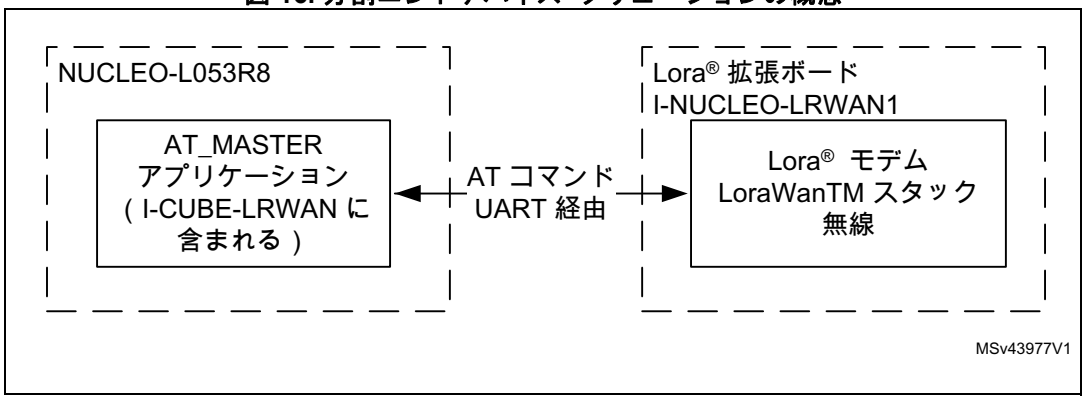
5.2 分割エンドデバイス･ハードウェアの説明（2 チップ･ソリューション）

アプリケーション層、Mac 層、および PHY ドライバは分割されます。LoRa End_Node は、LoRa モデムとホスト･コントローラで構成されます。LoRa モデムは Lora スタック（Mac 層と物理層）を実行し、アプリケーション層を実装している Lora ホストによって制御されます。

NUCLEO ボード上で Lora ホストを実装する AT_Master アプリケーションは、AT_Slave アプリケーションと互換性があります（セクション 5.6 を参照）。AT_Slave アプリケーションは､CMWX1ZZABZ-091 LoRa モジュール（村田製作所製）上でモデムを実現します。AT_Master アプリケーションは、USI 社製 WM-SG-SM-42 LPWAN モジュール搭載の I-NUCLEO-LRWAN1 拡張ボードおよび **P-NUCLEO-LRWAN3 に含まれる RiSiNGHF 社製モデム RHF0M003 搭載の LRWAN_NS1 拡張ボード**（セクション 5.7 を参照）との互換性もあります。

この分割ソリューションを使用すると、LoRaWAN スタックのリアルタイム要件に関する制約を受けずに、アプリケーション層を設計できます。

図 16. 分割エンドデバイス･ソリューションの概念



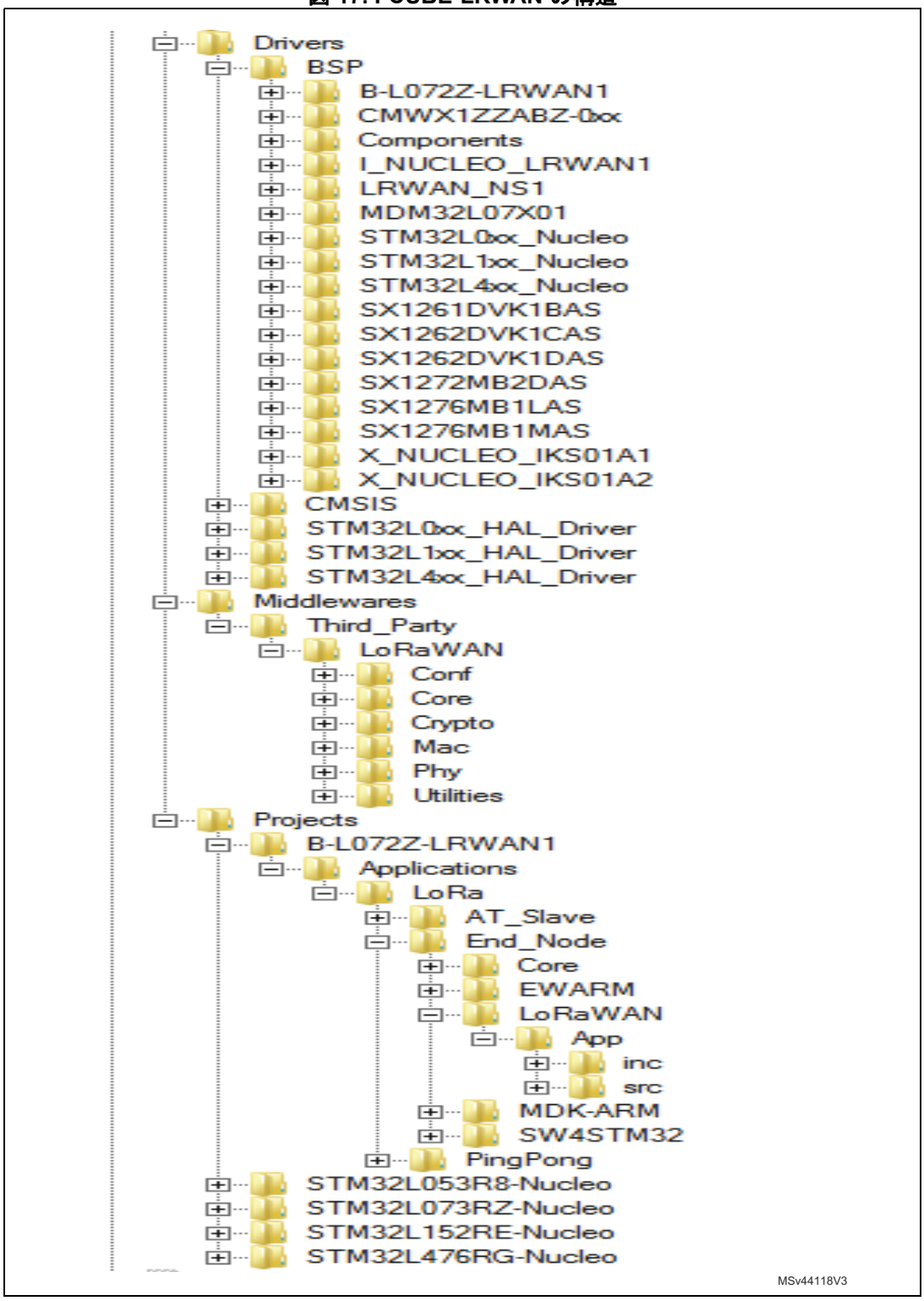
LoRa モデムと LoRa ホストとの間のインタフェースは UART で、AT コマンドのやりとりが行われます。

参考資料

5.3 パッケージの説明

I-CUBE-LRWAN を解凍すると、図 17 に示す構造に展開されます。

図 17. I-CUBE-LRWAN の構造



I-CUBE-LRWAN パッケージには、End_Node、PingPong、AT_Slave、および AT_Master の 4 つのアプリケーションが格納されています。各アプリケーションは、MDK-ARM、IAR、および SW4STM32 の 3 つのツールチェーンで利用できます。

5.4 End_Node アプリケーション

このアプリケーションは、I2C 経由でセンサから温度、湿度、および大気圧を読み取ります。マイクロコントローラは、バッテリ·レベルを計算するために、V_{REFINT} を通じて供給電圧を測定します。これらの 4 つのデータ（温度、湿度、大気圧、およびバッテリ·レベル）は、（EU868設定の場合）868 MHz でクラス A の LoRa 無線を使用して、定期的に LoRa ネットワークに送信されます。

LoRa End_Node プロジェクトを起動するには、IDE 環境で \Projects\<target>\Applications\LoRa\End_Node に移動して、好みのツールチェーン·フォルダを選択する必要があります。その後で、適切なターゲット·ボードから LoRa プロジェクトを選択します。

5.4.1 アクティベーション方法とキー

ネットワーク上でデバイスを有効化するには、OTAA または ABP の 2 つのアクティベーション方法があります。

ファイル \Projects\<target>\Applications\LoRa\End_Node\LoRaWAN\App\inc\Commissioning.h には、デバイスのアクティベーションに関連したデータがすべて集められています。選択した方法とコミッショニング·データが仮想 COM ポート（VCP）に出力され、端末に表示されます。

5.4.2 デバッグ·スイッチ

デバッグ·モードとトレース·モードの両方または一方を有効にするには、\Projects\Multi\Applications\LoRa\End_Node\inc\hw_conf.h を開いて、次の行をアンコメントする必要があります。

```
#define DEBUG
```

デバッグ·モードでは、DBG_GPIO_SET と DBG_GPIO_RST の両マクロが有効になり、また、マイクロコントローラが省電力モードに移行してもデバッガ·モードが有効になります。

トレース·モードでは、VERBOSE_LEVEL_0（トレースは無効）、VERBOSE_LEVEL_1（関数トレースが有効）、および VERBOSE_LEVEL_2 （デバッグ·トレースが有効）の 3 つのトレース·レベルを使用できます。

\Projects\<platform>\Applications\LoRa\<App>\LoRaWAN\App\inc\utilities_conf.h を開いて、選択するトレース·レベルを設定する必要があります。

注： **真の省電力を有効にするには、上記の “#define DEBUG” をコメントアウトする必要があります。**

5.4.3 センサ·スイッチ

センサ拡張ボードが接続されていない場合は、\Projects\<target>\Applications\LoRa\End_Node\LoRaWAN\App\inc\hw_conf.h で #define SENSOR_ENBALED をコメントアウトする必要があります。

表 42 に、アプリケーション設定の主なオプションの一覧を示します。

表 42. アプリケーション設定のスイッチ･オプション

プロジェクト	スイッチ･オプション	定義	場所
LoRa スタック	OVER_THE_AIR_ACTIVATION	アプリケーションは OTAA 手順を使用	Commissioning.h
	STATIC_DEVICE_EUI	静的または動的なエンドデバイス ID	Commissioning.h
	LORAMAC_CLASSB_ENABLED	関連するコードをクラス B モードでコンパイル	コンパイラ･オプション設定
	USE_DEVICE_TIMING	"LORA_DeviceTimeReq ()" または "LORA_BeaconTimeReq (void)" のいずれかをインクルード	lora.c
	STATIC_DEVICE_ADDRESS	静的または動的なエンドデバイス･アドレス	Commissioning.h
	REGION_EU868	バンド選択を有効化	コンパイラ･オプション設定
	REGION_EU433		
	REGION_US915		
	REGION_AS923		
	REGION_AU915		
	REGION_CN470		
	REGION_CN779		
	REGION_IN865		
	REGION_RU864		
	REGION_KR920		
センサ	DEBUG	‘LED on/off’ を有効化	hw_conf.h
	VERBOSE_LEVEL	トレース･レベルを有効化	utilities_conf.h
	SENSOR_ENABLED	センサ･ボードの呼出しを有効化	hw_conf.h

注： **最大許容ペイロード長は、地域と選択したデータ･レートに依存します。したがって、それらのパラメータに従って、ペイロード･フォーマットを慎重に設計する必要があります。**

5.5 PingPong アプリケーションの説明

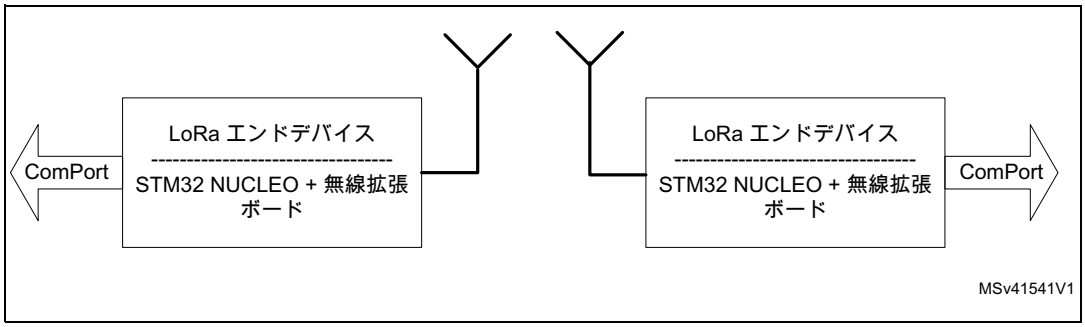
このアプリケーションは、2 つの LoRa エンドデバイス間の単純な Rx/Tx RF リンクです。デフォルトでは、各 LoRa エンドデバイスがマスタとして開始し、‘Ping’ メッセージを送信して応答を待機します。‘Ping’ メッセージを最初に受信した LoRa エンドデバイスがスレーブになり、‘Pong’ メッセージでマスタに応答します。ここから、PingPong が始まります。

PingPong プロジェクトを起動するには、\Projects\<platform>\Applications\LoRa\PingPong フォルダに移動して、LoRa End_Node プロジェクトと同じ手順で好みのツールチェーンを起動します。

ハードウェアとソフトウェアのセットアップ環境

STM32LXxx-NUCLEO をセットアップするには、NUCLEO（または B-L072Z-LRWAN1）ボードとコンピュータを、USB ケーブル（Type A - mini B）と ST-LINK コネクタ（CN1）で接続します。CN2 の ST-LINK コネクタのジャンパが接続されていることを確認してください。PingPong のセットアップ図については、図 18 を参照してください。

図 18. PingPong セットアップ



5.6 AT_Slave アプリケーションの説明

このサンプルプログラムの目的は、外部ホストから UART 経由で AT コマンド･インタフェースで制御する LoRa モデムを実装することです。

外部ホストには、アプリケーションと AT ドライバを組み込んだホストマイクロコントローラか、または単にターミナルを実行するコンピュータを使用できます。

このアプリケーションは、CMWX1ZZABZ-091 LoRa モジュールを搭載した B-L072Z-LRWAN1 ディスカバリ･ボードを対象としています。このアプリケーションでは、STM32L072CZ 専用の Cube Low Layer ドライバ API を使用することで、コード･サイズを最適化します。

AT_Slave のサンプルプログラムは、内蔵 LoRa 無線を駆動する Lora スタックを実装しています。このスタックは、UART 経由で AT コマンド･インタフェースを使用して制御します。モデムは、外部ホストからの AT コマンドを処理しない限り、常に STOP モードです。

AT_Slave プロジェクトを起動するには、Projects\B-L072Z-LRWAN1\Applications\LoRa\AT_Slave フォルダに移動して、LoRa End_Node プロジェクトと同じ手順で好みのツールチェーンを起動します。

詳細については、アプリケーション･ノート AN4967 の AT コマンドとその説明の一覧を参照してください。

制限事項：AT_Slave モデムは、LoRaWAN V1.0.3 仕様を統合しています。ただし、クラス B モードを実行する適切な AT コマンドは提供しません。

クラス B モードを実行するには、ユーザが独自の AT コマンド･セットを開発する必要があります。

5.7 AT_Master アプリケーションの説明

このアプリケーションの目的は、センサ･データを読み取り、外部の LoRa モデム経由で LoRa ネットワークに送信することです。AT_Master アプリケーションは、外部 LoRa モデムに組み込まれている LoRa スタックを駆動するための AT コマンド一式を実装しています。

外部 LoRa モデムは、B-L072Z-LRWAN1 ディスカバリ･ボード、I-NUCLEO-LRWAN1 ボード（WM-SG-SM-42 USI モジュール･ベース）、または RiSiNGHF 社製モデム搭載の LRWAN-NS1 拡張ボード（P-NUCLEO-LRWAN3）を対象としています。

このアプリケーションは、STM32Cube HAL ドライバ API を使用し、STM32L0 シリーズを対象としています。

詳細については、アプリケーション･ノート AN4967 の AT コマンドとその説明の一覧を参照してください。

BSP プログラミングのガイドライン

表 43 では、外部 LoRa モジュールとやりとりする BSP （ボード･サポート･パッケージ）ドライバ API について説明します。

表 43. BSP プログラミングのガイドライン

関数	説明
ATEerror_t Modem_IO_Init (void)	モデムを初期化します。
void Modem_IO_DeInit (void)	モデムを初期化解除します。
ATEerror_t Modem_AT_Cmd (ATGroup_t, at_group, ATCmd_t Cmd, void *pdata)	モデムの IO コマンドです。

注： **NUCLEO ボードは、UART (PA2,PA3) 経由で拡張ボードと通信します。次の変更を適用する必要があります（UM1724 の 5.8 章を参照）。**

- **SB62 と SB63 を接続する必要があります。**
- **SB13 と SB14 をオープンにし、STLINK から STM32 の UART を切断する必要があります。**

6 システム性能

6.1 メモリ･フットプリント

表 44 の値は、Keil コンパイラ（ARM コンパイラ 5.05）の以下の設定に対して測定したものです。

- 最適化：サイズ･レベル 3 で最適化
- デバッグ･オプション：オフ
- トレース･オプション：オフ
- ターゲット：P-NUCLEO-LRWAN1 （STM32L073 + SX1272MB2DAS）

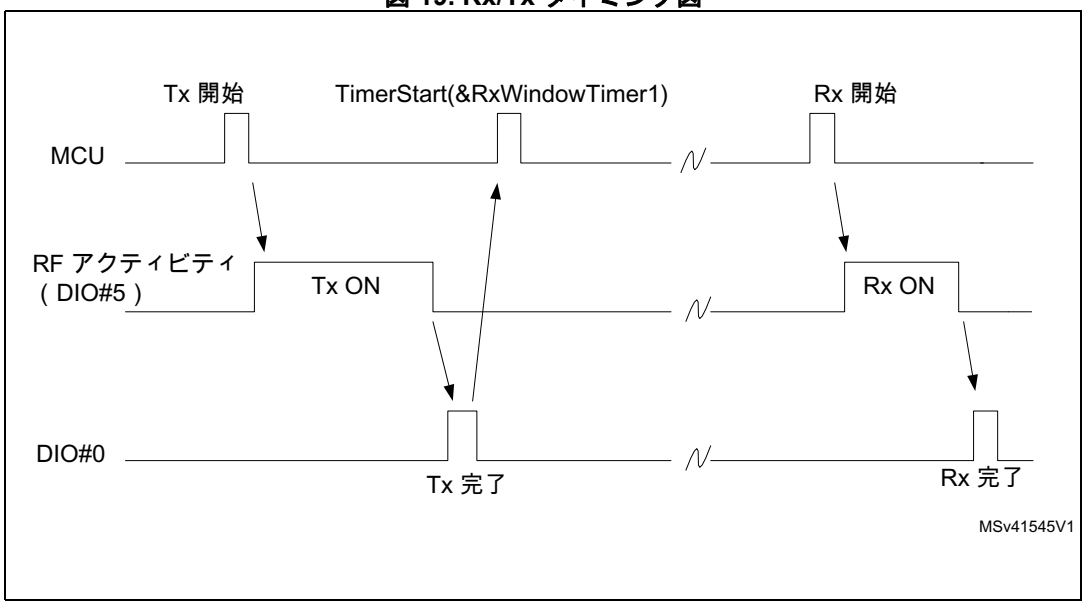
表 44. End_Node アプリケーションのメモリ･フットプリント値

プロジェクト	Flash （バイト）	RAM （バイト）	説明
アプリケーション層	4336	456	すべての microlib を含みます。
LoRa スタック	29926	3486	MAC + RF ドライバを含みます。
ハードウェア抽象化レベル	10362	1536	-
ユーティリティ	2474	464	system、timeserver、vcom、queue などのサービスを含みます。
全アプリケーション	52468	6362	全アプリケーションのメモリ･フットプリント。

6.2 リアルタイム制約

LoRa RF 非同期プロトコルでは、推奨される厳密な Tx/Rx タイミングに従います（Tx/Rx 図の例については、図 19 を参照）。SX1276MB1MAS 拡張ボードは、ユーザ透過な低ロック時間、高速自動較正に向けて最適化されています。LoRa 拡張パッケージ設計には、トランスミッタ起動時間とレシーバ起動時間の制約が組み込まれます。

図 19. Rx/Tx タイミング図



Rx ウィンドウチャネルの開始

アップリンク変調の終了後、Rx ウィンドウは、RECEIVE_DELAY1 として 1 秒間（±20 マイクロ秒）、または JOIN_ACCEPT_DELAY1 として 5 秒間（±20 マイクロ秒）開きます。

現在スケジュールされている割込みレベル優先度に従う必要があります。言い換えると、受信した起動時間に遅れるのを避けるために、すべての新しいユーザ割込みの割込み優先度は DI0#n 割込みより高い必要があります（表 41 を参照）。

6.3 消費電力

消費電力は、Nucleo ボードに SX1276MB1MAS シールドを取り付けた状態で測定しました。

測定セットアップ：

- DEBUG：無効
- TRACE：無効
- SENSOR_ENABLED：無効

測定結果：

- STOP モードの消費電流標準値：1.3 µA
- RUN モードの消費電流標準値：8.0 mA

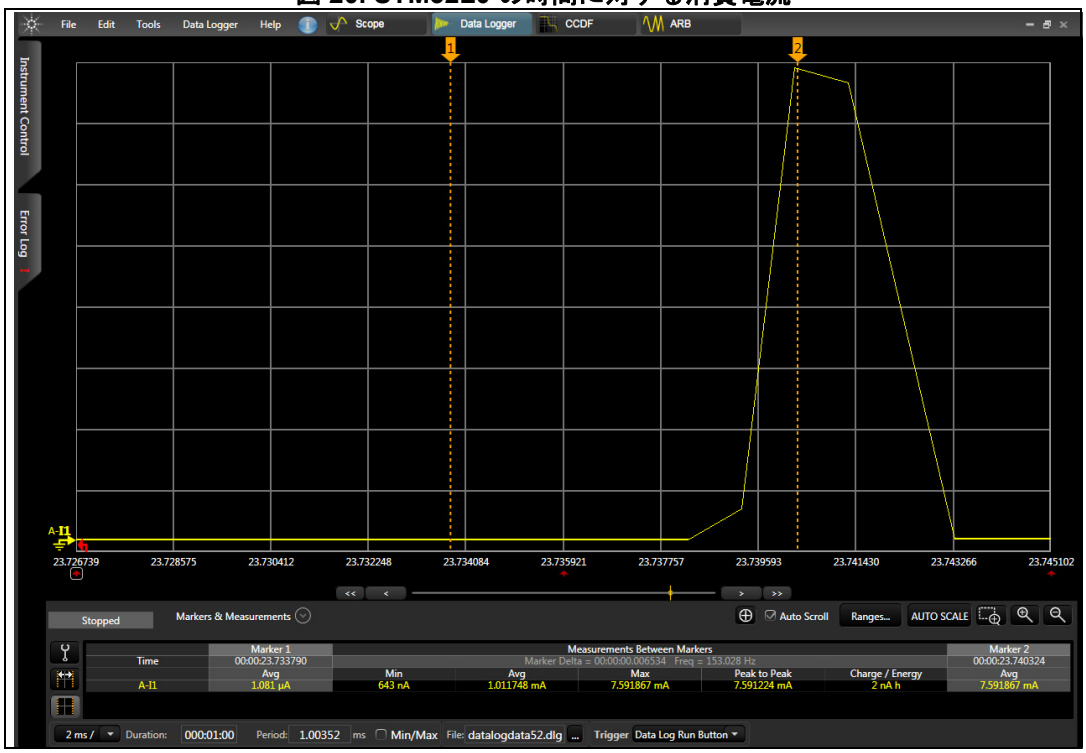
測定グラフ：

- 30 秒間の瞬時消費電流

図 20 に、STM32L0 シリーズのマイクロコントローラの時間に対する消費電流の例を示します。

参考資料

図 20. STM32L0 の時間に対する消費電流



参考資料

7 改版履歴

表 45. 文書改版履歴

日付	版	変更内容
2016 年 6 月 27 日	1	初版発行
2016 年 11 月 10 日	2	更新： – はじめに – セクション 2.1：概要 – セクション 3.2：機能 – セクション 5：サンプルプログラムの説明 – セクション 6：システム性能
2017 年 1 月 4 日	3	更新： – はじめに：CMWX1ZZABZ-xxx LoRa モジュール（村田製作所社）を参照。 – セクション 5.1：ハードウェアの説明：3 番目のハードウェア設定ファイルを追加。 – セクション 5.2：パッケージの説明：AT_Slave アプリケーションの追加。 追加： – セクション 5.5：AT_Slave アプリケーションの説明
2017 年 2 月 21 日	4	更新： – はじめに：I-NUCLEO-LRWAN1 LoRa 拡張ボード。 – 図 10：プロジェクト·ファイルの構造 – セクション 5.1：単一マイクロコントローラ·エンドデバイス·ハードウェアの説明 – 図 15：I-CUBE-LRWAN の構造 – セクション 5.4：End_Node アプリケーション – 表 27：アプリケーション設定のスイッチ·オプション – セクション 5.5：PingPong アプリケーションの説明 – セクション 5.6：AT_Slave アプリケーションの説明 – 表 29：End_Node アプリケーションのメモリ·フットプリント値 追加： – セクション 5.2：分割エンドデバイス·ハードウェアの説明（2 チップ·ソリューション） – セクション 5.7：AT_Master アプリケーションの説明
2017 年 7 月 18 日	5	追加： – セクション 5.4：End_Node アプリケーションに最大許容ペイロード長に関する注 – セクション 5.7：AT_Master アプリケーションの説明に NUCLEO ボードと拡張ボードの UART 経由の通信に関する注

表 45. 文書改版履歴 （続き）

日付	版	変更内容
2017 年 12 月 14 日	6	追加： – 新しいモデムの参照：RiSiNGHF 社モデム RHF0M003 を搭載した拡張ボード 更新： – 新しいアーキテクチャ設計（LoRa FSM を削除） – 図 10：プロジェクト･ファイルの構造 – 図 13：動作モデル
2018 年 7 月 4 日	7	追加： – 新しい拡張ボード – LoRaWAN クラス B モードの導入 更新： – 図 10 を 図 17 に更新、表 4、表 10 を表 45 に更新

表 46. 日本語版文書改版履歴

日付	版	変更内容
2018 年 12 月	1	日本語版　初版発行

参考資料

重要なお知らせ（よくお読み下さい）

STMicroelectronics NV およびその子会社（以下、ST）は、ST製品及び本書の内容をいつでも予告なく変更、修正、改善、改定及び改良する権利を留保します。購入される方は、発注前にST製品に関する最新の関連情報を必ず入手してください。ST 製品は、注文請書発行時点で有効なST の販売条件に従って販売されます。

ST製品の選択並びに使用については購入される方が全ての責任を負うものとします。購入される方の製品上の操作や設計に関してST は一切の責任を負いません。

明示又は黙示を問わず、STは本書においていかなる知的財産権の実施権も許諾致しません。

本書で説明されている情報とは異なる条件でST 製品が再販された場合、その製品についてSTが与えたいかなる保証も無効となります。

ST およびST ロゴはSTMicroelectronics の商標です。その他の製品またはサービスの名称は、それぞれの所有者に帰属します。

本書の情報は本書の以前のバージョンで提供された全ての情報に優先し、これに代わるものです。

この資料は、STMicroelectronics NV 並びにその子会社(以下ST)が英文で記述した資料（以下、「正規英語版資料」）を、皆様のご理解の一助として頂くためにSTマイクロエレクトロニクス㈱が英文から和文へ翻訳して作成したものです。この資料は現行の正規英語版資料の近時の更新に対応していない場合があります。この資料は、あくまでも正規英語版資料をご理解頂くための補助的参考資料のみにご利用下さい。この資料で説明される製品のご検討及びご採用にあたりましては、必ず最新の正規英語版資料を事前にご確認下さい。ST及びSTマイクロエレクトロニクス㈱は、現行の正規英語版資料の更新により製品に関する最新の情報を提供しているにも関わらず、当該英語版資料に対応した更新がなされていないこの資料の情報に基づいて発生した問題や障害などにつきましては如何なる責任も負いません。