Renesas Synergy™ プラットフォーム

Wi-Fi 移植ガイド

R11AN0342JU0100
Rev.1.00
2019.04.04

本資料は英語版を翻訳した参考資料です。内容に相違がある場合には英語版を優先します。資料によっては英語版のバージョンが更新され、内容が変わっている場合があります。日本語版は、参考用としてご使用のうえ、最新および正式な内容については英語版のドキュメントを参照ください。

要旨（Introduction）

Synergy プラットフォームに移植する Wi-Fi ドライバは、SSP ガイドラインに従う必要があります。このドキュメントは、サードパーティの Wi-Fi ドライバを移植する場合のガイドラインと手順を説明します。また、このドキュメントは ThreadX と SSP アーキテクチャを使用する場合の設計に関する検討事項（design considerations）も要約しています。

必須リソース（Required Resources）

ハードウェア

- 任意の Synergy キット (https://www.renesas.com/jp/ja/products/synergy/hardware/kits.html)
- Windows 7/8/10 が動作しているホスト PC

開発ツールとソフトウェア

- e² studio ISDE (https://www.renesas.com/jp/ja/products/synergy/software/tools/e2-studio.html)、または IAR Embedded Workbench® for Renesas Synergy™ (https://www.renesas.com/jp/ja/products/synergy/software/tools/iar-ew-for-synergy.html)
- Synergy ソフトウェアパッケージ (SSP) (https://www.renesas.com/jp/ja/products/synergy/software/ssp.html)

前提条件と対象読者（Prerequisites and Intended Audience）

- このアプリケーションノートは、ユーザに Renesas e² studio ISDE と Synergy ソフトウェアパッケージ (SSP) の使用経験があることを前提としています。Wi-Fi ドライバの移植を開始する前に、『SSP ユーザーズマニュアル』の手順に従い「Blinky」プロジェクトをビルドして実行してください。この作業を実行することで、e² studio と SSP の使用方法を理解できます。また、ユーザが SSP のベストプラクティスとガイドラインを理解していることも想定しています。
- 対象読者は、Wi-Fi モジュールを Synergy プラットフォームに移植することを希望しているユーザです。ユーザは、組み込みと ThreadX の概念を理解している必要があります。

目次

1. テンプレートの概要（Overview of Templates）

SSP に移植する Wi-Fi ドライバは、パックファイル (pack file) の形式で提供されます。カスタムパックの作成を容易にする目的で、Pack File Templates (パックファイルテンプレート) が提供されています。

スタックが存在する場所に応じて、2 種類の Wi-Fi テンプレートが利用できます。

- Wi-Fi On-MCU：Synergy MCU 上に Wi-Fi スタックが存在
- Wi-Fi On-Module：Wi-Fi モジュール上に Wi-Fi スタックが存在

これらの各テンプレートをカスタマイズし、カスタム Wi-Fi ドライバの実装、SSP フレームワークの追加、コンフィギュレータ機能の追加、Synergy ドライバへの接続をおこなうことができます。

1.1 パックファイルの構造（Pack File Structure）

パックファイルは、以下のもので構成されています。

- Content（コンテンツ）：パックとともに配布されるコードやドキュメントです。
- PDSC File（PDSC ファイル）：Pack Descriptor File（パック記述子ファイル）であり、パック内に格納されているすべてのファイルとフォルダを記述しています。
- XML Configurator（XML コンフィギュレータ）：Synergy Configurator の **[Threads]** (スレッド) タブ内で、モジュールの位置とモジュールのプロパティを定義している XML ファイルです。すべての XML ファイルは、`.module descriptions` フォルダ内に配置されます。

1.2 フォルダ構造（Folder Organization）

すべてのコードは `/synergy/company` フォルダ内に配置されます。

1.2.1 On-MCU の場合のフォルダ構造（Folder Organization for On-MCU Case）

パックファイル内にあるすべての SSP 関連コードは、`inc` と `src` の各フォルダ内で構成（organized）されます。

SSP 関連コードは、以下の 4つのファイルで構成されています。これらのファイルに、Wi-Fi モジュールドライバの API、マクロ（macro）、または構造体（structure）の定義（definition）を記述しないでください。

1. `sf_wifi_partnumber.h`：これは、インスタンス（instance）のヘッダファイル（header file）です。Wi-Fi モジュールの設定に関する構造体を定義しています。この中で、低レベルドライバ（low-level driver）、リセット端子、スレーブセレクト端子、タイムアウト、外部 IRQ インスタンス、構成可能なその他のプロパティに関するインスタンスを定義しています。

   このファイルは、`/synergy/company/inc/framework/instances/` フォルダ内に配置されます。

2. `sf_wifi_partnumber_private_api.h`：このファイルは、Wi-Fi モジュールがサポートする API インスタンスに関する宣言（declaration）を記述しています。

   このファイルは、`/synergy/company/src/framework/sf_wifi_partnumber/` フォルダ内に配置されます。

3. `sf_wifi_partnumber_private.h`：このファイルは、フレームワーク内で使用する構造体とマクロを記述しています。また、モジュールに対応するユーザ定義の構成で使用する、拡張された cfg 構造体も定義しています。

   このファイルは、`/synergy/company/src/framework/sf_wifi_partnumber/` フォルダ内に配置されます。

4. `sf_wifi_partnumber.c`：このファイルは、contains the implementation of the API instances declared in `sf_wifi_partnumber_private.h` ファイル内で宣言された API インスタンスの実装を記述しています。

これらのファイルの内容に関する詳細な説明は、4.2.2 章 を参照してください。

Wi-Fi モジュールのすべてのドライバは、`vendor_wifi_driver` フォルダに配置されています。

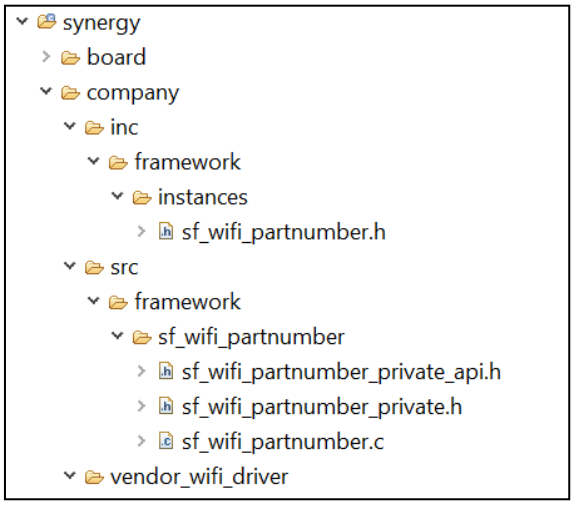

図 1　フォルダ構造 – On-MCU の場合

1.2.2　On-Module の場合のフォルダ構造（Folder Organization for On-Module Case）

On-Module の場合、パックファイルはベンダドライバまたは Wi-Fi モジュールドライバのみを格納しています。

これらのドライバは、`vendor_wifi_driver` フォルダに配置されています。フォルダ内のファイル構造に関して、特に規定はありません。ライブラリファイルを配置することも可能です。

1.3　パックファイルの命名規約（Pack File naming）

パックファイルの命名規約は、そのパックファイルが On-MCU と On-Module のどちらを想定しているかによって決まります。

On-MCU パックファイルは、<Company>.Synergy_wifi_<partnumber>_OnMCU_<Version> という形式で命名されます。

On-Module パックファイルは、<Company>.Synergy_wifi_<partnumber>_OnModule_<Version> という形式で命名されます。

`.pdsc` ファイルの名前（ファイル名本体）はパックファイルと同じであり、XML ファイルはパックファイルの中(.pdscファイル)で指定した名前と同じにする必要があります。命名規約の詳細は、3 章で説明します。

2.　テンプレートのカスタマイズ（Customizing templates）

2.1　概要（Overview）

以下のものに合わせてテンプレートをカスタマイズする必要があります。

- コード
- XML ファイルとその名前
- PDSC ファイルとその名前
- パックファイルの名前

テンプレート全体で、「company」という用語をコード設計会社の名前に置き換える必要があります。同様に、「vendor」を Wi-Fi モジュールのベンダ名に、「partnumber」を Wi-Fi モジュールの型番に置き換える必要があります。

2.2　コードのカスタマイズ（Customizing Code）

SSP Wi-Fi フレームワークを使用するのは、On-MCU スタックの場合のみにとどめる必要があります。

2.2.1　On-Module の場合（On-Module Case）

On-Module の場合、すべてのアプリケーションコード、および抽象化レイヤ（abstraction layer）を使用する場合はそれらすべてを、`/src` フォルダ内に配置する必要があります。

2.2.2 On-MCU の場合（On-MCU Case）

On-MCU の場合、提供されているテンプレートに従ってください。すべてのコードは、テンプレート内の適切なファイルの中に記述する必要があります。On-MCU の場合にサポートされているすべての API のインスタンスは、Wi-Fi モジュールに合わせて定義を行う必要があります。

2.3 XML コンフィギュレータのカスタマイズ（Customizing XML configurator）

XML コンフィギュレータのファイルは、`.module_descriptions` フォルダ内にあります。

コンフィギュレータファイル内に、以下の 2 種類の要素があります。

- config
- module

module 要素はコンフィギュレータのプロパティを定義するのに対し、config 要素はアプリケーションレベルでモジュールの構成を定義します。

2.3.1 config 要素（Config elements）

一般的に、config 要素は、スタックサイズと、Parameter Checking (パラメータチェック) のようなプリプロセッサ設定を記述します。また、`cfg.h` ファイルへのパスも記述します。このファイル内で config のパラメータが生成されます。
On-MCU と On-Module どちらの場合も、このパスを
`synergy_cfg/ssp_cfg/framework/company/sf_wifi_partnumber_cfg.h` にする必要があります。

XML ファイル内で、次の行:`id="config.framework.sf_wifi_partnumber_v2" path="ssp_cfg/framework/company/sf_wifi_partnumber_cfg.h" version="0"` は、適切なフォルダを指すように変更する必要があります。

エラーを最小化するために、パラメータチェックを記述し、有効にすることを推奨します。

2.3.2 module 要素（module elements）

`module` 要素には以下のような 3 つのパートがあります。

1. コンフィギュレータ内でのモジュールの位置:コンフィギュレータ内にあるモジュールの位置を指すように、次の行を変更する必要があります。

   ```
   config="config.framework.sf_wifi_partnumber_v2"
   display="Framework|Networking|Wi-Fi|${module.framework.sf_wifi_v2.name}
   CUSTOM Wi-Fi Device Driver on sf_wifi_partnumber"
   id="module.framework.sf_wifi_partnumber_v2" version="1"
   ```

   On-MCU の場合、モジュールを **[Framework|Networking|Wi-Fi]** タブ内に配置する必要があります。そのモジュールは、**Vendor Wi-Fi Device Driver on sf_wifi_partnumber** という形で表示されます。

   On-Module の場合、モジュールを **[Framework|Networking|Wi-Fi]** タブ内に配置する必要があります。そのモジュールは、**On-Module Vendor Wi-Fi Driver**. という形で表示されます。

   XML の名前は、コンフィギュレータの [components] タブ内にあるモジュールの位置を定義します。

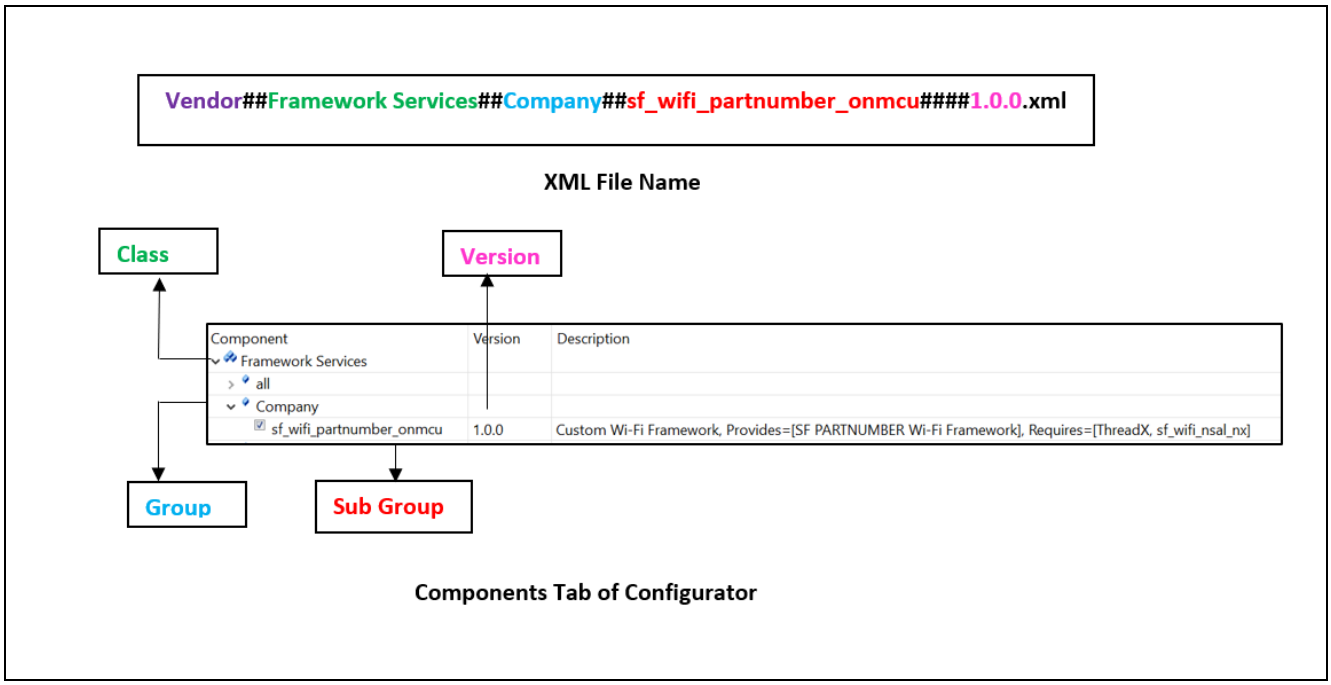

図 2　コンフィギュレータの [Components] タブ内にあるモジュールの位置

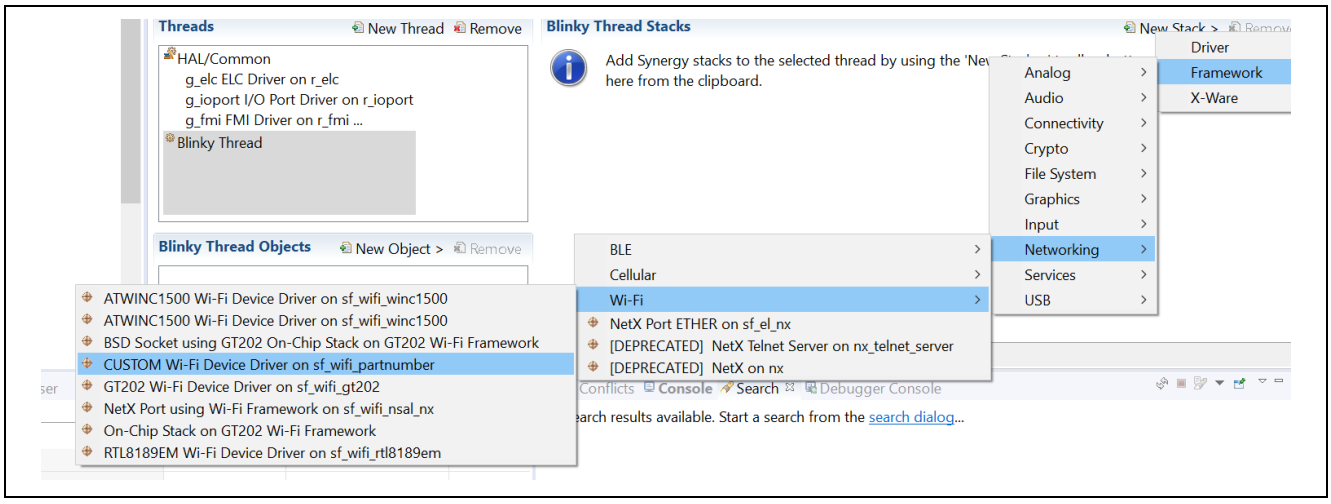

図 3　コンフィギュレータの [Threads] タブ内にある On-MCU モジュールの位置

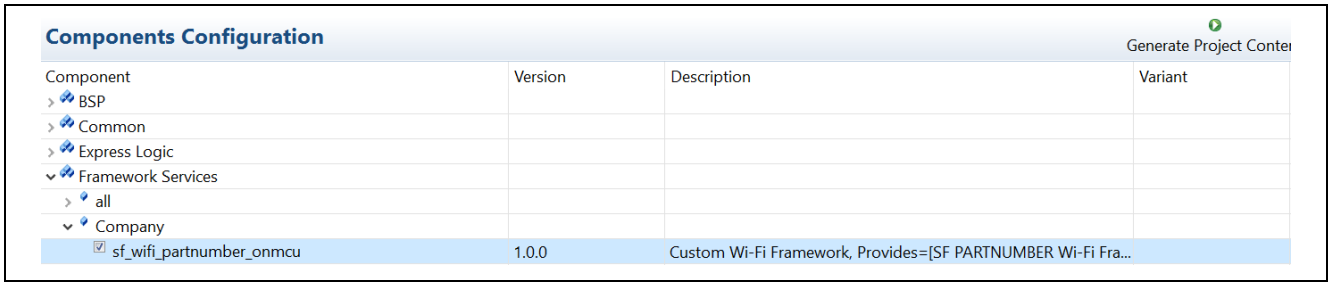

図 4　コンフィギュレータの [Components] タブ内にある On-MCU モジュールの位置

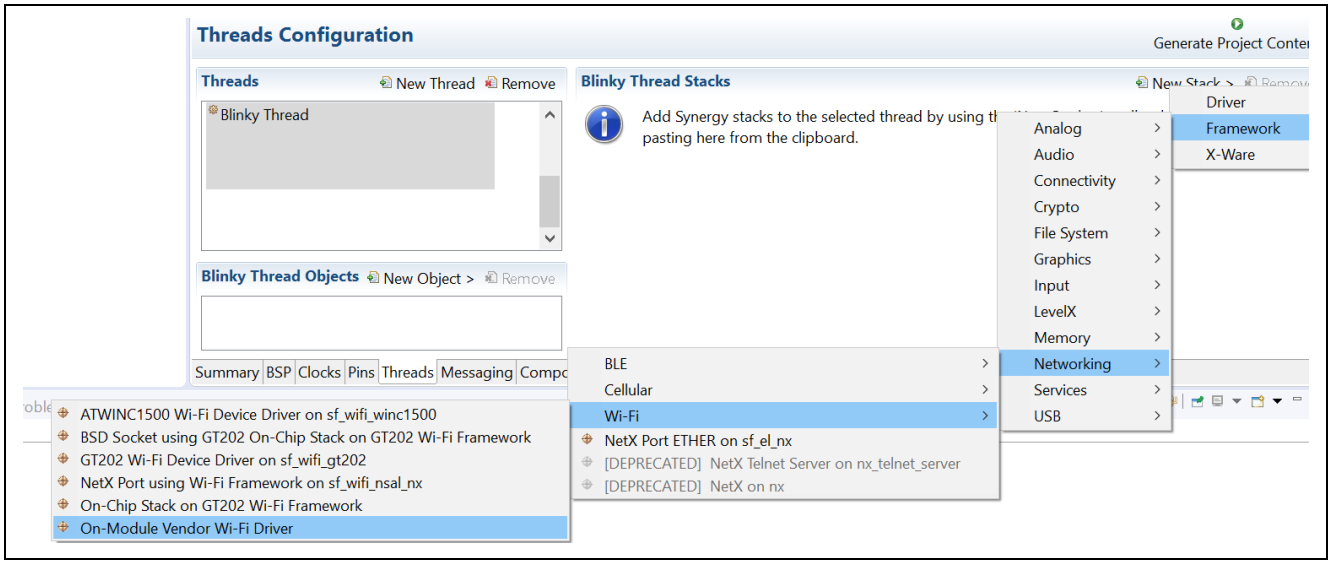

図 5　コンフィギュレータの [Threads] タブ内にある On-Module モジュールの位置

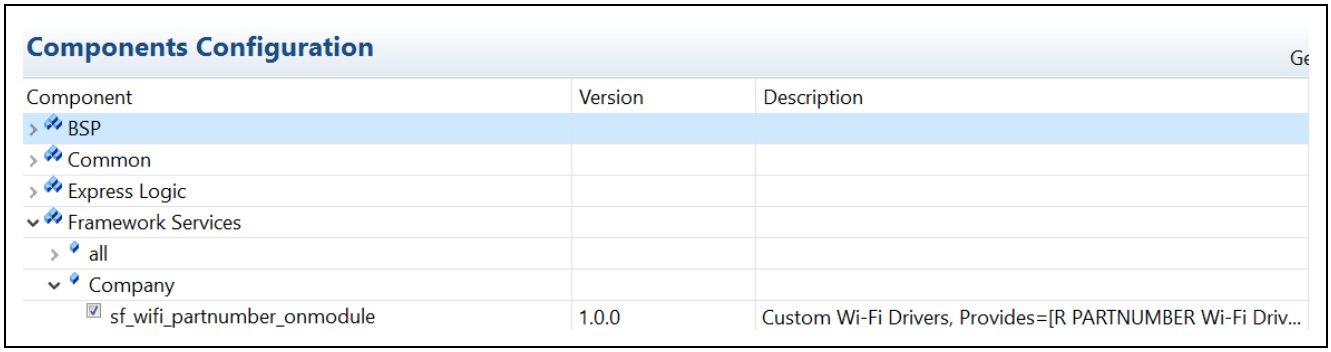

図 6　コンフィギュレータの [Components] タブ内にある On-Module モジュールの位置

2. コンフィギュレータのプロパティ:これらのプロパティは、コンフィギュレータ内に表示されるモジュールのプロパティと、それぞれのデフォルト値を記述します。「requires」要素は Wi-Fi モジュールより上のレベル、また「provides」要素は Wi-Fi モジュールより下のレベルにある他のモジュールを表します。テンプレート形式の XML ファイルでは、SPI、UART、SDMMCの3 種類の周辺機能（peripheral）を例として記述しています。モジュールの要件に基づいて、これらのいずれかを使用する必要があります。

   その他のプロパティに該当するのは、Reset Pin (リセット端子)、Slave Select Pin (スレーブセレクト端子)、tx power (送信電力)、Thread Priority (スレッド優先順位)、Timeouts (タイムアウト)、callbacks (コールバック)、external IRQs (外部割り込み要求) などです。モジュールの必要に応じて、これらを使用できます。

   注記:アプリケーションが動作するように、端子構成を変更する必要があります。

3. 定義:XML ファイルの 3 番目のパートは、コンフィギュレータが生成する構造体の定義と、必須のヘッダファイルの定義です。

   On-MCU の場合、 `sf_wifi_api.h`、`sf_wifi_partnumber.h`、 `sf_wifi_nsal_api.h` の各ファイル、および他のファイル内でインクルードされていない場合は他の SSP インクルードファイル（include file）すべてを、このファイルの includes セクション内でインクルードする必要があります。

```
<includes>
#include &amp;quot;sf_wifi_api.h&amp;quot;
#include &amp;quot;sf_wifi_partnumber.h&amp;quot;
#include &amp;quot;sf_wifi_nsal_api.h&amp;quot;
</includes>
```

ctrl と cfg の各構造体の定義、およびモジュールのインスタンスも、XML ファイル内で宣言します。モジュールのインスタンスで、api、ctrl、cfg の各構造体をインクルードする必要があります。

上記の各構成に加えて、Wi-Fi フレームワークの NSAL レイヤがサポートしているゼロコピー（zero-copy）機能も、構成の中で宣言されます。ゼロコピー機能の詳細は、4.2 章を参照してください。

XML ファイルをさらに変更する方法の詳細は、『SSP Module Guide』を参照してください。

2.4 PDSC ファイルのカスタマイズ（Customizing pdsc file）

PDSC ファイルで、以下のものを記述します。

- GCC コンパイラのサポート
- IAR コンパイラのサポート
- XML コンフィギュレータのファイルに関する説明。また、コンフィギュレータの [Components] タブ内にあるモジュールの位置も定義します。
- パック内に格納されているすべてのファイルとフォルダの宣言

パックに格納しようとするフォルダは、`category="include"` の下に記述する必要があります。ヘッダファイルは `category="header"` の下に記述し、ソースファイル（source file）は `category="source"` の下に記述する必要があります。

PDSC ファイルを作成する際に、以下の 2 点を満たしていることを確認してください。

- .pdsc ファイル内にある XML ファイルの記述は、その XML ファイルの名前と一致している必要があります。
  図 7 に、PDSC ファイル内にある XML ファイルの記述と、それに対応する XML ファイル名の例を示します

```
<component Cclass="Framework Services" Cgroup="Company"
Csub="sf_wifi_partnumber_onmcu" Cvendor="Vendor"
Cversion="1.0.0" condition="">
```

図 7 PDSC ファイル内にある XML の記述

```
Vendor##Framework
Services##Company##sf_wifi_partnumber_onmcu####1.0.0.xml
```

図 8 XML ファイル名

- PDSC ファイルの名前は、パックファイルの名前からバージョン番号を取り除いた名前にする必要があります。
  例：パックファイルの名前が `Company.Synergy_wifi_partnumber_OnMCU.1.0.0.pack` である場合、PDSC ファイルに `Company.Synergy_wifi_partnumber_OnMCU.pdsc` という名前を付ける必要があります。

上記の条件のどちらか一方でも満たしていない場合、パックファイルは機能しなくなります。

2.5 バージョン管理（Version Control）

パックファイルのバージョンは、SSP のバージョンから独立しています。パックファイルの初期バージョンは、1.0.0 です。パックファイルに変更を加えるたびに、バージョンを大きくする必要があります。メジャー（major）、マイナー（minor）、パッチ（patch）の各バージョンは、開発者の裁量で決定できます。

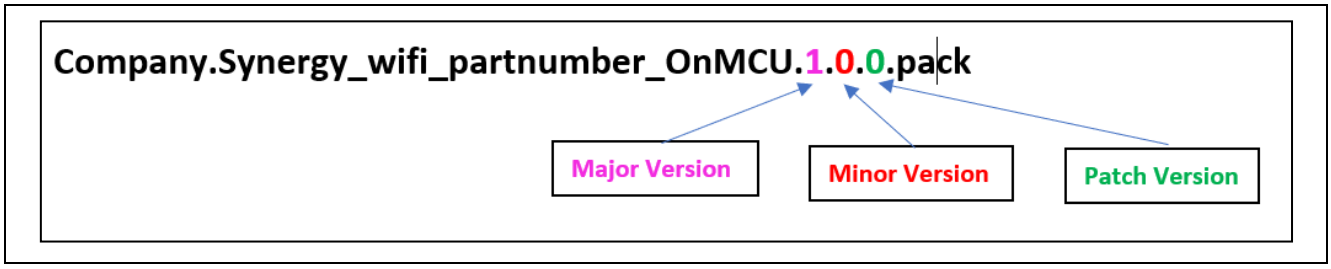

図 9　パックファイルのバージョン構造

パックファイル内に格納されているファイルのバージョンは、パックのバージョンに従う必要はありません。この結果、パックから独立した形で、ファイルに対する変更を追跡することができます。

```
<component Cclass="Framework Services" Cgroup="Company" Csub="sf_wifi_partnumber_onmcu" Cvendor="Vendor" Cversion="1.5.0" condition="">
<description>Custom Wi-Fi Framework, Provides=[SF PARTNUMBER Wi-Fi Framework], Requires=[ThreadX, sf_wifi_nsal_nx]</description>
<files>
    <file category="include" name="synergy/company/inc/framework/instances/"/>
    <file category="include" name="synergy/company/src/framework/sf_wifi_partnumber/"/>
    <file category="include" name="synergy/company/vendor_wifi_driver"/>
    <file category="header" condition="" name="inc/framework/instances/sf_wifi_partnumber.h" select="" src="" version="1.0.0"/>
    <file category="header" condition="" name="vendor_wifi_driver/example_folder/include/nm_bsp.h" select="" src="" version="1.2.0"/>
    <file category="source" condition="" name="vendor_wifi_driver/example_folder/source/nm_bsp.c" select="" src="" version="1.5.0"/>
</files>
</component>
```

図 10　PDSC ファイル (バージョン 1.5.0) で、バージョン番号が異なるファイルを記述

/packs フォルダ内に、同じ名前のパックファイルが複数存在している場合、コンフィギュレータはデフォルトで、バージョン番号がもっとも新しいパックファイルを選択します。この動作を変更するには、[Components] タブで希望のパックファイルを手動で選択し、他のパックファイルを選択解除する必要があります。

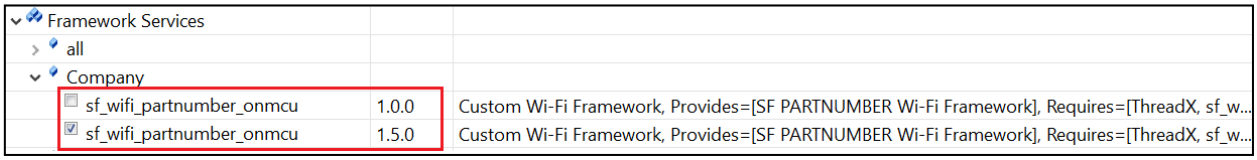

図 11　コンフィギュレータは、バージョンがより大きいパックを自動的に選択

2.6　パックファイルの生成（Generating the pack file）

- 図 12 に示すように、パック内に格納しようとするすべてのフォルダとファイルを選択します。
- 右クリックし、7-Zipのメニューから「”～.zip”に圧縮」を選択します。

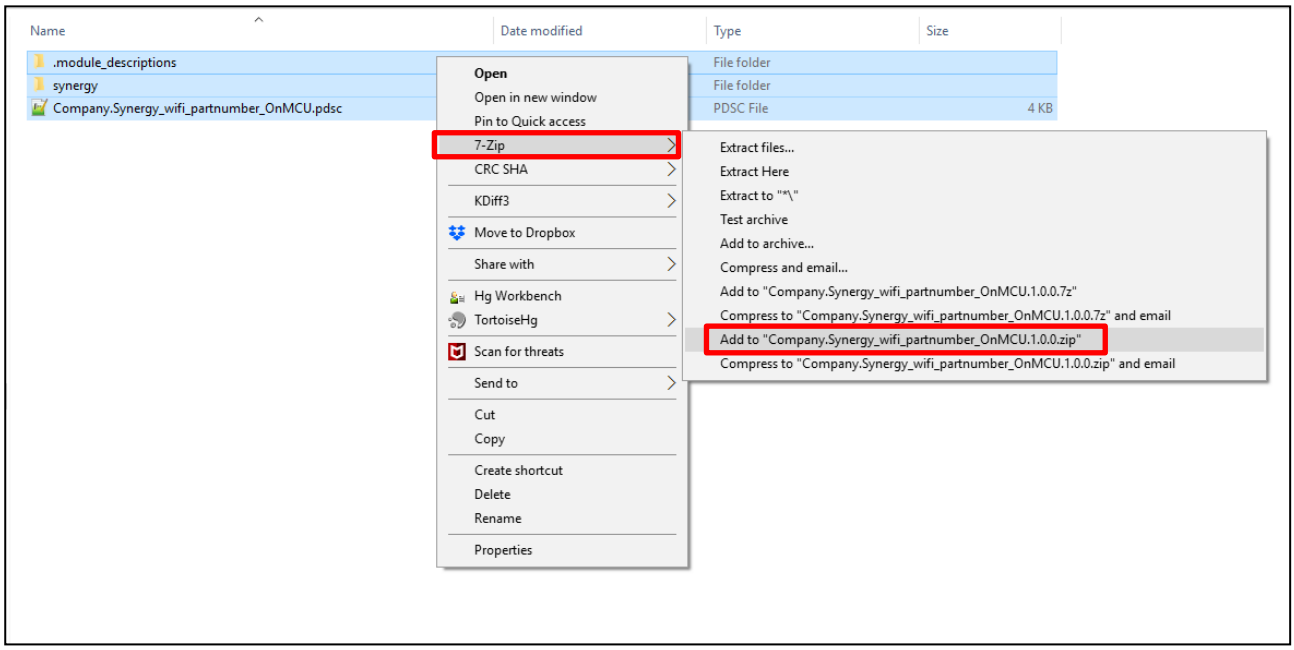

図 12　パックファイルの作成

- ファイルの拡張子 を .pack に変更します。
  例：Company.Synergy_wifi_partnumber_OnMCU.1.0.0.**zip** を、
  Company.Synergy_wifi_partnumber_OnMCU.1.0.0.**pack** に変更します
- e² studio の場合、その .pack ファイルを /Synergy/e2studio/internal/projectgen/arm/packs/ フォルダにコピーします。IAR Workbench の場合、Synergy/ssc/internal/projectgen/arm/packs/ フォルダにコピーします。

2.7 新しい SSP と新しい e² studio へのパックファイルの移行（Migration of Pack Files with new SSP and e2 studio）

パックファイルのバージョンは SSP のバージョンに依存していないので、新しいバージョンの SSP を使用する場合でもパックファイルに変更を加える必要はありません。パックファイル内で変更を加える必要があるのは、コードに変更を加えた場合のみです。一方、新しい e² studio または IAR Workbench を使用する場合は必ず、それに対応する /packs フォルダにパックファイルをコピーする必要があります。

パックファイルの作成に関する詳細は、Reference Pack File (パックファイルに関するリファレンス) を参照してください。

3. パックファイルのインポート（Importing Pack File）

.pack ファイルは、以下の 2 つの方法でインストールできます。

- e² studio を使用している場合、パックファイルを手動で /Synergy/e2studio/internal/projectgen/arm/packs/ フォルダにコピーします。
-
- IAR Workbench を使用している場合、パックファイルを手動で /Synergy/ssc/internal/projectgen/arm/packs/ フォルダにコピーします。
- e² studio の [Import] (インポート)を使用して、パックファイルを以下の方法でインストールします。
  — [File] (ファイル) -> [Import] (インポート) -> [General] (一般) -> [CMSIS Pack] (CMSIS Pack) を選択します
  — [Next] (次へ) をクリックします
  — パックファイルの場所を参照し、そのファイルを選択します
  — [Finish] (終了) をクリックします

パックファイルのインストールまたは削除を行うときは、e² studio と SSC を閉じておくことを強く推奨します。

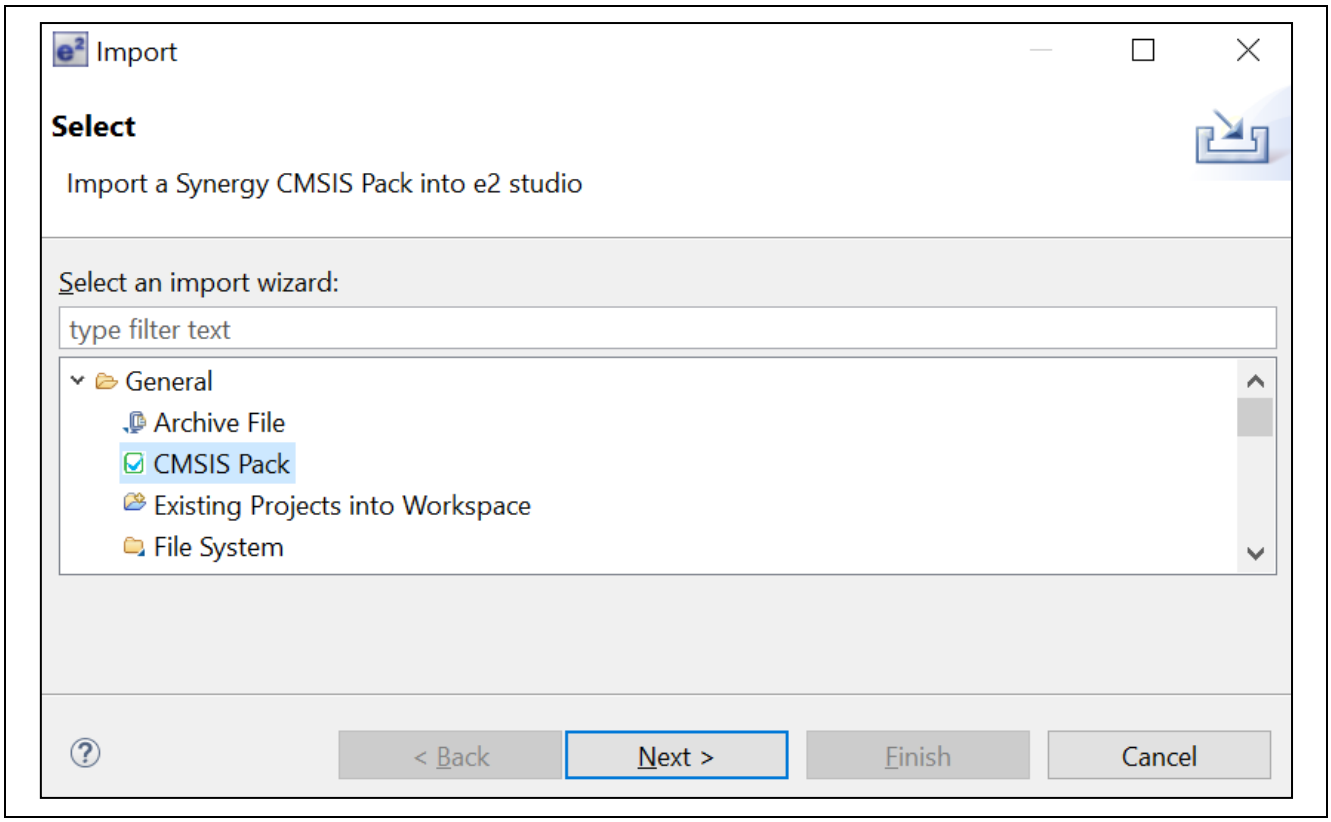

図 13 e² studio でのパックファイルのインポート

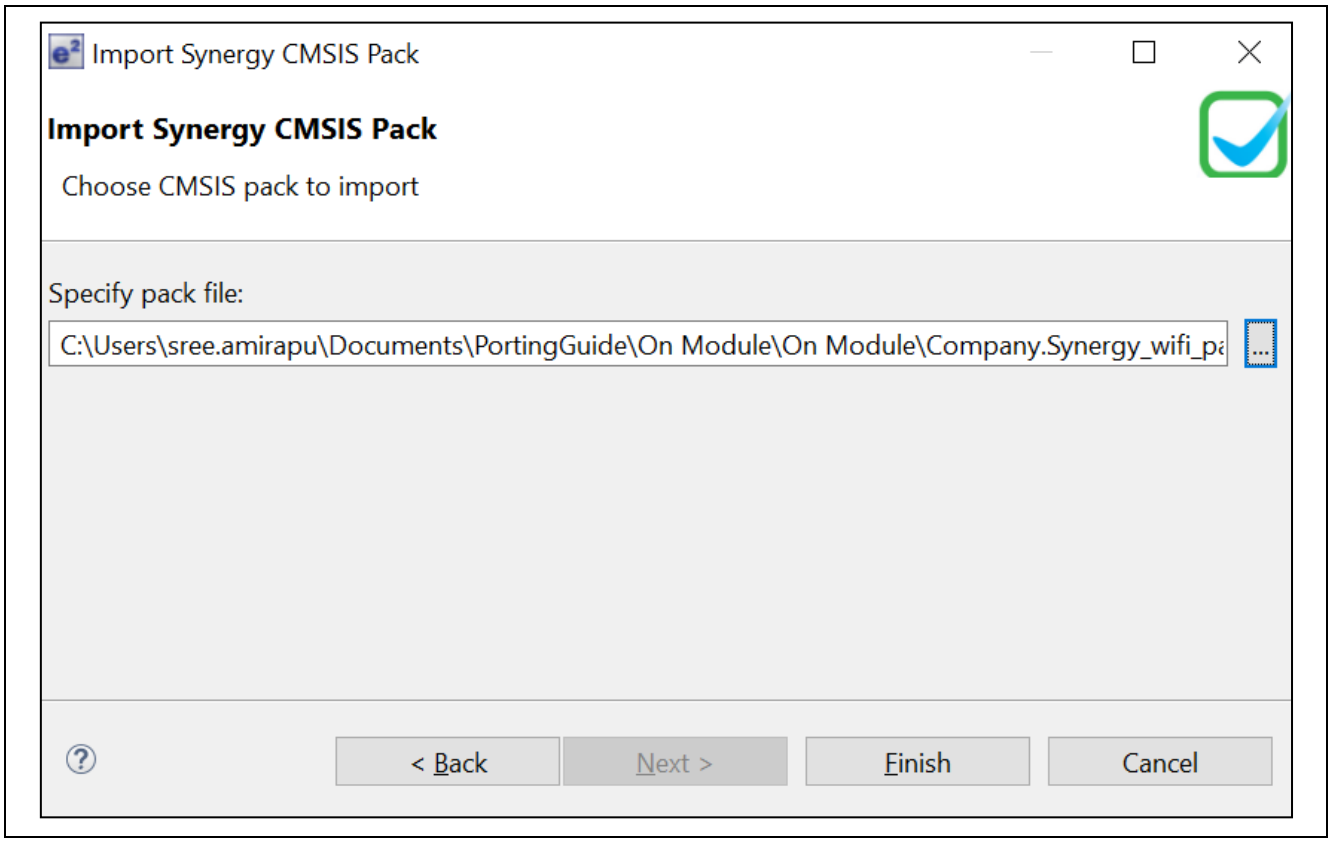

図 14 e² studio でのパックファイルのインポート

パックファイルのアンインストールは、手動で実行します。アンインストールしようとするパックファイルを、IAR Workbench または e² studio に対応する /packs/ フォルダから削除する必要があります。

パックのインストールとアンインストールのどちらを実行した場合も、新しいパックファイルが使用できるように、e² studio または IAR Workbench を再起動する必要があります。

時には、/packs/ フォルダ内の supportfilepacks.xml ファイルを削除した後、e² studio または IAR workbench を再起動する必要が生じることがあります。

4. Synergy プラットフォームに合わせたドライバ適応の概要（Overview of Driver Adaptation for Synergy Platform）

4.1 RTOS の適応（RTOS Adaptation）

この章で、ThreadX と Synergy プラットフォームと組み合わせて動作させる場合の高レベルの概要を説明します。ただし、ユーザが RTOS の全般的な概念を理解していることを想定しています。この章では、RTOS の詳細な概念について説明していません。

Synergy プラットフォームでは、ライブラリの形で付属している ThreadX がデフォルトでリンクされます。この結果、コンパイル時間が大幅に改善します。アプリケーションとは別に ThreadX Source (ThreadX ソース) を毎回コンパイルする必要がなくなるからです。以下に、ThreadX に移行する際に考慮する必要のある、ThreadX と Synergy プラットフォームに関する基本的な情報を示します。

4.1.1 コンパイラの依存関係（Compiler Dependency）

- ThreadX は、ANSI C で記述されています
- ThreadX のソースコードは、ASCII 形式です
- ThreadX はエンディアンに依存しません（endian neutral）

4.1.2 スレッド（Threads）

スタックサイズ（stack size）、優先順位（priority）、タイムスライス間隔（time slicing interval）など、スレッドのプロパティが構成可能である場合、コンフィギュレータを使用して SSP 内でスレッドを追加することができます。

4.1.2.1 スケジューリング（Scheduling）

ThreadX のスケジューリングは、優先順位に基づいています。優先順位が同じ複数のスレッドをスケジューリングするために、ラウンドロビン（round-robin）スケジューリングを使用しています。

4.1.2.2 スレッドの同期（Thread Synchronization）

メッセージキュー：メッセージキューは、ThreadX 内でスレッドを同期するための主要な手段です。メッセージサイズは少なくとも 32 ビット (1 ワード) の長さであり、最大サイズとして 16 ワードを使用できます。16 ワードより長いメッセージは、ポインタ（pointer）の形で渡す必要があります。この仕様は、メモリ使用量の最適化にも貢献します。図 15 に示すように、Synergy Configurator でメッセージキューを作成できます。メッセージサイズとキューサイズは、**[Properties]** (プロパティ) タブで構成できます。

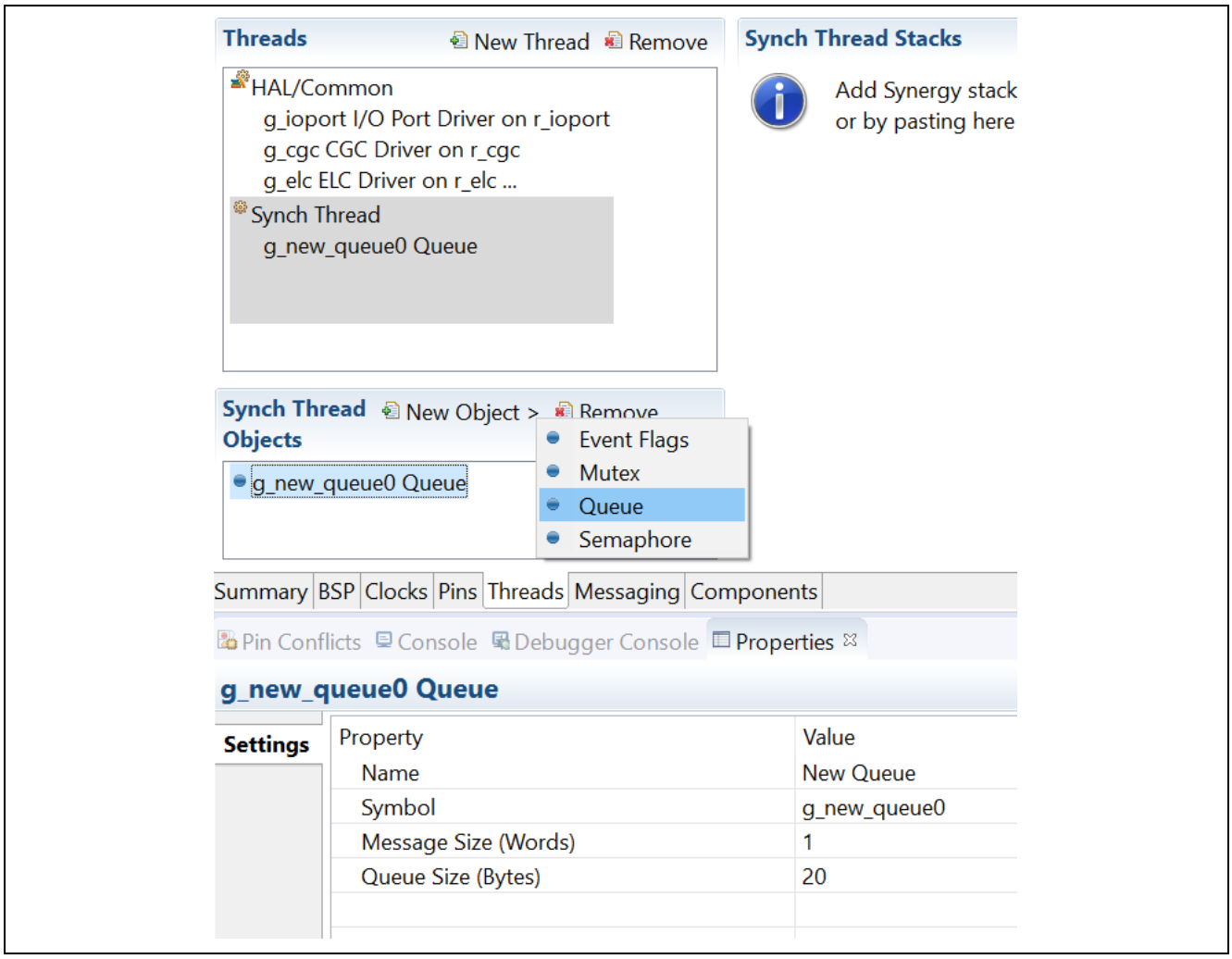

図 15　スレッド同期オブジェクト

ミューテックス：コンフィギュレータでミューテックス（mutex）を作成することもできます。スレッドを選択し、メッセージキューと同様の方法でミューテックスを作成します。[Priority Inheritance] (優先順位の継承) は、デフォルトで無効になっています。優先順位の逆転を防止するには、[priority inheritance] を有効にします。また、[priority inheritance] に関連して、タイムフレーム（timeframe）を必ず使用してください。

セマフォ：ThreadX 内のセマフォ（semaphore）は、32 ビットのカウント用セマフォ（counting semaphore）です。セマフォは、イベント通知（event notification）の目的でも使用できます。セマフォも、コンフィギュレータ内で作成します。セマフォカウントを増減させるには、`tx_semaphore_get` および `tx_semaphore_put` を使用します。

イベントフラグ：アプリケーションによっては、イベントフラグ（event flag）がスレッド同期の最も効率的な手段になることがあります。ThreadX 内のイベントフラグは、同じメモリ位置にある、32 個の個別ビットで形成されたグループです。イベントフラグの設定は、AND/OR の論理演算（logical operation）で実施します。そのため、メモリの使用量と速度を最適化する場合、セマフォよりイベントフラグを推奨します。

4.1.3　メモリ（Memory）

ThreadX の静的なメモリ使用法はコンパイラとリンカが決定するのに対し、動的なメモリ使用法はアプリケーションが決定します。つまり、スタックやキューに関連付けられるメモリは、どこに配置してもかまいません。

ThreadX には、複数の内蔵メモリプール（built-in memory pool）があります。メモリブロックプール（Memory Block Pool）とメモリバイトプール（Memory Byte Pool）です。ブロックプールは、複数の固定サイズブロックで形成されています。ブロックプールは、動的メモリの割り当て（allocation）で最も推奨される手法です。フラグメンテーション（fragmentation、断片化）の問題が関係しないからです。バイトプールはヒープ（heap）に似ているので、フラグメンテーションを引き起こす可能性があります。ただし、メモリプールは柔軟に使用できるという利点があります。プール内のメモリが使用可能になるまで、アプリケーションのスレッドはサスペンドされることがあります。

4.1.4 ISR

アプリケーションは、ThreadX 内の割り込み(interrupt)を管理することができます。一部の ThreadX API は、ISR (割り込みサービスルーチン) 内から呼び出すことができます。

4.1.5 その他のプロパティ(Other Properties)

システムタイマ(system timer)の長さは、デフォルトで 100 ティック(tick)、つまり 10 ms に設定されています。ただし、この値はアプリケーションの必要に応じて変更することもでき、その場合は以下の図に示すようにコンフィギュレータ内で **[ThreadX Source]** を **[HAL/Common Thread]** に追加します。現実には、ThreadX アプリケーションはシステムタイマをまったく使用せずに動作させることも可能です。

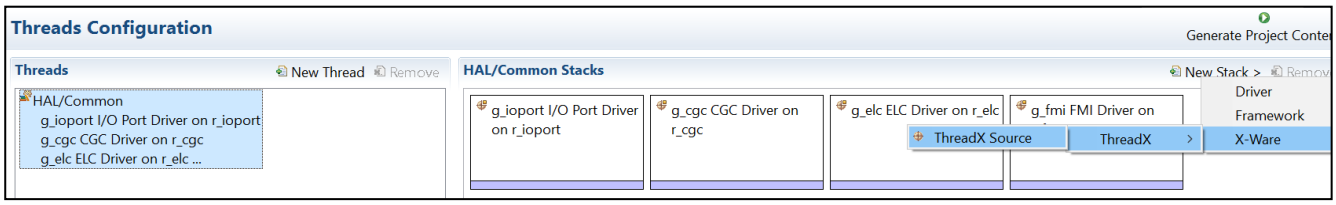

図 16 ThreadX Source のカスタマイズ

ThreadX には、ThreadX ソース内で変更できるその他のプロパティが複数あります。最小スタックサイズ(minimum stack size)、プリエンプションしきい値(preemption threshold)、イベントトレース(event trace)、パフォーマンス情報(performance info)などです。

リソース不足を最小化するために、プリエンプションしきい値を有効にすることを強く推奨します。動作中スレッドに対するプリエンプト (抑止) が発生する可能性があるのは、プリエンプト側 (他のスレッドを抑止する側) スレッドの優先順位が、動作中スレッドの PreemptionThreshold (プリエンプションしきい値) より高い場合のみです。

4.1.6 エラーコード(Error Codes)

以下の 2 種類のエラーコードがあります

- 共通エラーコード(Common Error Code)は、すべてのモジュール間で共通です。これらは、`/synergy/SSP` フォルダに配置されている `ssp_common_api.h` ファイルに記述されています。
- モジュール固有のエラーコード(module specific error code)。これらは、モジュールの API ヘッダファイル内で定義されています。

詳細は、『ThreadX User Manual』を参照してください。

4.2 SSP への移植(Porting to SSP)

Wi-Fi ドライバを SSP に移植するときに最初に登場する設計上の検討事項は、Synergy MCU と Wi-Fi モジュールのどちらでスタックを実行するか、という点です。どちらのアプローチにも、それぞれの利点があります。

On-MCU の場合、SSP Wi-Fi フレームワークを活用できます。Wi-Fi フレームワークを形成しているのは、Wi-Fi API、NetX スタック(NetX Stack)、NASL(Network Stack Abstraction Layer、ネットワークスタック抽象化レイヤ)、Wi-Fi モジュールインタフェース(Wi-Fi module interface)です。On-MCU の場合、モジュールドライバの詳細を理解する必要なく、Synergy プラットフォーム上でのアプリケーションのビルド (開発) を容易に実施できるようになります。また、ドライバを統合する際の整合性が向上し、アプリケーションと複数のモジュールを組み合わせて動作させることが可能になります。

On-Module の場合、メモリに制約が課されている状況で最適な手段になります。また、アプリケーションがサードパーティ製スタックとの統合を必要とする場合、On-Module の手法を推奨します。

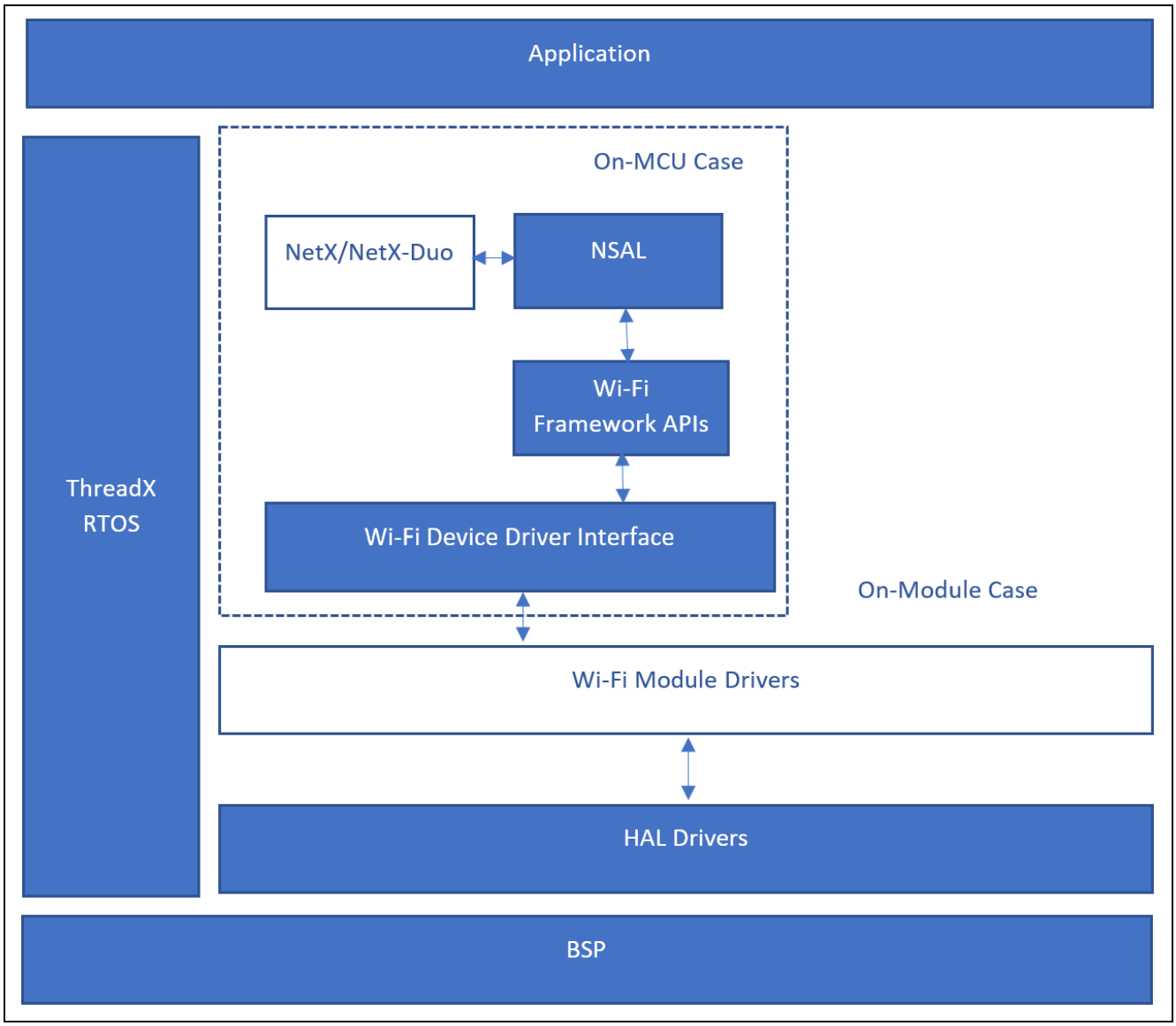

図 17　SSP と Wi-Fi モジュールの組み合わせ – On-MCU の場合と On-Module の場合

4.2.1　SSP フレームワークによる Wi-Fi 機能のサポート（SSP Wi-Fi Framework Feature Support）

SSP フレームワーク内で、Wi-Fi の汎用機能の大半に対応する列挙型（enum）が定義されています。これらの列挙型は、`sf_wifi_api.h` ファイル内で定義されています。以下に、これらの機能のいくつかを示します。

4.2.1.1　Wi-Fi 規格（Wi-Fi standards）

SSP Wi-Fi フレームワークは、802.11a、802.11b、802.11g、802.11n の各規格をサポートしています。また、インフラストラクチャ（Infrastructure）BSS (Basic Service Set、基本サービスセット) と、IBSS (Independent (ad-hoc) BSS、独立 (アドホック) BSS) 両方のタイプの BSS をサポートしています。

4.2.1.2　Wi-Fi の複数のモード（Wi-Fi Modes）

SSP Wi-Fi フレームワークは、アクセスポイントモード（Access Point mode）とクライアント (ステーション) モード（Client (Station) mode）の両方をサポートしています。

4.2.1.3　セキュリティタイプ（Security Types）

SSP フレームワークは、WEP、WPA2、WPA、Open セキュリティをサポートしています。TKIP、CCMP、WEP の各暗号化タイプもサポートしています。WEP 鍵は、ASCII 鍵および 16 進（Hex）鍵の両方の形式に対応します。

4.2.1.4 電力管理(Power Management)

SSP フレームワーク内には、電力管理向けの具体的な API は存在していません。ユーザは、モジュールの Tx (送信) 電力を設定することができます。この電力は、1 ～ 17 dBm の範囲内で設定できます。ただし、.close API は、ネットワークインタフェースの初期化を取り消すほか、インタフェースを低消費電力モードに移行するか、モジュールの動作が無効になっている場合はインタフェースの電源をオフにします。

4.2.1.5 その他の機能(Other features)

- アクセス制御リスト(Access Control List)の管理:どのデバイスをアクセスポイント(access point)に接続できるのか、アプリケーションが制御できるようになります。
- マルチキャストフィルタリスト(Multicast Filter List)の管理:アプリケーションがマルチキャストグループ(multicast group)に参加または離脱できるようになります
- ゼロコピー(Zero-Copy):この機能を使用すると、NetX パケットのツールと構造体をサポートできるモジュールによるメモリ使用量を最適化することができます。ゼロコピーが有効な場合、Wi-Fi モジュールから NetX に、また NetX からモジュールにパケットが直接コピーされます。処理の目的でドライバのバッファを経由することはありません。この動作により、データの複数のコピーが生じることはなく、処理時間も大幅に短縮されます。

4.2.2 On-MCU の場合のフレームワークの使用法(Framework Usage for On-MCU case)

この章で、SSP Wi-Fi フレームワークの概要について説明します。フレームワークの詳細な説明は、『Wi-Fi フレームワークモジュールガイド』を参照してください。

Wi-Fi フレームワークには、3つの主要なコンポーネントがあります

- API
- ネットワークスタック抽象化レイヤ (NSAL)
- NetX ライブラリ

  NSAL レイヤは、Wi-Fi ドライバの MAC レイヤデータフレームと NetX スタックの間でインタフェースとして機能します。

  SSP フレームワークの各インスタンスは、以下の 3 種類の構造体を使用します
  - `sf_wifi_ctrl_t`:ユーザ定義の Wi-Fi 用ctrl構造体へのポインタ
  - `sf_wifi_cfg_t`:ユーザ定義の Wi-Fi 用cfg構造体へのポインタ
  - `sf_wifi_api_t`:Wi-Fi フレームワーク API へのポインタを保持しています

  これら 3 種類の構造体は、`sf_wifi_instance_t` struct 内で定義されています。
  SSP が定義する API と構造体は、`sf_wifi_api.h` ファイル内で定義されています。
  これらの構造体に加えて、Wi-Fi モジュールは、そのモジュールの機能に固有の拡張された cfg と ctrl の各構造体も記述しています。これらの構造体は、それぞれ `sf_wifi_partnumber.h` と `sf_wifi_partnumber_private.h` のファイルで定義されています。
  すべての API インスタンスは、`sf_wifi_partnumber.c` ファイル内で定義されています。これらの API インスタンスは、Wi-Fi モジュール API の呼び出しを行います。

4.2.3 API の実装(API Implementation)

1. `SF_WIFI_PARTNUMBER_Open()`

   この関数は、モジュールドライバを初期化します (モジュールの open() 関数を呼び出します)。また、cfg 構造体内で定義されている複数のパラメータを構成し、プロセス全体で必要とされるパラメータ全般を作成します。

2. `SF_WIFI_PARTNUMBER_Close()`

   この関数は Wi-Fi ドライバとネットワークインタフェースの初期化を取り消し、モジュールドライバに関連付けられているスレッドや他のオブジェクトを削除します。この関数は、対応するモジュール API の呼び出しを行います。

3. `SF_WIFI_PARTNUMBER_MulticastListAdd()`

   この関数は、MAC アドレスをマルチキャストフィルタリストに追加します。

4. `SF_WIFI_PARTNUMBER_MulticastListDelete()`

   この関数は、MAC アドレスをマルチキャストフィルタリストから削除します。

5. SF_WIFI_PARTNUMBER_StatisticsGet()

   この関数は、Wi-Fi インタフェースの統計情報を取得します。統計情報として、受信に成功したパケットの数（number of packets received successfully）、送信に成功したパケットの数（number of packets transmitted successfully）、送信エラーの数（number of transmit errors）が該当します。

6. SF_WIFI_PARTNUMBER_Transmit()

   この関数は、Wi-Fi ドライバから送信キュー（transmit queue）にパケットを送信します。

7. SF_WIFI_PARTNUMBER_ProvisioningSet ()

   この関数は、モジュールをクライアントモードと AP モードのどちらかに設定します。この関数は、SSID と他のプロビジョニング情報を必要とします。

8. SF_WIFI_PARTNUMBER_ProvisioningGet()

   この関数はプロビジョニングに必要な情報である、SSID、暗号化タイプ（encryption type）、セキュリティタイプ（security type）、鍵（key）、チャネルタイプ（channel type）を取得します。

9. SF_WIFI_PARTNUMBER_InfoGet()

   この関数は、モジュール情報とネットワーク情報を取得します。取得する情報は、チップセット/ドライバ情報（chipset/driver information）、RSSI の値（RSSI value）、信号ノイズレベル（Signal noise level）、リンク品質（link quality）などです。

10. SF_WIFI_PARTNUMBER_Scan()

    この関数は、アクセスポイントをスキャンします

11. SF_WIFI_PARTNUMBER_ACLAdd()

    この関数は、MAC アドレスをアクセス制御リストに追加します

12. SF_WIFI_PARTNUMBER_ACLDelete()

    この関数は、MAC アドレスをアクセス制御リストから削除します

13. SF_WIFI_PARTNUMBER_MacAddressSet()

    この関数は、Wi-Fi モジュールの MAC アドレスを構成します。

14. SF_WIFI_PARTNUMBER_MacAddressGet()

    この関数は、Wi-Fi モジュールの構成済み MAC アドレスを読み取ります

15. SF_WIFI_PARTNUMBER_VersionGet()

    この関数は、モジュールのバージョンを設定します

4.2.4 プロセスフロー（Process Flow）

Wi-Fi モジュールの初期化プロセスは、以下のとおりです。

1. NetX IP インスタンスが nx_ip_create を呼び出します。
2. nx_ip_create は、NetX NSAL ドライバのエントリポイントを呼び出します。
3. NSAL ドライバのエントリポイントは Wi-Fi フレームワークの open() 関数を呼び出します。
4. Wi-Fi フレームワークの open() 関数は、Wi-Fi デバイスドライバの open() 関数を呼び出し、Wi-Fi モジュールを初期化して有効にします。
5. nx_ip_interface_status_check API を呼び出し、NX_IP_LINK_ENABLED ステータスが設定されるまで待ちます。
6. 実行が成功した時点で、Wi-Fi モジュールはスキャンとプロビジョニングを実行する準備ができます。

Wi-Fi パケットの送信は以下の手順で実施されます。

1. ユーザアプリケーションコードが Wi-Fi モジュールをプロビジョニングします。
2. ユーザアプリケーションコードは NetX の TCP/UDP Socket Send API を呼び出します。
3. NetX Send API は、パケット送信の目的で、NSAL ドライバのエントリポイントを呼び出します。
4. NSAL ドライバのエントリポイントは Synergy Wi-Fi フレームワークの Transmit API 関数を呼び出します。
5. Synergy Wi-Fi フレームワークの Transmit API 関数は、Wi-Fi デバイスドライバの Transmit API 関数を呼び出します。
6. Wi-Fi デバイスドライバはユーザデータを送信します。

Wi-Fi パケットの受信は以下の手順で実施されます。

1. Wi-Fi パケットの受信は、Wi-Fi デバイスドライバの割り込みサービスルーチンが開始します。
2. パケットを受信した時点で、Wi-Fi デバイスドライバの receive コールバック関数は受信データをバッファに転送し、Wi-Fi フレームワークの receive コールバック関数を開始します。
3. Wi-Fi フレームワークの receive コールバック関数は、NSAL の receive コールバック関数を呼び出します。
4. NSAL の receive (受信) コールバック関数は、NetX の deferred receive (遅延受信) 処理コールバックを呼び出します。

4.3 SSP Wi-Fi フレームワークの制限事項（SSP Wi-Fi Framework Limitations）

メモリの制約上、Wi-Fi フレームワークは Synergy MCU の S1 シリーズをサポートしていません。

5. テストのガイドライン（Testing Guidelines）

- パックファイルアプリケーションは、ビルド時のエラーや警告なしで生成する必要があります。
- 最小限の受け入れ基準とテスト基準:コードは、プロジェクト開始の時点で合意した最小限の受け入れ条件（acceptance criterion）とテスト条件（testing criterion）を満たす必要があります。
- モジュールの機能を提示できるように、アプリケーションをパックファイルに格納する必要があります
- アプリケーションプロジェクト内に C-stat レポートを格納する必要があります。
- e² studio の -O2 オプションを指定してアプリケーションを最適化する必要があります。
- IAR 内で、[Optimization] (最適化) を [High (Balanced)] (高い (バランス型)) に設定する必要があります。
- パラメータチェックを有効にします
- コードは、『SSP Best Practices Guide』に準拠している必要があります

6. 既知の問題と制限（Known Issues & Limitations）

- e² studio で Export Pack Tool (エクスポートパックツール) を使用する場合の制限:このツールは、Class (クラス) カテゴリで半角スペース文字を使用することを許可しません。したがって、半角スペースを含む「Framework Services」というタイプのクラスを作成することはできません。したがって、テンプレートを使用することを推奨します。
- Export Pack (エクスポートパック) ツールを使用してパックファイルを作成した時点で、パックファイルを解凍（unzip）し、`.module_description` フォルダと XML ファイルを手動で追加する必要があります。
- SSP とは独立したバージョン番号を割り当てるには、パックファイル内と XML ファイル内の Vendor 名を「Renesas」以外の値にする必要があります。ベンダフィールドが「Renesas」であり、バージョンが SSP のバージョンと一致していない場合、コンフィギュレータ内にパックファイルが表示されません。
- 新しいパックファイルをインストールした場合、既存のプロジェクトはそのパックファイルをサポートしません。新しくインストールするパックファイルを、すでに開いているプロジェクト内に含める必要がある場合は、以下に従ってください。

- `supportfilepacks.xml` ファイルを `/packs` フォルダから削除し、e² studio または IAR Workbench を再起動する必要があります。
- `.module_descriptions` フォルダをプロジェクトから削除し、プロジェクトコンテンツをもう一度生成する必要があります。
- 上記のどの回避方法も成功しない場合、そのプロジェクトを作成しなおす必要があります。

7. Reloc の On-MCU パックファイルを使用する Wi-Fi アプリケーションの例（Example Wi-Fi Application using Reloc On-MCU Pack File）

ATWINC1500_OnMCU アプリケーションプロジェクトは、アプリケーション内で Wi-Fi On-MCU のカスタムパックファイルを使用する例を示します。このプロジェクトは、Reloc PMOD.WM1A Wi-Fi モジュールを AP モードに設定します。

7.1 必須リソース（Required Resources）

ハードウェア

- Renesas Synergy™ SK-S7G2
- Reloc PMOD.WM1A (Microchip® ATWINC15x0-MR210 Wi-Fi™ 無線モジュール)
- 2 本の micro USB コネクタケーブル

ソフトウェア

- Synergy ソフトウェアパッケージ (SSP) v1.5.0-rc.1
- Renesas Synergy™ e² studio 統合ソリューション開発環境 (ISDE) v6.2.1
- IAR Embedded Workbench® for Renesas Synergy™ (IAR EW for Synergy) v8.23.1
- Microsoft® Windows® 7 または 10

7.2 アプリケーションのセットアップ（Application Setup）

Synergy USB CDC ドライバファイルを https://www.renesas.com/jp/ja/products/synergy/software/add-ons/usb-cdc-drivers.html からダウンロードします

Synergy USB CDC ドライバをインストールするには、『Application Note – Installing Synergy signed USB CDC Drivers』の説明に従います。

1. Reloc PMOD.WM1A Wi-Fi モジュールを PMOD B に接続します。ジャンパ J15 を 3V3 の位置に設定します。

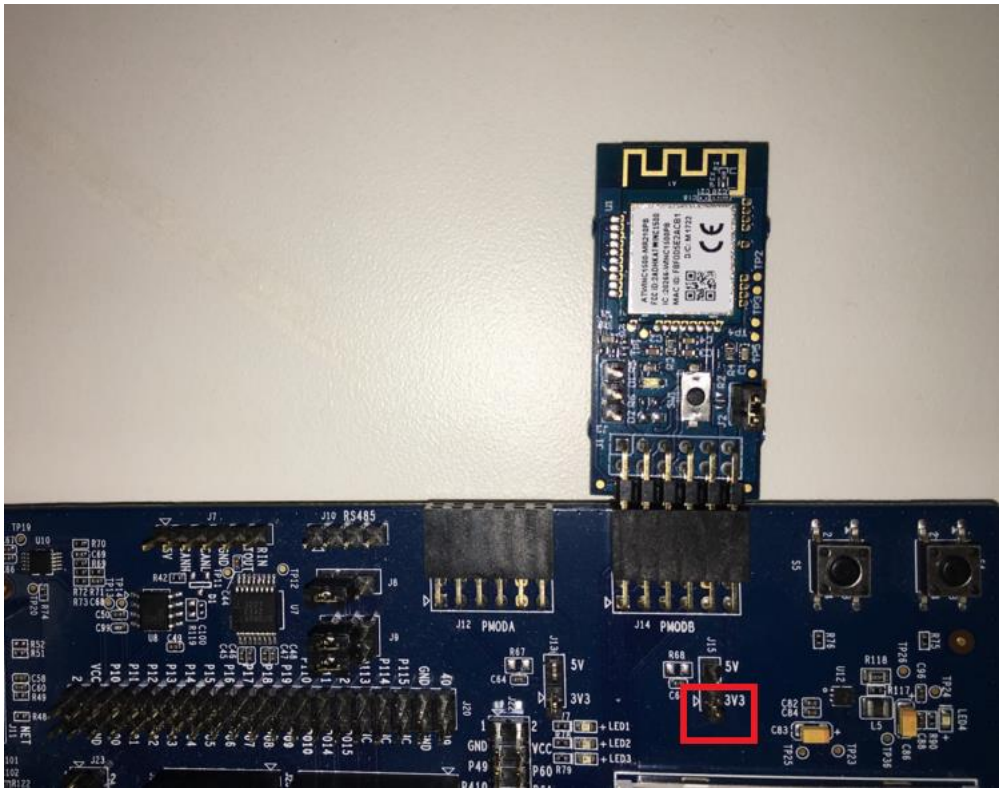

図 18　PMOD とジャンパの設定

2. micro USB ケーブルを J19 ポートに接続し、ボードの電源を投入します
3. もう一本の micro USB ケーブルを、シリアルコンソールに対応する J5 に接続します

7.3　ATWINC1500_OnMCU プロジェクトのインポートとビルド（Importing and Building ATWINC1500_OnMCU Project）

e² studio ver 6.2.1/IAR ver 8.23.1 の Packs フォルダにファイルをコピーします。

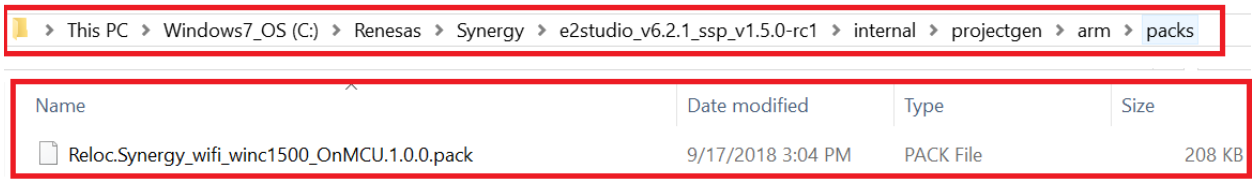

図 19　e² studio の Packs フォルダにコピーしたパックファイル

Synergy プロジェクトのインポート、ビルド、実行方法の説明は、『Synergy SSP Import Guide』を参照してください。

7.4 アプリケーションの実行（Running the Application）

コンピュータからコマンドコンソールにアクセスするには、TeraTerm Pro (または同等のシリアルターミナルエミュレータ) をインストールし、適切な COM ポートを構成してターミナルを開きます。

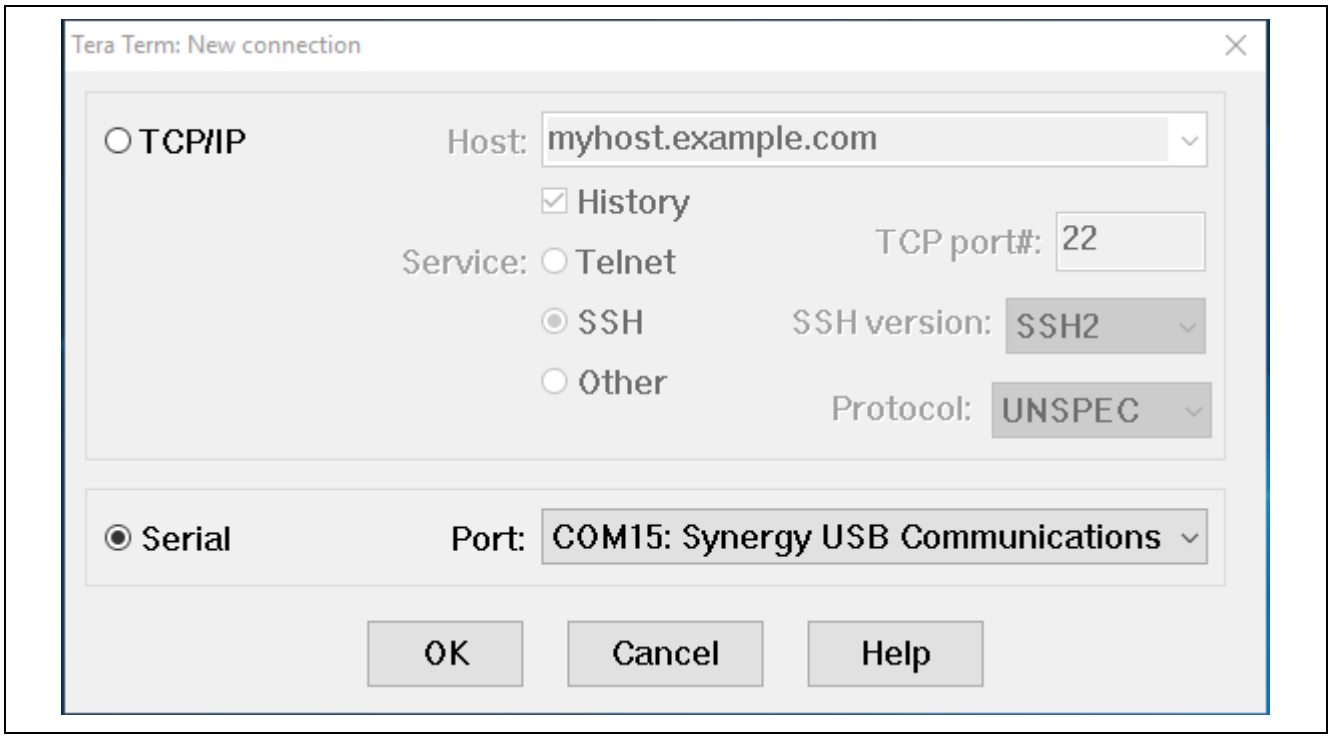

図 20 COM ポート

「?」と入力し、Enter キーを押して、コマンドメニューを表示します。

```
Synergy Wi-Fi Framework>?
  Synergy Wi-Fi Framework Help Menu
    1 : Press 1 and then the Enter key to input the Wi-Fi AP SSID
    2 : Press 2 and then the Enter key to input the Wi-Fi AP Password if needed
```

「1」を押して SSID を入力します。

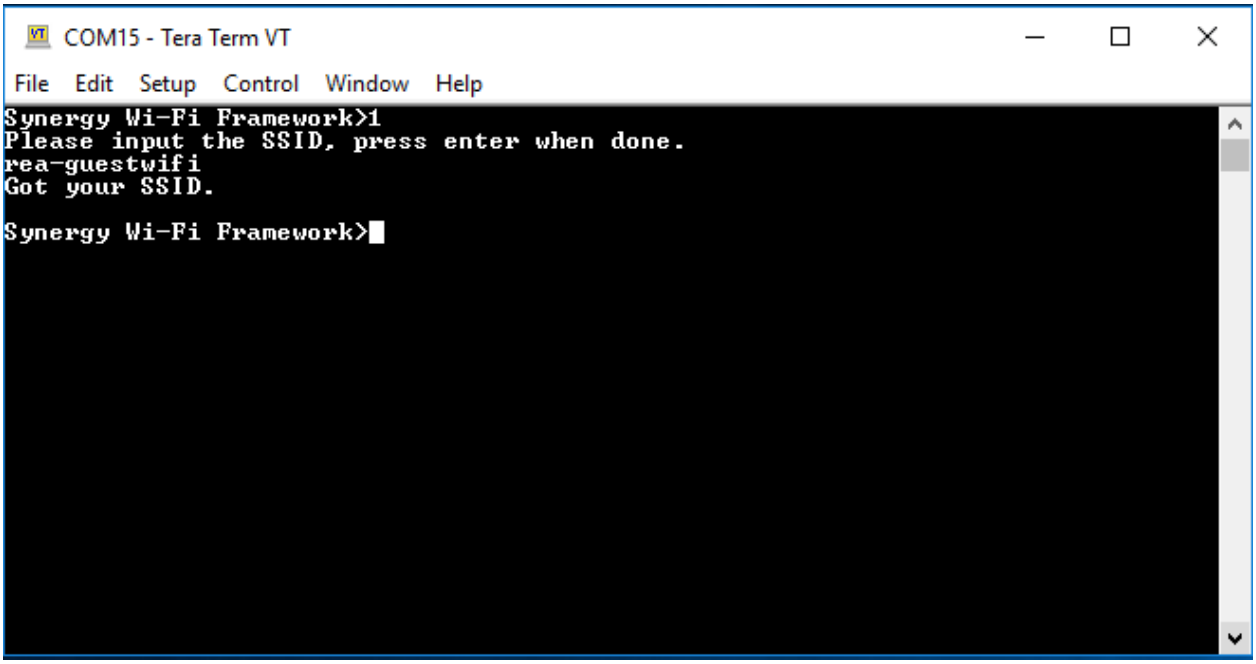

図 21 SSID の入力

「2」を押してパスワードを入力します。

図 22 パスワードの入力

プロビジョニングが成功した場合、IP アドレスが割り当てられます。

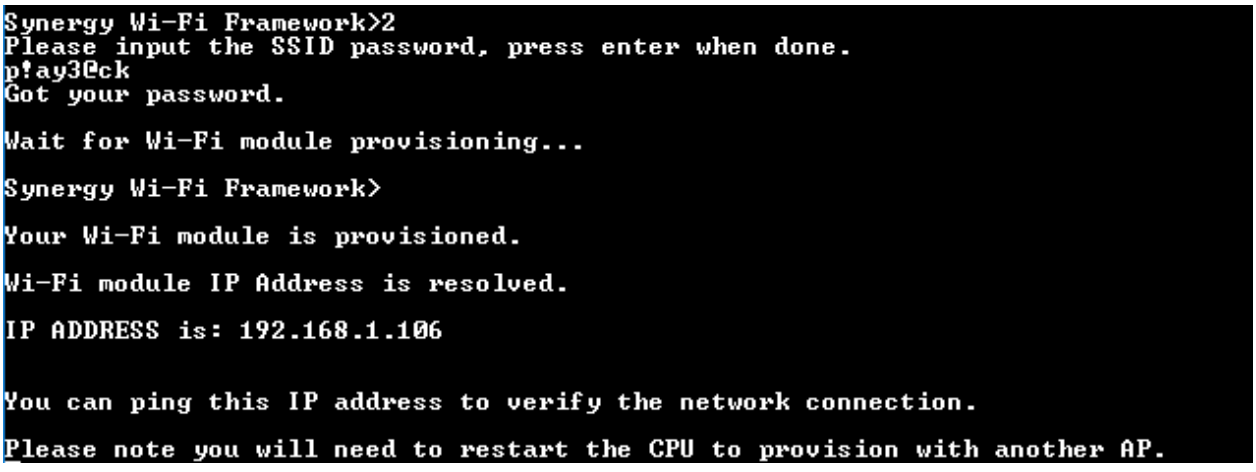

図 23 プロビジョニングの成功

プロビジョニングに失敗した場合、以下のエラーメッセージが表示されます。デバイスを再起動し、適切なクレデンシャル（credential、資格情報）を入力します。

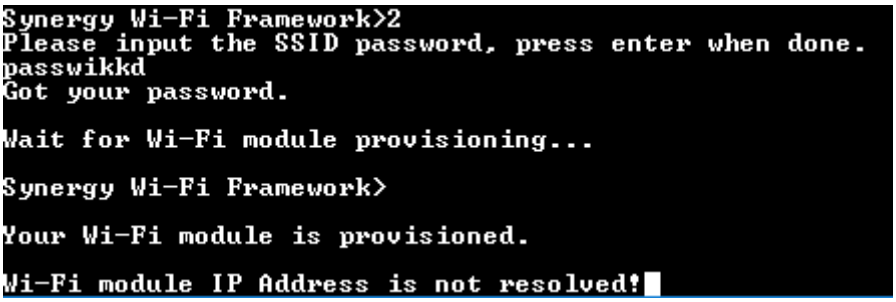

図 24　プロビジョニングの失敗

プロビジョニングが成功した場合、プロビジョニング成功時に表示された IP アドレスに対して ping を実行します。この場合、IP アドレスは 192.168.1.106 です。

この作業を実行するには、PC がキットと同じアクセスポイントに接続されていることを確認してください。

コマンドウィンドウを開き、以下の画像に示すようにコマンドを入力します

```
ping <IP address>
```

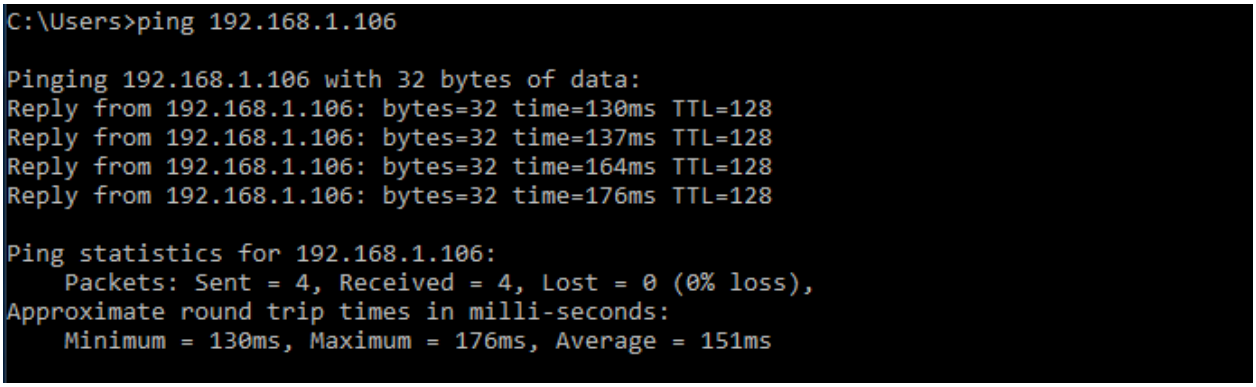

図 25　Wi-Fi モジュールへの ping

Web サイトおよびサポート

以下の URL で、Synergy プラットフォームの詳細の確認、関連するドキュメントのダウンロード、サポートの活用ができます。

Synergy ソフトウェア	renesassynergy.com/software
Synergy ソフトウェアパッケージ	renesassynergy.com/ssp
ソフトウェアアドオン	renesassynergy.com/addons
ソフトウェア用語集	renesassynergy.com/softwareglossary
開発ツール	renesassynergy.com/tools
Synergy ハードウェア	renesassynergy.com/hardware
マイクロコントローラ	renesassynergy.com/mcus
MCU 用語集	renesassynergy.com/mcuglossary
パラメトリック検索	renesassynergy.com/parametric
キット	renesassynergy.com/kits
Synergy ソリューション Gallery	renesassynergy.com/solutionsgallery
パートナープロジェクト	renesassynergy.com/partnerprojects
アプリケーションプロジェクト	renesassynergy.com/applicationprojects
セルフサービスサポートリソース:	
ドキュメント	renesassynergy.com/docs
ナレッジベース	renesassynergy.com/knowledgebase
フォーラム	renesassynergy.com/forum
トレーニング	renesassynergy.com/training
ビデオ	renesassynergy.com/videos
Web チケット	renesassynergy.com/support

改訂履歴

Rev.	発行日	説明	
		ページ	ポイント
1.00	2019.04.04	―	・日本語版初版 ・英語版（R11AN0342EU0100 Rev.1.00、2018.10.1 発行）を翻訳

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、
   家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、
   金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注2. 本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。

(Rev.4.0-1 2017.11)

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/