Renesas Synergy™ プラットフォーム

セルラーフレームワーク

R30AN0311JU0103
Rev.1.03
2018.08.07

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)

本アプリケーションノートにより、ユーザが独自に設計したシステムでセルラーフレームワークモジュール(Cellular Framework Module)を使用することが可能になります。本アプリケーションノートの内容を理解すると、ユーザ独自のシステムへの本モジュールの追加、ターゲットアプリケーション向けの設定が行え、さらに付属のアプリケーションコードを参照にしてコードを作成することが可能になります。他のアプリケーションのより詳細なAPI記述等関する参考情報は、本書の参考情報の項に記載されているRenesas Synergy™ウェブやナレッジベースから入手することが可能です。これは、本モジュールの高度な利用方法について解説しており、より複雑な設計を行う上で重要な情報になります。

セルラーフレームワークモジュールは、SSPアプリケーションフレームワーク(SSP Application Framework)でのセルラーモデム統合のための上位アプリケーション層インタフェースであり、データ通信用のセルラーネットワークのプロビジョニング(provision)、設定(configure)、および通信を行うAPIセットを提供します。セルラーフレームワークはSSPアプリケーションフレームワーク（コンソールフレームワーク(console framework)）を使用して、内部的にATコマンドを使用することによりシリアルインタフェースでセルラーモデム(Cellular Modem)と通信します。また、SSPアプリケーションフレームワークはデータ通信用のシリアルインタフェースでシリアルデータパイプを作成し、NetX™で提供されるPPP WANプロトコル(protocol)を利用します。ソケット、NetXアプリケーションプロトコル、およびIoTプロトコル（MQTT、COAPなど）を使用して、このワイドエリアネットワーク（WAN）リンクでTCP/IP通信を確立することが可能です。

セルラーフレームワークは、特定のセルラーハードウェアモジュールのオンチップ（セルラーハードウェアモジュール内部）に存在するTCP/IPスタックと通信するフレームワークレベルのソケットAPI(Socket API)も提供するため、ソケットAPIを使用してインターネットネットワークと通信します。

必要リソース(Required Resources)

セルラーフレームワークアプリケーションサンプル(Cellular Framework Application example)をビルドして実行するには、以下のものが必要です。

- Renesas Synergy™ PK-S5D9キット
- e² studio ISDE v5.4.0.023以降またはRenesas Synergy™用IAR組込みWorkbench® v7.71.3以降
- Synergyソフトウェアパッケージ（SSP）1.3.2以降またはSynergyスタンドアロンコンフィグレータ（SSC）5.4.0.023以降
- SEGGER J-Link®およびそれに対応するUSBドライバ
- Windows® 7用Renesas Synergy USB CDCドライバ（バンドルに付属）
- Tera Termコンソールアプリケーションまたは同等のアプリケーションがインストールされたWindows 7/10のテスト用PC
- PMODアダプタモジュール付きNimbeLink™ LTE CAT3セルラー モデム（型名NL-SW-LTE-TSVG（北米向け））
- PMODアダプタモジュール付きNimbeLink™ LTE CAT1セルラー モデム（型名NL-SW-LTE-GELS3-B（北米向け））
- サービス事業者（ベライゾン）のSIMカード
  （本ドキュメントはフレームワークの概念を説明する参考資料であり、上記記載の具体例（モジュールおよびサービス事業者）は日本で使用できません。
- マイクロUSBケーブル
- 必要なすべてのRenesas（SSP）（Renesas Synergy™ウェブ（https://www.renesas.com/ja-jp/products/synergy.html）からダウンロードします）

前提条件と対象ユーザ(Prerequisites and Intended Audience)

本アプリケーションノートは、Synergy e² studio ISDEまたはIAR Embedded Workbench for Renesas Synergy(IAR EW for Synergy)、およびSynergyソフトウェアパッケージ（SSP）に関して、ある程度経験があるユーザを前提としています。本アプリケーションノートの手順に進む前に、*SSPユーザーズマニュアル(SSP User Manual)*の手順に従って**Blinky**プロジェクトをビルドおよび実行してください。そうすることで、e² studioおよびSSPに慣れ、ボードへのデバッグ接続が適切に機能していることを確認できるようになります。

さらに、本アプリケーションノートは、セルラーネットワーク、3GPP規格、およびその通信プロトコルに関して、ある程度知識があるユーザを前提としています。また、TCP/IPとその階層アーキテクチャ、LAN技術、WAN技術、BSDソケット通信などに関しても理解しているユーザを前提としています。

対象ユーザは、S3、S5、S7 Synergy MCUシリーズを使用してセルラーフレームワークモジュールでアプリケーションを開発しようとしているユーザです。

目次

要旨(Introduction) 1

必要リソース(Required Resources) 2

前提条件と対象ユーザ(Prerequisites and Intended Audience) 2

1. セルラーフレームワークモジュールの概要 (Cellular Framework Module Overview) 6

1.1 セルラーフレームワークの主要ブロック (Major blocks of the Framework) 7

1.1.1 セルラーフレームワークアプリケーション層インタフェース (Cellular framework application layer interface) 8

1.1.2 ネットワークスタック抽象化層 (Network stack abstruction layer) 8

1.1.3 ソケットインタフェース層 (Socket interface layer) 8

1.1.4 PPPスタック (PPP stack) 8

1.1.5 セルラー デバイスドライバ(Cellular device driver) 8

2. セルラーフレームワークモジュールの動作概要 (Cellular Framework Module Operation Overview) 8

2.1 セルラーフレームワークモジュールの初期化 (Cellular framework module initialization) 9

2.2 セルラーハードウェアモジュールのプロビジョニング (Cellular hardware module provisioning) 10

2.3 ソケットインタフェースによるアプリケーションフロー制御 (Application flow control using socket interface) 11

2.4 セルラーパケット送信(Cellular packet transmisson) 12

2.5 セルラーパケット受信 (Cellular Packet Reception) 13

2.6 セルラーフレームワークモジュールの重要な動作上の注意事項と制限 (Cellular framework module important operation notes and limitations) 14

3. セルラーフレームワークモジュールAPIの概要 (Cellular Framework Module APIs Overview) 15

3.1 セルラーフレームワークAPI (Cellular Framework API) 20

3.1.1 open 20

3.1.2 close 20

3.1.3 provisioningGet 20

3.1.4 provisioningSet 21

3.1.5 infoGet 21

3.1.6 statisticsGet 22

3.1.7 transmit 22

3.1.8 versionGet 22

3.1.9 networkConnect 22

3.1.10 networkDisconnect 23

3.1.11 networkStatusGet 24

3.1.12 simPinSet 24

3.1.13 simLock 25

3.1.14 simUnlock 25
3.1.15 simIDGet 25
3.1.16 fotaCheck 25
3.1.17 fotaStart 27
3.1.18 fotaStop 28
3.1.19 reset 28
3.2 セルラー フレームワークソケットインタフェースAPI (Cellular Framework Socket Interface API) 29
3.2.1 open 30
3.2.2 close 30
3.2.3 versionGet 30
3.2.4 socket 31
3.2.5 close 31
3.2.6 bind 31
3.2.7 listen 32
3.2.8 connect 32
3.2.9 accept 32
3.2.10 send 33
3.2.11 recv 33
3.2.12 sendto 34
3.2.13 recvfrom 34
3.2.14 setsockopt 35
3.2.15 getsockopt 35
3.2.16 select 36

4. アプリケーションにセルラー フレームワークモジュールを組み込む (Including the Cellular Framework Module in an Application) 37
4.1 セルラー フレームワークモジュールをNetXにTCP/IPスタックとして組み込む (Including the Cellular Framework Module with NexX as TCP/IP stack) 37
4.2 セルラー フレームワークモジュールをTCP/IPのオンチップスタックに組み込む (Including the Cellular Framework Module with On-chip Stack for TCP/IP) 40

5. セルラーフレームワークモジュールの設定(Configuring the Cellular Framework Module) 41
5.1 セルラー フレームワークをNetXでTCP/IPスタックとして設定 (Configuring Cellular Framework with NetX as TCP/IP Stack) 41
5.1.1 セルラーフレームワークモジュールCAT3/CAT1モデムデバイスドライバ設定 (Cellular Framework Module CAT3/CAT1 Modem Device configuration) 44
5.1.2 セルラー フレームワークモジュールパケットプール設定 (Cellular Framework Module Packet Pool configuration) 45
5.1.3 セルラーフレームワークモジュールUART設定(Cellular Framework Module UART configuration) 47
5.2 BSDソケットによるセルラー フレームワークの設定 (Configuring Cellular Framework with BSD Socket) 49
5.2.1 セルラー フレームワークモジュールオンチップスタック設定

(Cellular framework module on-chip stack configuration) 49
5.2.2 セルラーフレームワークモジュールCAT3/CAT1モデムデバイス設定
(Cellular Framework Module CAT3/CAT1 modem device configuration) 49
5.2.3 セルラーフレームワークモジュールUART設定(Cellular Framework Module UART configuration) 49
5.2.4 セルラー フレームワークモジュール依存層設定
(Cellular Framework Module dependency layer configuration) 49

6. アプリケーションのセルラーフレームワークモジュールの使用
(Using the Cellular Framework Module in an Application) 50

7. セルラーフレームワークモジュールアプリケーションプロジェクト
(The Cellular Framework Module Application Project) 50
7.1 セルラー アプリケーションソフトウェアアーキテクチャの概要
(The cellular application software architecture overview) 51
7.1.1 コンソールスレッド(Console thread) 51
7.1.2 セルラーアプリケーションスレッド(Cellular Application Thread) 53
7.1.3 セルラーフレームワークモジュールコードの概要(Cellular framework module code overview) . 67

8. セルラー フレームワークモジュールアプリケーションプロジェクトの実行(Running the Cellular Framework Module Application Project) 70
8.1 セルラーハードウェアモジュールのアクティベーション(有効化)と設定の詳細 (Cellular harware module activation and setup details) 70
8.2 PK-S5D9ボード設定の詳細 (PK-S5D9 Synergy MCU board setup details) 71
8.3 サンプルアプリケーションの実行(Run the sample application) 71
8.4 USB CDCデバイスドライバのインストール(Install the USB CDC device driver) 74

9. セルラーフレームワークモジュールの終了(Cellular Framework Module Conclusion) 74

10. セルラーフレームワークモジュールの次の手順(Cellular Framework Module Next Steps) 75

11. 参考情報(Reference Information) 75

1. セルラーフレームワークモジュールの概要 (Cellular Framework Module Overview)

このセルラーフレームワークは、ベンダー固有のインタフェースコード(Interface code)を記述しなくても、アプリケーションが様々なベンダーのセルラーハードウェアモジュール(Cellular harware module)と通信するための汎用インタフェース(generic interface)を提供します。セルラーフレームワークは主にAPIの共通セットで構成されます。この共通セットは、ネットワークスタック(networking stack)とのインタフェース、および各種セルラーハードウェアモジュールに対応する汎用インタフェースドライバとのインタフェースに使用されます。本項では、セルラーフレームワークの基本ブロックと主な機能について説明します。これにより、ユーザが対象とするセルラーアプリケーションをセルラーフレームワークで開発できるかどうかを判断することが可能になります。

このアプリケーションでは、ロウレベルのベンダードライバコードがセルラーフレームワークによって抽象化されています。汎用APIと抽象化により、セルラーハードウェアモジュール向けに開発されたアプリケーションは別のセルラーハードウェアモジュールとともに簡単に移植することが可能です。ネットワークスタックNetX もネットワークソフトウェア抽象化層 (NSAL) によりフレームワークに統合されています。

1.1　セルラーフレームワークの主要ブロック (Major blocks of the Framework)

Synergyセルラーフレームワークは以下の論理ブロック(logical block)で構成されます。

- Synergyセルラーフレームワークアプリケーションインタフェース
- NetX TCP/IPスタックに対応するネットワークスタック抽象化層 (NSAL)
- セルラーデバイスドライバ(Cellular Device Driver)
- オンチップネットワークスタックをサポートするセルラーハードウェアモジュールとのインタフェースに使用するBSDソケット互換API(BSD Socket compatible API)
- Synergyソフトウェアパッケージ (SSP) HALインタフェース

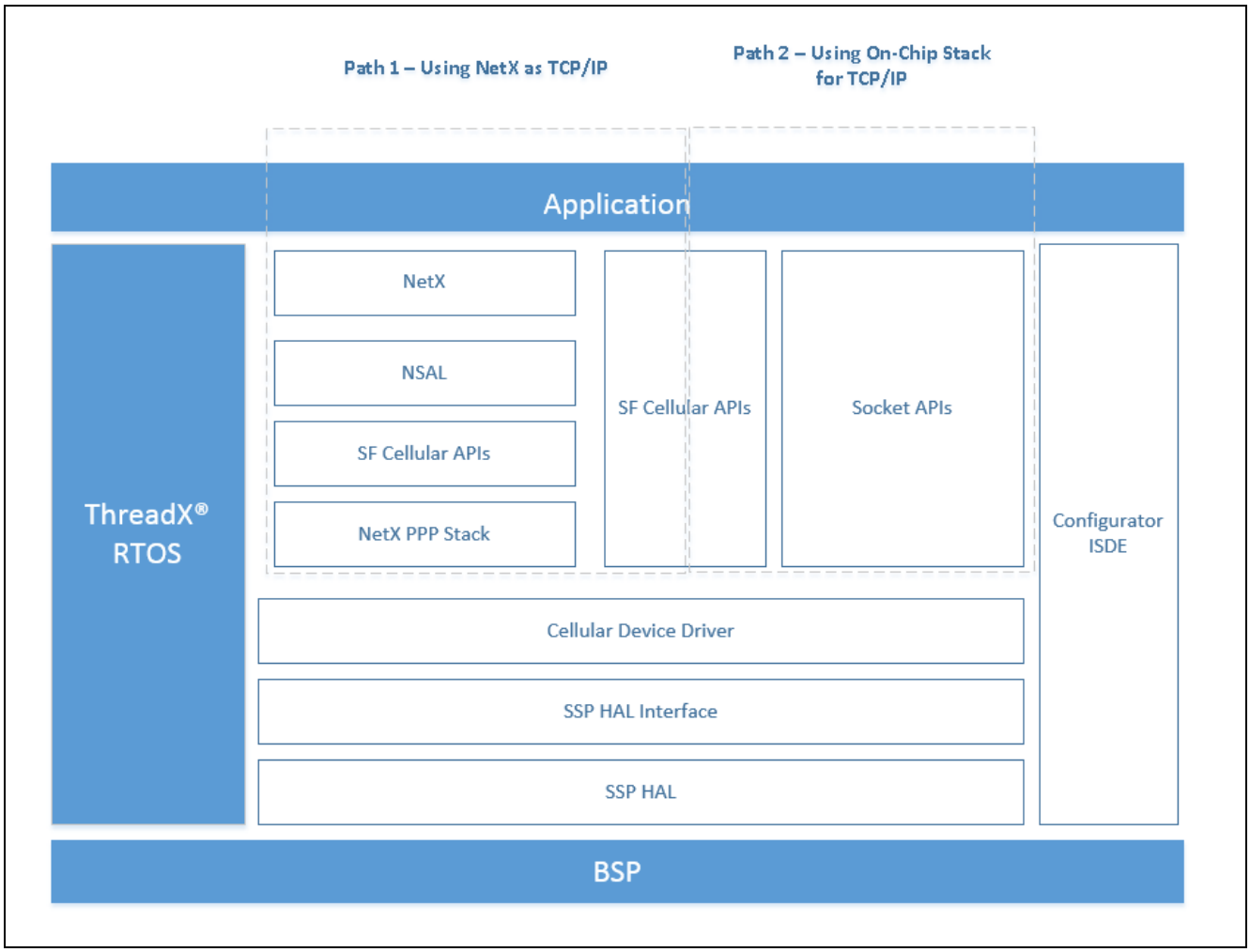

図1　セルラーフレームワークモジュール編成とインタフェース層

1.1.1 セルラーフレームワークアプリケーション層インタフェース (Cellular framework application layer interface)

セルラーフレームワークは、アプリケーションがセルラーハードウェアモジュールの設定(configure)、プロビジョニング(provision)、および通信を行うためのインタフェースの共通セット(common set)を提供します。こうした汎用インタフェース(generic interface)により、ユーザはSynergy MCUを使用してセルラーベースのアプリケーションを開発することが可能です。セルラーハードウェアモジュールには、3GPP規格によって規定された様々な設定パラメータがありますが、個別のデバイスドライバやセルラーチップセット／モジュールは必ずしもすべてのパラメータの設定をサポートしていない可能性があります。モジュールが機能するためには、ネットワークオペレータ(network operator)、アクセスポイント名 (APN) 、およびセキュリティー証明書(security credential)が最低限必要です。

1.1.2 ネットワークスタック抽象化層 (Network stack abstruction layer)

セルラーフレームワークは、ネットワークスタック抽象化層 (NSAL) を提供します。NSALは、WANリンクでのデータ通信に使用される (PPP) スタックによってNetXとセルラードライバ(Cellular driver)を接続する層です。

1.1.3 ソケットインタフェース層 (Socket interface layer)

セルラーフレームワークは、セルラーハードウェアモジュール(Cellular hardware module)に存在するオンチップネットワークスタック(on-chip network stack)とアプリケーションが連携するためのソケットレベルAPI(Socket level API)を提供します。この場合、オンチップネットワークスタックとソケットインタフェースをサポートするために、セルラー ハードウェアモジュール／ドライバが必要です。アプリケーションは、こうしたAPIを使用する際に、セルラーハードウェアモジュールのオンチップネットワークスタックを使用し、NSALまたはNetXとそのソケットAPIを使用しません。また、Synergy MCUグループで動作しているネットワークスタックも使用しません。

1.1.4 PPPスタック (PPP stack)

ポイントツーポイントプロトコル (PPP:Point to point protocol) は、データ通信で広く利用されているWANプロトコルです。NetXはSSPの一環としてPPPスタックサポートを提供します。NSALはPPPスタックを利用して、シリアルインタフェースでセルラーサービス事業者のネットワークと通信します。PPPは、PAP/CHAPのような認証方式(authentication method)に対応するオプションを提供します。こうした認証機構はオプションですが、NSALはフレームワークAPIを利用して、セルラーハードウェアモジュールのデータを送受信します。NSALにより、ネットワークスタック固有の変更を行わずにセルラーデバイスドライバ(cellular device driver)を再利用することが可能です。

1.1.5 セルラー デバイスドライバ(Cellular device driver)

セルラーフレームワークはATコマンドセット(AT command)を使用して、シリアルドライバ(serial driver)でセルラーモデム(Cellular modem)と連携します。モデムと連携するために使用されるシリアルインタフェースはUARTです。フレームワークのデフォルトで使用されるUART速度は最大115200ビット／秒です。

2. セルラーフレームワークモジュールの動作概要 (Cellular Framework Module Operation Overview)

上記の図1ではユーザアプリケーションの観点から、セルラーモデムとそれで利用可能なサポートに応じて、フレームワークを使用した通信(communication)に2種類の経路(path)が使用できることを示します。ホストでTCP/IPスタックを使用するためのオプションを提供するモジュールもあれば、セルラーモデム自体に存在するTCP/IPスタックを使用するためのオプションを提供するモジュールもあります。場合によっては、セルラーハードウェアモジュールが両方のオプションを提供します。ホストTCP/IPスタック (NetX) が使用されると、アーキテクチャ(図1)に示すようにNetX、NSAL、PPPの各論理層(logical layer)が使用されます。オンチップスタック(on-chip stack)が使用されると、セルラーモデムに存在するTCP/IPスタックとの通信にソケットAPIが使用されます。ただし、ユーザは両方を同時に使用することはできません。

2.1 セルラーフレームワークモジュールの初期化 (Cellular framework module initialization)

図2は、ユーザが指定したセルラーモデム向け設定(configuration)による初期化(initialization)の制御フローを示します。この時、NetX nx_ip_createが呼び出されます。この関数は、リンクレベル(link level)の初期化に対応してセルラーハードウェアモジュール(cellular hardware module)を初期化するNSALドライバエントリ関数(NSAL driver entry function)を内部的に呼び出します。さらに、モジュールをプロビジョニングし、PPPインタフェースでネットワーク接続を確立します。

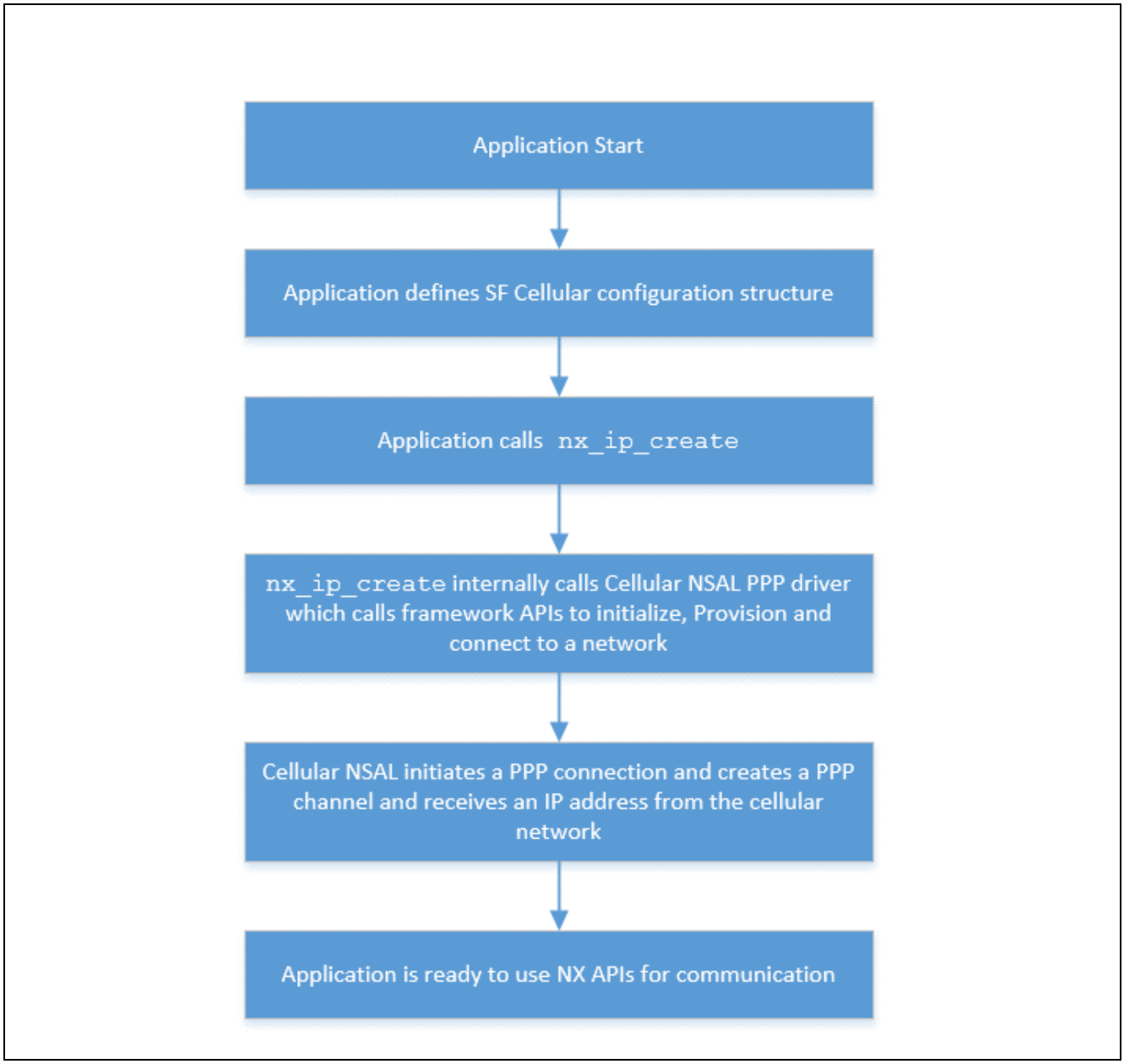

図2　セルラーフレームワークモジュールの初期化シーケンス

2.2 セルラーハードウェアモジュールのプロビジョニング (Cellular hardware module provisioning)

プロビジョニングに使用される引数(arguments)は制御構造体(control structure)と、ユーザ設定パラメータ(user configured parameter)で指定されます。セルラーモデムのプロビジョニングとしてユーザが設定するパラメータは認証(authentication)、APN(access point name)、ユーザ名、およびパスワードです。セルラーフレームワークの場合、コールバック関数(callback function)がモジュールをプロビジョニングします。ユーザはプロビジョニングの際に、APN名、認証タイプに加えて、モジュールのプロビジョニングに必要な詳細を指定する必要があります。

2.3　ソケットインタフェースによるアプリケーションフロー制御 (Application flow control using socket interface)

以下の図3は、セルラーソケットインタフェース(Cellular Socket interface)でオンチップスタック経路(on-chip stack path)を使用するフローを示しています。

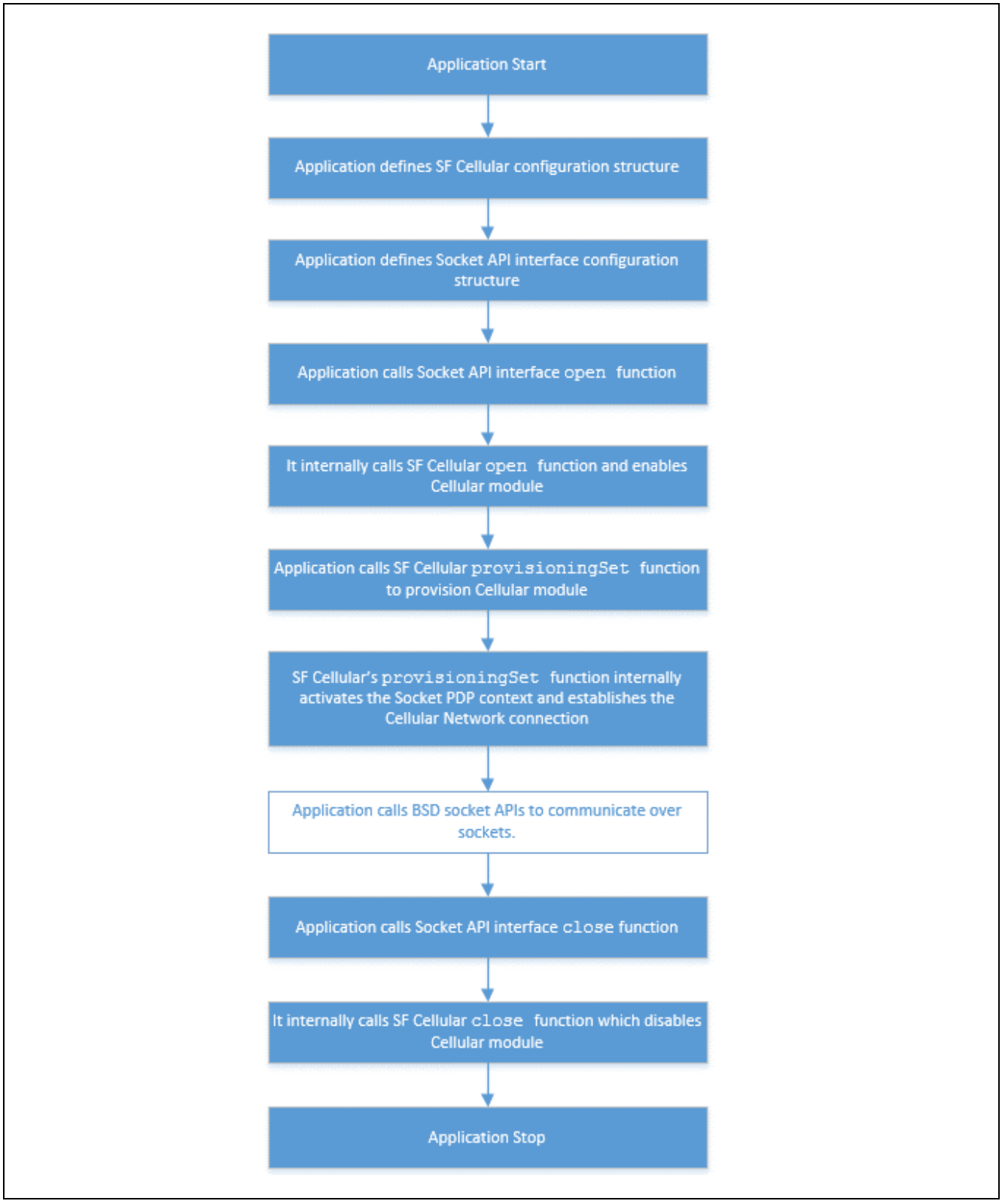

図3　セルラーフレームワークモジュールのソケットインタフェース

2.4　セルラーパケット送信(Cellular packet transmisson)

下記の図4のフロー図は、NetXアプリケーションのパケット送信で実行される手順のシーケンスを示します。

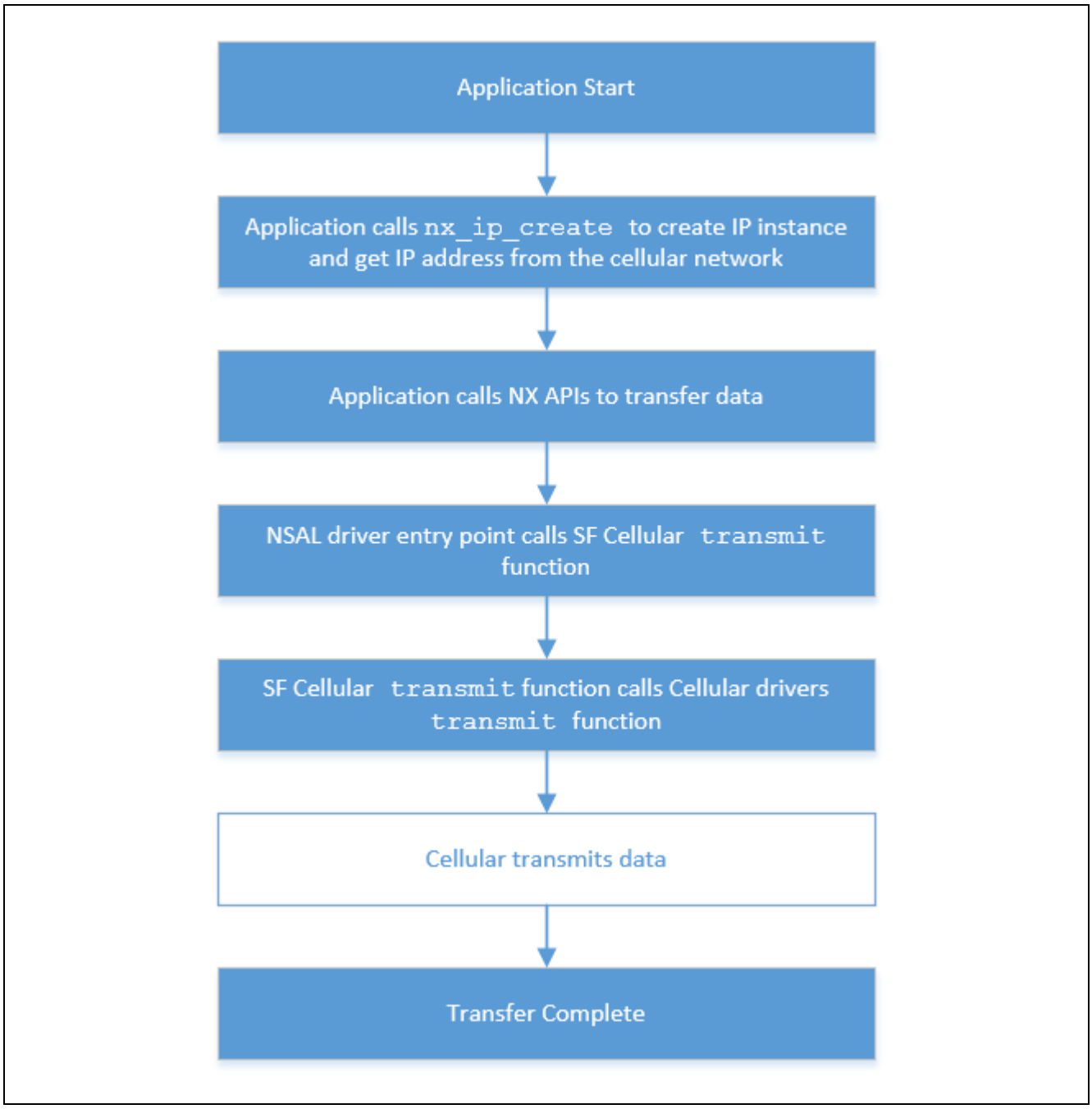

図4　セルラーフレームワークのパケット送信シーケンス

2.5　セルラーパケット受信 (Cellular Packet Reception)

下記の図5は、NetXによるセルラーフレームワークのパケット受信のフローを示しています。データがシリアルインタフェースで受信されたときは、処理スレッド(processing thread)がコールバック関数(callback function)をトリガします。続いて、コールバック関数がデータを処理し、次の処理のためにNetXの層に送信します。

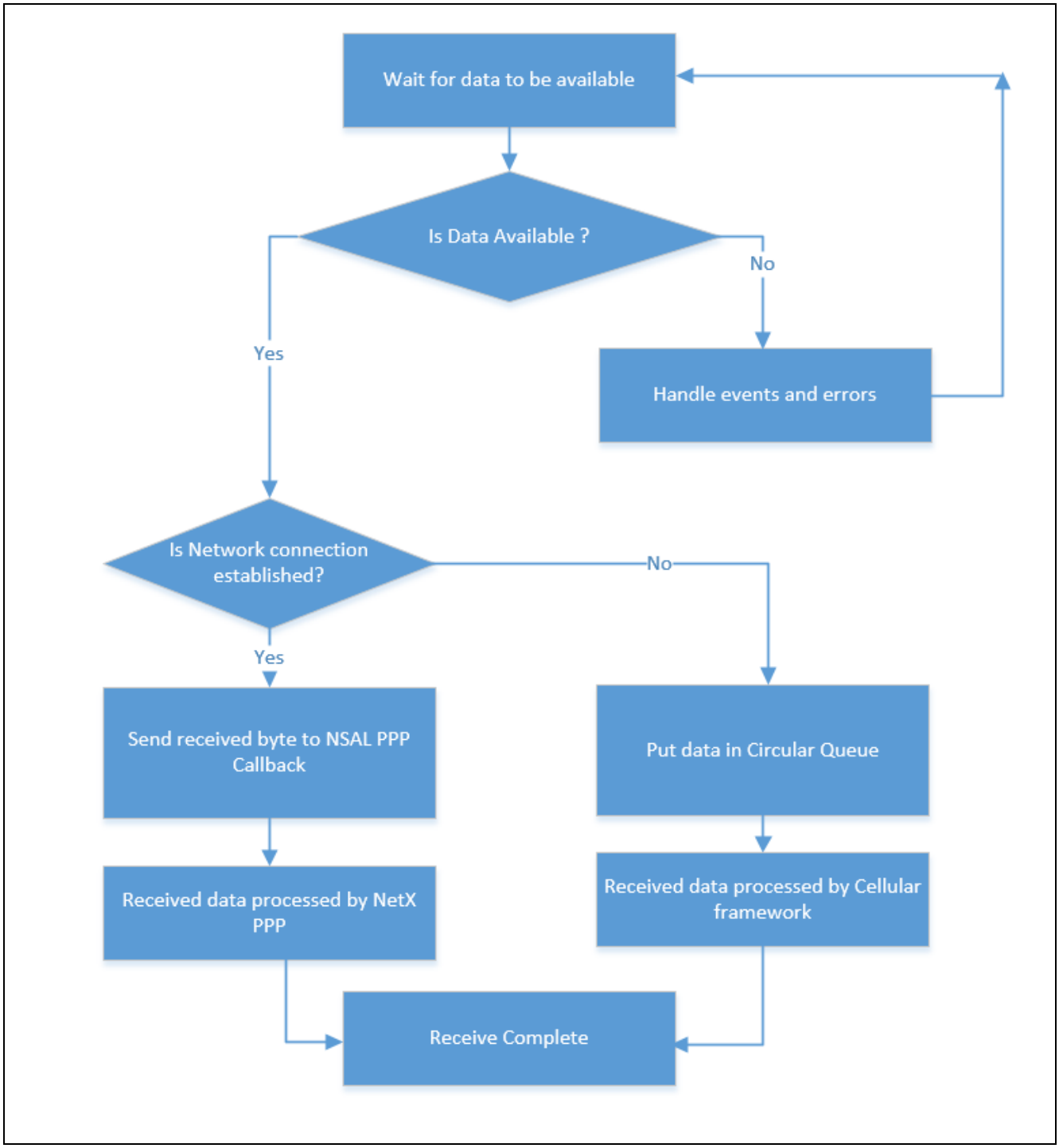

図5　セルラーフレームワークのパケット受信シーケンス

2.6 セルラーフレームワークモジュールの重要な動作上の注意事項と制限 (Cellular framework module important operation notes and limitations)

- 現在のフレームワークは、NimbeLink CAT3およびCAT1セルラー ハードウェアモジュールのみをサポートしています。
- 無線でのファームウェアアップグレード (FOTA) は、NimbeLink CAT3およびCAT1セルラーハードウェアモジュールでサポートされていません。

本モジュールに関するその他の動作上の制限については、最新の*SSPリリースノート(SSP Release Note)*を参照してください。

3. セルラーフレームワークモジュールAPIの概要(Cellular Framework Module APIs Overview)

セルラーフレームワークモジュールは汎用インタフェース(generic interface)を使用して、モジュール下層部と連携するためのAPIセットを定義します。以下は、ドライバおよびセルラーハードウェアモジュールと通信するために、セルラーフレームワークで使用されるAPIです。ほとんどのセルラー フレームワークAPIは、インスタンス(instance)作成時に作成されるAPIである**p_ctrl**および**p_cfg**データ構造体を使用します。APIについて効率よく理解を深めるために、構造体とその詳細の一部をここで説明します。インスタンス構造体の詳細については、*SSPユーザーズマニュアル(SSP User Manual)*を参照してください。
このインスタンス構造体は、セルラーフレームワークインタフェースのインスタンスを使用するために必要なすべての要素を包含しています。ほとんどのAPIは、アプリケーションで使用される場合、制御構造体(control structure) と設定構造体(config structure)をパラメータとして使用します。

```
typedef struct st_sf_cellular_instance
{
```

sf_cellular_ctrl_t	* p_ctrl;	セルラー フレームワークインスタンスの制御構造体へのポインタ
sf_cellular_cfg_t	const * p_cfg;	セルラー フレームワークインスタンスの設定構造体へのポインタ
sf_cellular_api_t	const * p_api;	セルラー フレームワークインスタンスのAPI構造体へのポインタ

```
} sf_cellular_instance_t;
```

下記の表は、設定構造体に含まれるセルラー設定パラメータ(Cellular configuration parameter)を示しています。これらのパラメータの一部はコンフィグレータで設定されます。この設定情報は、APIの呼び出し時にドライバ下層部で使用されます。

```
typedef struct st_sf_cellular_cfg
{
```

sf_cellular_op_select_mode_t	op_select_mode	セルラーオペレータ選択モード。動作モード選択に利用できる4種類のオプションがあります。 • 自動（自動オペレータ選択） • 手動（手動オペレータ選択） • 登録解除(De-register)（ネットワークからの登録解除） • 手動フォールバック（自動へのフォールバック機能付き手動）
sf_cellular_op_t	op	セルラーオペレータ。オペレータ選択モードが手動モードの場合に有効です。これは、セルラーオペレータ名と名前の形式を保持する設定構造体に含まれる構造体です。
uint16_t	num_pref_ops	pref_ops配列の優先セルラーオペレータの数。ユーザは優先オペレータリストを保持することが可能です。
sf_cellular_op_t	pref_ops [SF_CELLULAR_MAX_	優先オペレータを示す構造体の配列

	PREFFERED_OPERATOR_COUNT]	
sf_cellular_timezone_update_mode_t	tz_upd_mode	タイムゾーン更新モードポリシー。これはタイムゾーン自動更新（有効／無効）のオプションです。
uint8_t	* p_sim_pin	SIM PIN。SIM PINをアンロックする必要がある場合、ここで設定することが可能です。
uint8_t	* p_puk_pin	PUK。パーソナルアンロックキー（PUK）は、紛失または忘却した暗証番号（PIN）をリセットするために、3GPP携帯電話で使用されます。ほとんどのセルラーモデムはPIN保護機能を備えています。
ssp_err_t	(* p_prov_callback)(sf_cellular_callback_args_t * p_args)	プロビジョニングコールバック関数へのポインタ（NSALで使用）
void	(* p_recv_callback)(sf_cellular_callback_args_t * p_args)	これはNetXで使用される受信コールバック関数です。この関数は、セルラーハードウェアモジュールからデータパケットを取得し、次の処理のためにNetXに渡します。
void	const * p_context	コールバック関数に渡されるユーザ定義コンテキスト
void	const * p_extend	拡張設定に対応するインスタンス固有の設定
sf_cellular_at_cmd_set_t	const * p_cmd_set	インスタンス固有のATコマンドセットへのポインタ

} **sf_cellular_cfg_t**

```
typedef struct st_sf_cellular_ctrl
{
```

void	* p_driver_handle	セルラーデバイスドライバの下層部で必要な情報を格納します。

} sf_cellular_ctrl_t

セルラー プロビジョニング情報構造体(Cellular provisoning information structure)

```
typedef struct st_sf_cellular_provisioning
{
```

uint8_t	Apn [SF_CELLULAR_MAX_STRING_LEN]	アクセスポイント名
sf_cellular_auth_type_t	auth_type	認証タイプ：PAP/CHAP
uint8_t	Username [SF_CELLULAR_MAX_STRING_LEN]	認証に使用するユーザ名
uint8_t	Password [SF_CELLULAR_MAX_STRING_LEN]	認証に使用するパスワード
sf_cellular_airplane_mode_t	airplane_mode	機内モード(Airplane mode)
uint8_t	context_id	接続に使用するコンテキストID
sf_cellular_pdp_type_t	pdp_type	コンテキストのPDPタイプ

```
} sf_cellular_provisioning_t
```

セルラー情報構造体の情報(Cellular info structure information)

```
typedef struct st_sf_cellular_info
{
```

uint8_t	mfg_name[SF_CELLULAR_MFG_NAME_LEN]	メーカ名
uint8_t	chipset[SF_CELLULAR_CHIPSET_LEN]	セルラー チップセット／ドライバ情報を示す文字列へのポインタ
uint8_t	fw_version[SF_CELLULAR_FWVERSION_LEN]	セルラー ファームウェアバージョン
uint8_t	imei[SF_CELLULAR_IMEI_LEN]	IMEI番号
uint16_t	rssi	受信信号強度表示
uint16_t	ber	ビットエラーレート

```
} sf_cellular_info_t
```

セルラー インスタンスの統計カウンタとエラーカウンタ(The statistics and error counters for the cellular instance)

```
typedef struct st_sf_cellular_stats
{
```

uint32_t	rx_bytes	正常受信バイト数
uint32_t	tx_bytes	正常送信バイト数
uint32_t	rx_err	エラー受信バイト数
uint32_t	tx_err	エラー送信バイト数

```
} sf_cellular_stats_t
```

セルラー ネットワークステータス構造体(The Cellular network status structure)

```
typedef struct st_sf_cellular_network_status
{
```

uint16_t	country_code	国コード
uint16_t	operator_code	オペレータコード
uint16_t	rssi	RSSI
uint8_t	cid[SF_CELLULAR_CID_LEN]	セルID
uint8_t	imsi[SF_CELLULAR_IMSI_LEN]	IMSI
uint8_t	op_name[SF_CELLULAR_MAX_OPERATOR_NAME_LEN]	オペレータ名
uint8_t	service_domain	サービスドメイン(SErvice Domain)
Uint8_t	active_band	アクティブバンド

```
} sf_cellular_network_status_t
```

セルラー ハードウェアモジュールリセットタイプ (Cellular Hardware Module reset type)

```
typedef enum e_sf_cellular_reset_type
{
```

SF_CELLULAR_RESET_TYPE_SOFT	ATコマンドによるソフトリセットモジュール
SF_CELLULAR_RESET_TYPE_HARD	リセット端子のトグルによるハードリセットモジュール

```
} sf_cellular_reset_type_t
```

ssp_err_t (SSPエラーコード) : これらは、APIの使用時にSSPから返されるエラーコードです。以下のリストは、セルラー フレームワーク固有のエラーコードです。詳細およびすべてのSSPエラーコードについては、*SSPユーザーズマニュアルを参照してください。*

SSP_ERR_CELLULAR_CONFIG_FAILED	セルラー モジュールの設定に失敗しました。
SSP_ERR_CELLULAR_INIT_FAILED	セルラー モジュールの初期化に失敗しました。
SSP_ERR_CELLULAR_TRANSMIT_FAILED	送信に失敗しました。
SSP_ERR_CELLULAR_FW_UPTODATE	ファームウェアは最新です。
SSP_ERR_CELLULAR_FW_UPGRADE_FAILED	ファームウェアアップグレードに失敗しました。
SSP_ERR_CELLULAR_FAILED	セルラー処理に失敗しました。

3.1　セルラーフレームワークAPI (Cellular Framework API)

3.1.1　open

説明：データ転送のために、セルラーハードウェアモジュールを初期化し、有効にします。ドライバの初期設定、ドライバリンクの有効化、割り込みの許可、およびデータ転送に向けたデバイスの準備を行います。

パラメータ(Parameter)		
名前(Name)	方向(Direction)	内容(Description)
p_ctrl	In	Link-> st_sf_cellular_ctrl
p_cfg	In	Link-> st_sf_cellular_cfg
戻り値(Return value)	Link-> ssp_err_t (SSP エラーコード):	
関数プロトタイプ (Function Prototype)	ssp_err_t (*open) (sf_cellular_ctrl_t * p_ctrl,sf_cellular_cfg_t const * const p_cfg)	

3.1.2　close

説明：セルラーハードウェアモジュールを初期化せず(de-initialize)に、すべての通信を無効(disable)にします。PDPコンテキストを無効にします。

パラメータ		
名前	方向	内容
p_ctrl	In	Link-> st_sf_cellular_ctrl
戻り値	Link-> ssp_err_t (SSPエラーコード) :	
関数プロトタイプ	ssp_err_t (*close) (sf_cellular_ctrl_t * p_ctrl)	

3.1.3　provisioningGet

説明：セルラーハードウェアモジュールに関するプロビジョニング情報を入手します。

パラメータ		
名前	方向	内容
p_ctrl	In	Link-> st_sf_cellular_ctrl
p_cellular_provisioning	Out	See-> st_sf_cellular_provisioning
戻り値	Link->ssp_err_t (SSPエラーコード) :	
関数プロトタイプ	ssp_err_t (* networkConnect) (sf_cellular_ctrl_t * const p_ctrl)	

3.1.4 provisioningSet

説明：セルラーハードウェアモジュールに関するプロビジョニング情報を設定します。

パラメータ		
名前	方向	内容
p_ctrl	In	Link-> st_sf_cellular_ctrl
p_cellular_provisioning	In	See:-> st_sf_cellular_provisioning
戻り値	Link->ssp_err_t (SSPエラーコード) :	
関数プロトタイプ	ssp_err_t (* provisioningSet) (sf_cellular_ctrl_t * const p_ctrl, sf_cellular_provisioning_t const * const p_cellular_provisioning)	

3.1.5 infoGet

説明：セルラーハードウェアモジュールの情報を読み出します。

パラメータ		
名前	方向	内容
p_ctrl	In	Link-> st_sf_cellular_ctrl
p_cellular_info	Out	See:->st_sf_cellular_info
戻り値	Link->ssp_err_t (SSPエラーコード) :	
関数プロトタイプ	ssp_err_t (* infoGet) (sf_cellular_ctrl_t * const p_ctrl, sf_cellular_info_t * const p_cellular_info)	

3.1.6 statisticsGet

説明：セルラーハードウェアモジュールの統計情報(statistics information)を返します。

パラメータ		
名前	方向	内容
p_ctrl	In	Link-> st_sf_cellular_ctrl
p_cellular_device_stats	Out	Link-> st_sf_cellular_network_stats
戻り値	Link->ssp_err_t (SSPエラーコード) :	
関数プロトタイプ	ssp_err_t (* statisticsGet) (sf_cellular_ctrl_t * const p_ctrl, sf_cellular_stats_t * const p_cellular_device_stats)	

3.1.7 transmit

説明：送信のためにパケットバッファ(packet buffer)をPPPスタックに渡します。

パラメータ		
名前	方向	内容
p_ctrl	In	Link-> st_sf_cellular_ctrl
p_buf	In	Pointer to packet buffer to transmit
length	In	Length of packet buffer
戻り値	Link->ssp_err_t (SSPエラーコード) :	
関数プロトタイプ	ssp_err_t (* transmit) (sf_cellular_ctrl_t * const p_ctrl, uint8_t * const p_buf, uint32_t length)	

3.1.8 versionGet

説明：バージョンを取得し、指定されたポインタp_versionに格納します。

パラメータ		
名前	方向	内容
p_version	Out	バージョン番号を返すメモリ位置へのp_versionポインタは、APIおよびSSPコードのバージョン番号を取得します。
戻り値	Link->ssp_err_t (SSPエラーコード) :	
関数プロトタイプ	ssp_err_t (* versionGet)(ssp_version_t * const p_version)	

3.1.9 networkConnect

説明：データ接続を開始します。

パラメータ		
名前	方向	内容
p_ctrl	In	Link-> st_sf_cellular_ctrl
戻り値	Link->ssp_err_t (SSPエラーコード) :	
関数プロトタイプ	ssp_err_t (* networkConnect) (sf_cellular_ctrl_t * const p_ctrl)	

3.1.10 networkDisconnect

説明：データ接続を終了します。

パラメータ		
名前	方向	内容
p_ctrl	In	Link-> st_sf_cellular_ctrl
戻り値	Link->ssp_err_t (SSPエラーコード) ：	
関数プロトタイプ	ssp_err_t (* networkDisconnect) (sf_cellular_ctrl_t * const p_ctrl)	

3.1.11 networkStatusGet

説明：ネットワークステータス情報を取得します。

パラメータ		
名前	方向	内容
p_ctrl	In	Link-> st_sf_cellular_ctrl
p_network_status	Out	Link-> st_sf_cellular_network_stats
戻り値	Link->ssp_err_t (SSPエラーコード) ：	
関数プロトタイプ	ssp_err_t (* networkStatusGet) (sf_cellular_ctrl_t * const p_ctrl, sf_cellular_network_status_t * p_network_status)	

3.1.12 simPinSet

説明：SIM PINを設定します。

パラメータ		
名前	方向	内容
p_ctrl	In	Link-> st_sf_cellular_ctrl
p_old_pin	In	Pointer to char array containing current 4 digit pin
p_new_pin	In	Pointer to char array containing new 4 digit pin
戻り値	Link->ssp_err_t (SSPエラーコード) ：	
関数プロトタイプ	ssp_err_t (* simPinSet) (sf_cellular_ctrl_t * const p_ctrl, uint8_t * const p_old_pin, uint8_t * const p_new_pin)	

3.1.13 simLock

説明：SIMをロックします。

パラメータ		
名前	方向	内容
p_ctrl	In	Link-> st_sf_cellular_ctrl
p_pin	In	PIN number to lock the SIM
戻り値	Link->ssp_err_t (SSPエラーコード) ：	
関数プロトタイプ	ssp_err_t (* simLock) (sf_cellular_ctrl_t * const p_ctrl, uint8_t * const p_pin)	

3.1.14 simUnlock

説明：SIMをアンロックします。

パラメータ		
名前	方向	内容
p_ctrl	In	Link-> st_sf_cellular_ctrl
p_pin	In	PIN number to unlock the SIM
戻り値	Link->ssp_err_t (SSPエラーコード) ：	
関数プロトタイプ	ssp_err_t (* simUnlock) (sf_cellular_ctrl_t * const p_ctrl, uint8_t * const p_pin)	

3.1.15 simIDGet

説明：SIM IDを取得します。

パラメータ		
名前	方向	内容
p_ctrl	In	Link-> st_sf_cellular_ctrl
p_sim_id	Out	SIM ID
戻り値	Link->ssp_err_t (SSPエラーコード) ：	
関数プロトタイプ	ssp_err_t (* simIDGet)(sf_cellular_ctrl_t * const p_ctrl, uint8_t * p_sim_id)	

3.1.16 fotaCheck

説明：利用可能なファームウェアアップグレード(Firmware upgrade)を確認します。

パラメータ		
名前	方向	内容
p_ctrl	In	Link-> st_sf_cellular_ctrl
戻り値	Link->ssp_err_t (SSPエラーコード) ：	
関数プロトタイプ	ssp_err_t (* fotaCheck) (sf_cellular_ctrl_t * const p_ctrl)	

3.1.17 fotaStart

説明：ファームウェアアップグレードを開始します。

パラメータ		
名前	方向	内容
p_ctrl	In	Link-> st_sf_cellular_ctrl
戻り値	Link->ssp_err_t (SSPエラーコード) ：	
関数プロトタイプ	ssp_err_t (* fotaStart) (sf_cellular_ctrl_t * const p_ctrl)	

3.1.18 fotaStop

説明：ファームウェアアップグレードを停止します。

パラメータ		
名前	方向	内容
p_ctrl	In	Link-> st_sf_cellular_ctrl
戻り値	Link->ssp_err_t (SSPエラーコード) ：	
関数プロトタイプ	ssp_err_t (* fotaStart) (sf_cellular_ctrl_t * const p_ctrl)	

3.1.19 reset

説明：セルラー ハードウェアモジュールをリセットします。

パラメータ		
名前	方向	内容
p_ctrl	In	Link-> st_sf_cellular_ctrl
reset_type	In	Reset Type
戻り値	Link->ssp_err_t (SSPエラーコード) ：	
関数プロトタイプ	ssp_err_t (* reset) (sf_cellular_ctrl_t * const p_ctrl, sf_cellular_reset_type_t reset_type))	

3.2 セルラー フレームワークソケットインタフェースAPI(Cellular Framework Socket Interface API)

セルラーフレームワークモジュールはソケットインタフェース(socket interface)を使用して、オンチップスタック(on-chip stack)を持つセルラーハードウェアモジュール(cellular hardware module)と連携するためのAPIセットを提供します。セルラーハードウェアモジュールのオンチップスタックと通信するために、セルラーフレームワークで使用されるAPIを以下に示します。　フレームワークは、オンチップモジュールと通信するAPIの2つのセットを提供します。最初のAPIのセットは、インスタンスの作成時に作成されるAPIの一環としてp_ctrlおよびp_cfgデータ構造体を使用します。2番目のAPIセットは、データ通信のためにTCP/UDPソケットを作成するソケットインタフェース(socket interface)です。APIについて効率よく理解を深めるために、構造体とその詳細を*SSPユーザーズマニュアル(SSP User Manual)*で説明します。

このインスタンス構造体は、セルラーフレームワークインタフェースのインスタンスを使用するために必要なすべての要素を包含しています。

```
typedef struct st_sf_cellular_onchip_stack_instance
{
```

sf_cellular_onchip_stack_ctrl_t	* p_ctrl	セルラー フレームワークインスタンスの制御構造体へのポインタ
sf_cellular_onchip_stack_cfg_t	const * p_cfg	セルラー フレームワークインスタンスの設定構造体へのポインタ
sf_cellular_onchip_stack_api_t	const * p_api	セルラー フレームワークインスタンスのAPI構造体へのポインタ

```
} sf_cellular_onchip_stack_instance_t;
```

```
typedef struct st_sf_cellular_onchip_stack_ctrl
{
```

sf_cellular_instance_t	* p_lower_lvl_cellular	SFセルラーインスタンスへのポインタ

```
} sf_cellular_onchip_stack_ctrl_t
```

セルラー設定パラメータを定義します。

```
typedef struct st_sf_cellular_onchip_stack_cfg
{
```

sf_cellular_instance_t	const * p_lower_lvl	SFセルラーインスタンスへのポインタ
void	* p_extend	拡張設定

```
} sf_cellular_onchip_stack_cfg_t;
```

CAT3およびCAT1に関するオンチップソケットAPIエラーコード

SF_CELLULAR_CAT3_SOCKET_INVALID_FD	(-1)	無効なソケットディスクリプタ
SF_CELLULAR_CAT3_SOCKET_ERROR	(-1)	ソケットAPI処理のエラー
SF_CELLULAR_CAT3_SOCKET_SUCCESS	(0)	ソケット成功

SF_CELLULAR_CAT1_SOCKET_INVALID_FD	(-1)	無効なソケットディスクリプタ
SF_CELLULAR_CAT1_SOCKET_ERROR	(-1)	ソケットAPI処理のエラー
SF_CELLULAR_CAT1_SOCKET_SUCCESS	(0)	ソケット成功

3.2.1 open

説明：セルラーハードウェアモジュールを初期化し、データ転送に対して有効にします。ドライバの初期設定、ドライバリンクの有効化、割り込みの許可、およびデータ転送に向けたデバイスの準備を行います。

パラメータ		
名前	**方向**	**内容**
p_ctrl	In	Link-> st_sf_cellular_onchip_stack_ctrl
p_cfg	In	Link-> st_sf_cellular_onchip_stack_cfg
戻り値	Link->**ssp_err_t（SSPエラーコード）：**	
関数プロトタイプ	ssp_err_t (* open) (sf_cellular_socket_ctrl_t * p_ctrl, sf_cellular_socket_cfg_t const * const p_cfg)	

3.2.2 close

説明：ネットワークインタフェースを初期化せず(un-initialize)に、低消費電力モードにするか電源オフにすることが可能な関数へのポインタ。ドライバを閉じ、ドライバリンクを無効にし、割り込みを禁止します。

パラメータ		
名前	**方向**	**内容**
p_ctrl	In	Link-> st_sf_cellular_onchip_stack_ctrl
戻り値	Link->**ssp_err_t（SSPエラーコード）：**	
関数プロトタイプ	ssp_err_t (* close) (sf_cellular_socket_ctrl_t * const p_ctrl)	

3.2.3 versionGet

説明：バージョンを取得し、指定されたポインタp_versionに格納します。

パラメータ		
名前	**方向**	**内容**
p_version	Out	バージョン番号を返すメモリ位置へのp_versionポインタは、APIおよびSSPコードのバージョン番号を取得します。
戻り値	Link->**ssp_err_t（SSPエラーコード）：**	
関数プロトタイプ	ssp_err_t (* versionGet) (ssp_version_t * **const** p_version)	

3.2.4 socket

説明：本APIはソケットを作成します。

パラメータ		
名前	**方向**	**内容**
p_ctrl	In	Link-> st_sf_cellular_onchip_stack_ctrl
p_cfg	In	Link-> st_sf_cellular_onchip_stack_cfg
戻り値	Link-> On-chip socket API error codes for CAT3 and CAT1	
関数プロトタイプ	int socket (int domain, int type, int protocol)	

3.2.5 close

説明：本APIはソケットを閉じます。

パラメータ		
名前	**方向**	**内容**
socket_fd	In	ローカルソケット
戻り値	Link-> On-chip socket API error codes for CAT3 and CAT1	
関数プロトタイプ	**int close** (**int** socket_fd)	

3.2.6 bind

説明：IPアドレスで識別されるインタフェースへのAPIバインドソケット。

パラメータ		
名前	**方向**	**内容**
socket_fd	In	ローカルソケット
p_local_sock_addr	In	ローカルソケットアドレスへのポインタ
addrlen	In	ソケットアドレス構造体のサイズ
戻り値	Link-> On-chip socket API error codes for CAT3 and CAT1	
関数プロトタイプ	**int bind** (**int** socket_fd, **const struct** sockaddr * p_local_sock_addr, socklen_t addrlen)	

3.2.7 listen

説明：TCP接続をリッスン(Listen)します。TCP接続に対してソケットをリッスンモード(listen mode)に設定します。

パラメータ		
名前	**方向**	**内容**
socket_fd	In	ローカルソケット
backlog	In	接続キューの最大数
戻り値	Link-> On-chip socket API error codes for CAT3 and CAT1	
関数プロトタイプ	**int listen** (**int** sockfd, **int** backlog)	

3.2.8 connect

説明：リモートソケットでTCP接続を確立します。

パラメータ		
名前	**方向**	**内容**
socket_fd	In	ローカルソケット
p_local_sock_addr	In	ローカルソケットアドレスへのポインタ
addrlen	In	ソケットアドレス構造体のサイズ
戻り値	Link-> On-chip socket API error codes for CAT3 and CAT1	
関数プロトタイプ	**int connect** (**int** sockfd, **const struct** sockaddr * p_serv_addr, socklen_t addrlen)	

3.2.9 accept

説明：リモートからの接続要求を受け入れます。

パラメータ		
名前	**方向**	**内容**
sockfd	In	ローカルソケット
p_cliaddr	Out	接続を試行しているリモートソケットアドレスへのポインタ
p_addrlen	Out	クライアントソケットアドレスのアドレス長へのポインタ
戻り値	Link-> On-chip socket API error codes for CAT3 and CAT1	
関数プロトタイプ	**int accept** (**int** sockfd, **struct** sockaddr * p_cliaddr, socklen_t * p_addrlen)	

3.2.10 send

説明：リモートソケットにデータを送信します。

パラメータ		
名前	**方向**	**内容**
sockfd	In	ローカルソケット
p_buf	In	データバッファへのポインタ
length	In	データバッファ長
flags	In	ソケットフラグ
戻り値	成功の場合、これらの呼び出しから送信済み文字数が返されます。 エラーの場合、-1が返されます。	
関数プロトタイプ	**ssize_t send(int sockfd, const void * p_buf, size_t length, int flags)**	

3.2.11 recv

説明：リモートソケットからデータを受信します。

パラメータ		
名前	**方向**	**内容**
sockfd	In	ローカルソケット
p_buf	Out	データが受信されるデータバッファへのポインタ
length	In	受信可能なデータの最大長
flags	In	ソケットフラグ
戻り値	成功の場合、これらの呼び出しから受信済み文字数が返されます。 エラーの場合、-1が返されます。	
関数プロトタイプ	ssize_t **recv** (**int** sockfd, **void** * p_buf, size_t length, **int** flags)	

3.2.12 sendto

説明：リモートソケットにデータを送信します。

パラメータ		
名前	**方向**	**内容**
sock_fd	In	ローカルソケット
p_buf	Out	送信されるデータバッファへのポインタ
length	In	データバッファ長
flags	In	ソケットフラグ
p_dest_addr	In	データの送信先となるリモートソケットアドレスへのポインタ
addrlen	In	ソケットアドレス構造の長さ
戻り値	成功の場合、これらの呼び出しから送信済み文字数が返されます。エラーの場合、-1が返されます。	
関数プロトタイプ	ssize_t **sendto** (**int** sockfd, **const void** * p_buf, size_t length, **int** flags, **const struct** sockaddr * p_dest_addr, socklen_t addrlen)	

3.2.13 recvfrom

説明：リモートソケットからデータを受信します。

パラメータ		
名前	**方向**	**内容**
sockfd	In	ローカルソケット
p_buf	Out	データが受信されるデータバッファへのポインタ
length	In	受信可能なデータの最大長
flags	In	ソケットフラグ
p_dest_addr	In	データを送信したリモートソケットアドレスへのポインタ
addrlen	In	ソケットアドレス構造の長さ
戻り値	成功の場合、これらの呼び出しから受信済み文字数が返されます。エラーの場合、-1が返されます。	
関数プロトタイプ	ssize_t recvfrom (int sockfd, void * p_buf, size_t length, int flags, struct sockaddr * p_src_addr, socklen_t * p_addrlen)	

3.2.14 setsockopt

説明：ソケットオプションを設定します。

パラメータ		
名前	**方向**	**内容**
sockfd	In	ローカルソケット
level	In	ソケットAPIレベル
optname	In	設定されるオプション
p_optval	In	設定されるオプション値
optlen	In	オプション値の長さ
戻り値	Link-> On-chip socket API error codes for CAT3 and CAT1	
関数プロトタイプ	int setsockopt (int sockfd, int level, int optname, const void * p_optval, socklen_t optlen)	

3.2.15 getsockopt

説明：ソケットオプションを取得します。

パラメータ		
名前	**方向**	**内容**
sockfd	In	ローカルソケット
level	In	ソケットAPIレベル
optname	In	取得されるオプション
p_optval	Out	取得されるオプション値
optlen	In	オプション値の長さ
戻り値	Link-> On-chip socket API error codes for CAT3 and CAT1	
関数プロトタイプ	int getsockopt (int sockfd, int level, int optname, void * p_optval, socklen_t * p_optlen)	

3.2.16 select

説明：指定された時間にわたって、特定のソケットを待機します。パケットが動作または到着すると、待機から抜けます。

パラメータ		
名前	**方向**	**内容**
nfds	In	最大fd
p_readfds	In	データが読み出し可能かどうかを確認するfd_setへのポインタ
p_writefds	In	データが書き込み可能かどうかを確認するfd_setへのポインタ
p_exceptfds	In	例外条件が発生したかどうかを確認するfd_setへのポインタ
p_timeout	In	待機時間（ミリ秒単位）
戻り値	Link-> On-chip socket API error codes for CAT3 and CAT1	
関数プロトタイプ	int select (int nfds, fd_set * p_readfds, fd_set * p_writefds, fd_set * p_exceptfds, struct timeval * p_timeout);	

注：　関数のデータ構造体、typedef、define、APIデータ、API構造体、および関数変数に関する動作および定義の詳細については、*SSPユーザーズマニュアル*の「API References for the associated module」を確認してください。

4. アプリケーションにセルラー フレームワークモジュールを組み込む(Including the Cellular Framework Module in an Application)

本項では、ISDEコンフィグレータ(ISDE configurator)を使用してアプリケーションにセルラーフレームワークモジュールを組み込む方法について説明します。

注： ユーザがプロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内のブロックの設定に精通していることが前提です。ユーザがこれらの項目に精通していない場合は、*SSPユーザーズマニュアル*の最初の数章を参照して、SSPベースのアプリケーションを作成する際にこれらの重要な各手順を管理する方法を確認してください。

アプリケーションにセルラーフレームワークを追加するには、以下の表に示すスタック選択シーケンスを使用して、スレッドにセルラーフレームワークを追加するだけです。セルラーフレームワークは、アプリケーションにフレームワークを追加するために、以下のオプションをサポートしています。TCP/IPスタックがロードされてモジュールで動作する場所に基づいて、セルラーフレームワークを以下のように分類することが可能です。

- TCP/IPスタック（Synergyホストで動作するTCP/IPスタック）としてNetXを使用するセルラーフレームワーク
- オンチップスタック（セルラー ハードウェアモジュールに存在するTCP/IPスタック）を使用するセルラーフレームワーク

4.1 セルラー フレームワークモジュールをNetXにTCP/IPスタックとして組み込む(Including the Cellular Framework Module with NexX as TCP/IP stack)

セルラーフレームワークがNetXで使用される場合、下記の3とおりの方法で組み込むことが可能です。

- セルラーフレームワークをNetXポート（NSAL層）のみに組み込む
- セルラーフレームワークをIPインスタンス(IP instance)とともにアプリケーションに組み込む
- セルラーフレームワークをNetXアプリケーション層とともに組み込む

表1　セルラーフレームワークモジュールをNetXポートに組み込む

リソース	ISDEタブ	スタック選択シーケンス
g_sf_el_nx0（セルラーフレームワークを使用するNetXポート）	スレッド	「Framework」->「Networking」->「Cellular」->「NetX Port using Cellular Framework」

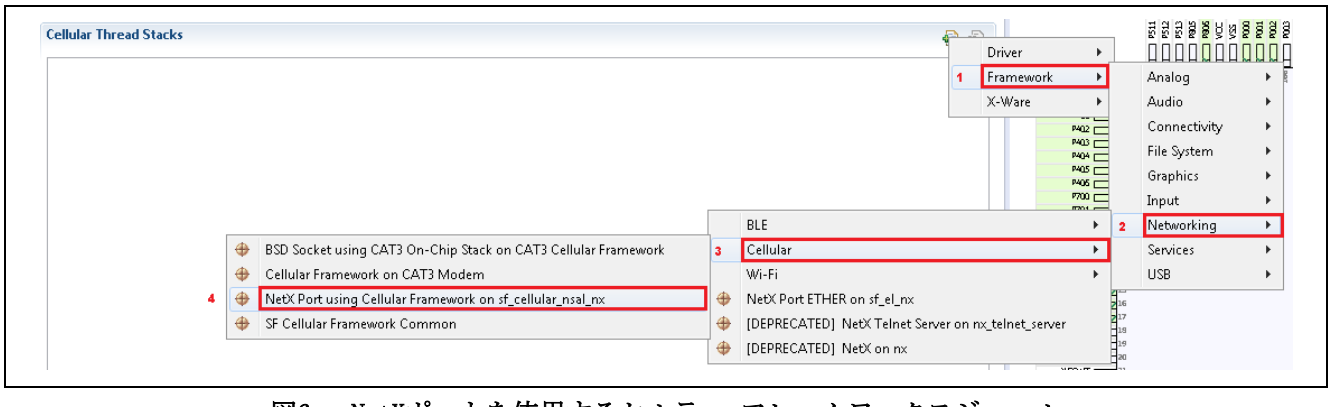

図6　NetXポートを使用するセルラー フレームワークモジュール

表2　セルラーフレームワークモジュールをNetX IPインスタンスに組み込む

リソース	ISDEタブ	スタック選択シーケンス
g_ip0 (NetX IPインスタンス)	スレッド	「X-Ware」->「NetX」->「NetX IP Instance」

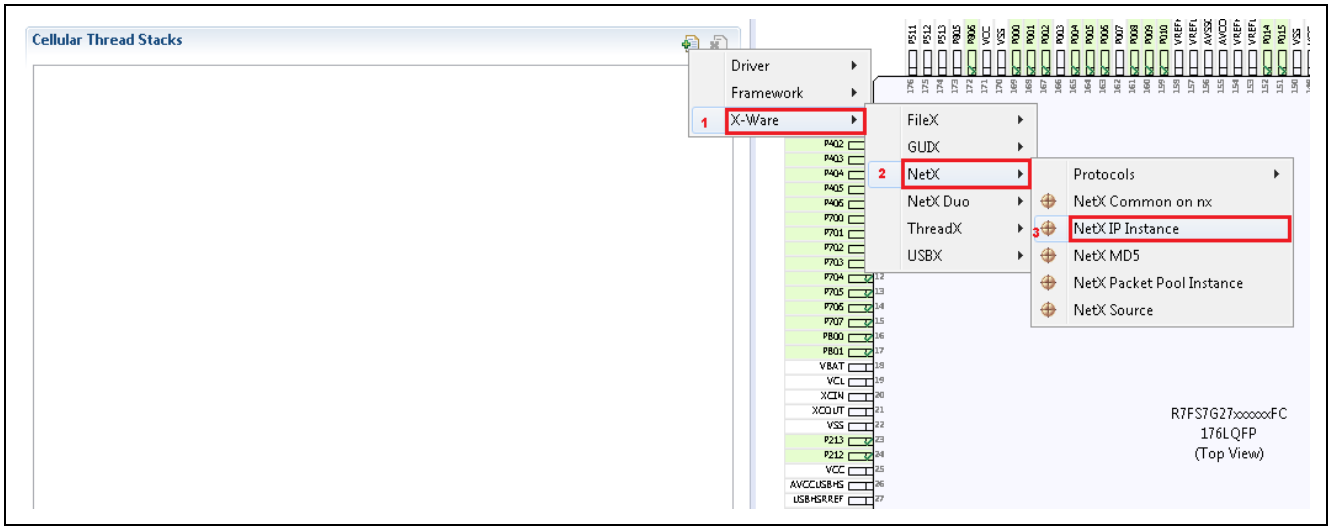

図7　セルラー フレームワークモジュールをNetX IPインスタンスに組み込む

表3　セルラーフレームワークを使用するNetXポートをIPインスタンスに組み込む

リソース	ISDEタブ	スタック選択シーケンス
g_sf_cellular_nx0（セルラーフレームワークを使用するNetXポート）	スレッド	組み込まれた（IPインスタンス）から、「Add NetX Network Driver」->「New」->「NetX Port using Cellular Framework on sf_cellular_nsal_nx」

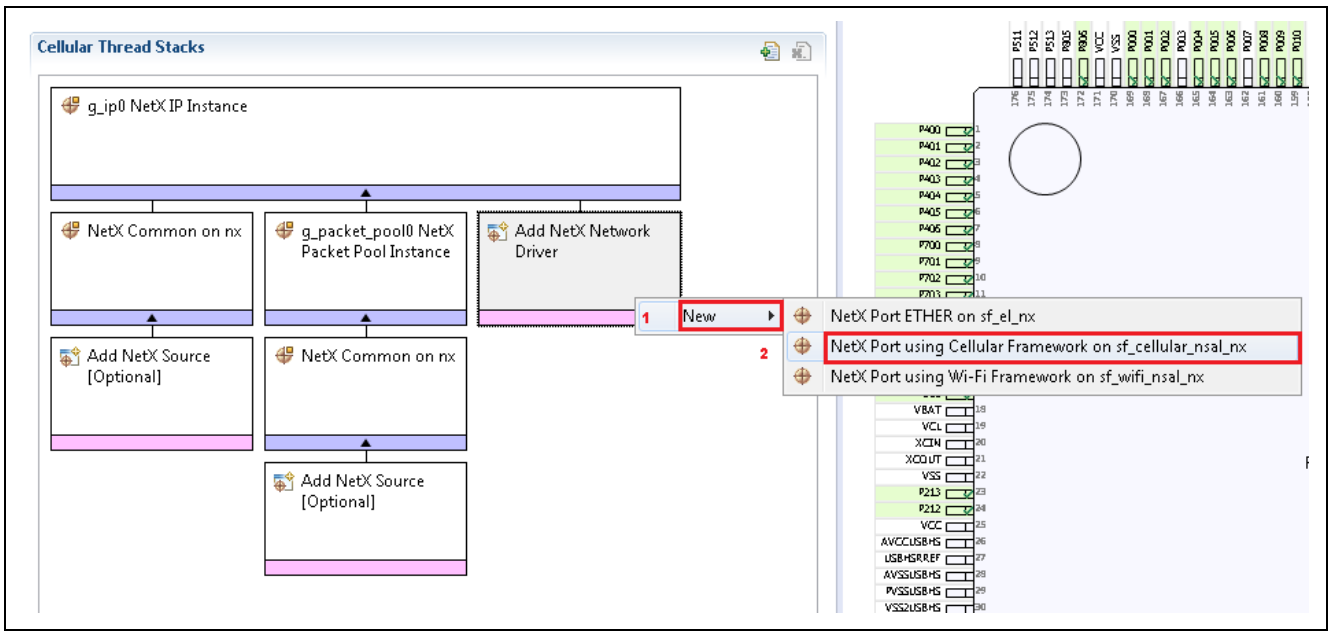

図8　セルラー フレームワークNSAL層を組み込む

一部のアプリケーションでは、NetXアプリケーション層またはIPインスタンス（Synergy Wi-FiおよびEthernetアプリケーションなど）とともにセルラーフレームワークを組み込む必要があります。HTTPクライアントシーケンスを組み込む際のシーケンスおよびサンプルスナップショットを以下に示します。

表4　NetX HTTPモジュール選択シーケンス

リソース	ISDEタブ	スタック選択シーケンス
g_http_client0（NetX httpクライアント）	スレッド	「X-Ware」->「Protocols」->「NetX HTTP Client」

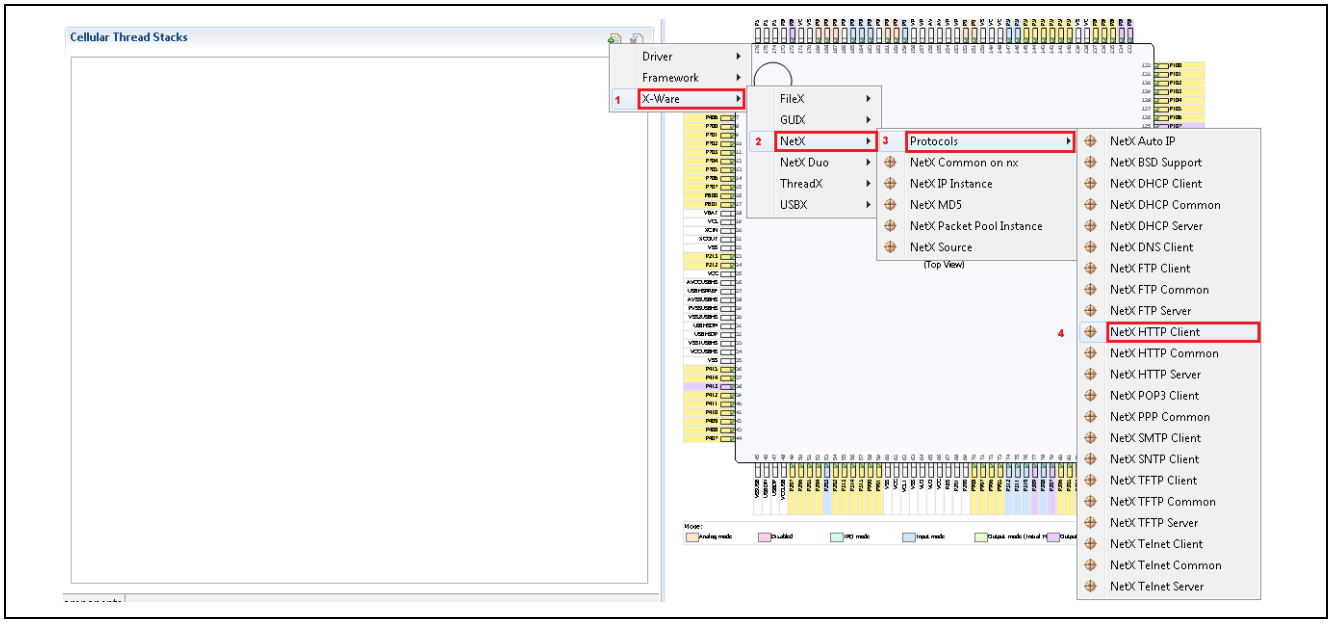

図9　NetXアプリケーションHTTPクライアントを組み込む

表5　NetXスタックによるセルラーフレームワークモジュール選択シーケンス

リソース	ISDEタブ	スタック選択シーケンス
g_sf_cellular_nx0（セルラーフレームワークを使用するNetXポート）	スレッド	組み込まれたNetXアプリケーション（HTTPクライアント）から、「Add NetX Network Driver」->「New」->「NetX Port using Cellular Framework on sf_cellular_nsal_nx」

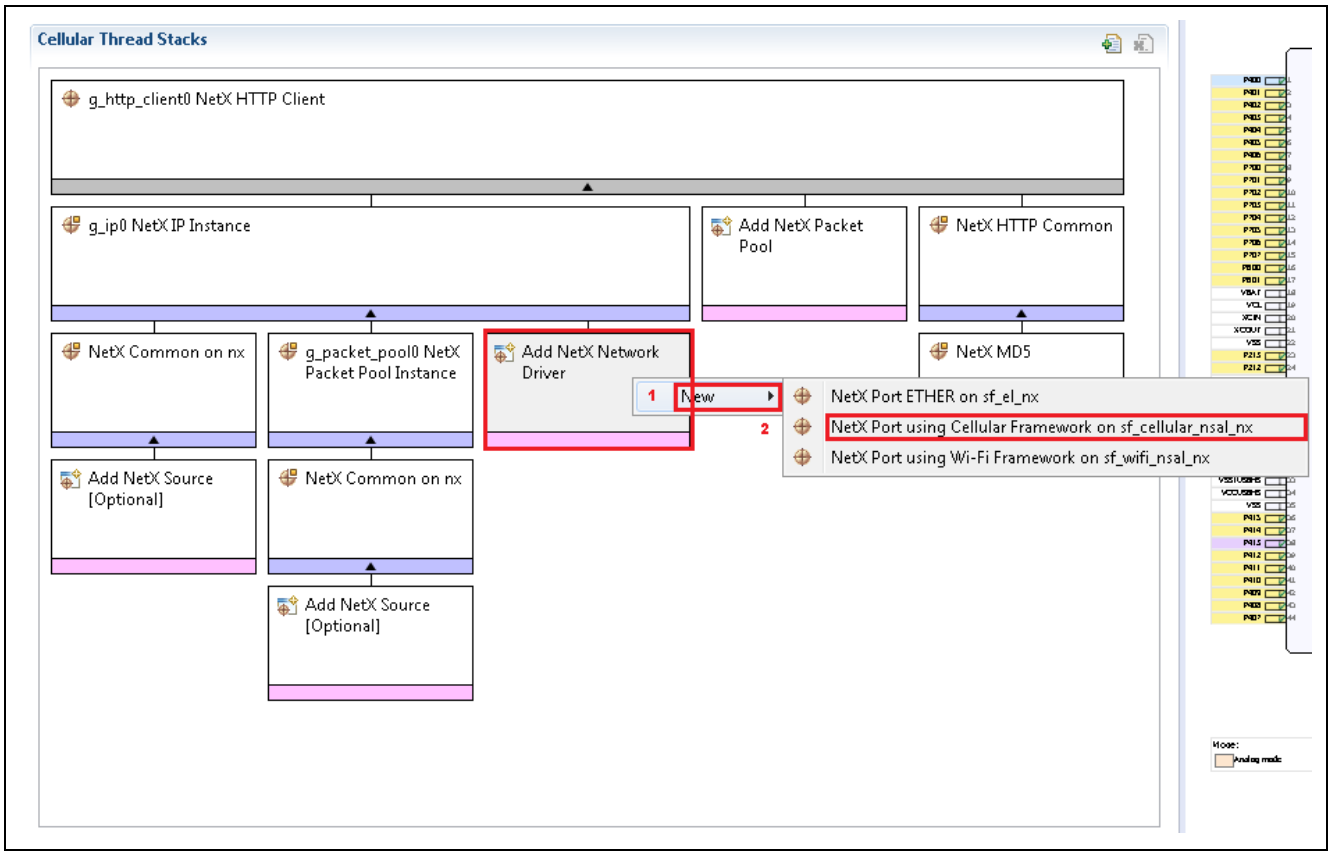

図10　NetXアプリケーション層にNSAL層を組み込む

4.2 セルラー フレームワークモジュールをTCP/IPのオンチップスタックに組み込む (Including the Cellular Framework Module with On-chip Stack for TCP/IP)

一部のアプリケーションでは、セルラーハードウェアモジュール自体に存在するオンチップTCP/IPスタックにセルラーフレームワークを組み込む必要があります。セルラーハードウェアモジュールで動作するスタックが使用される場合、NetXスタックはSynergyホストで使用されません。セルラーフレームワークをオンチップスタックサポートシーケンスとともに組み込む際のシーケンスおよびサンプルスナップショットを以下に示します。

表3　　オンチップスタックによるCAT3のセルラー フレームワークモジュール選択シーケンス

リソース	ISDEタブ	スタック選択シーケンス
g_sf_cellular_socket0 (CAT3セルラー フレームワークのオンチップスタックを使用するBSDソケット)	スレッド	「Framework」 -> 「Networking」 -> 「Cellular」 -> 「BSD Socket using On-Chip Stack on CAT3 Cellular Framework」

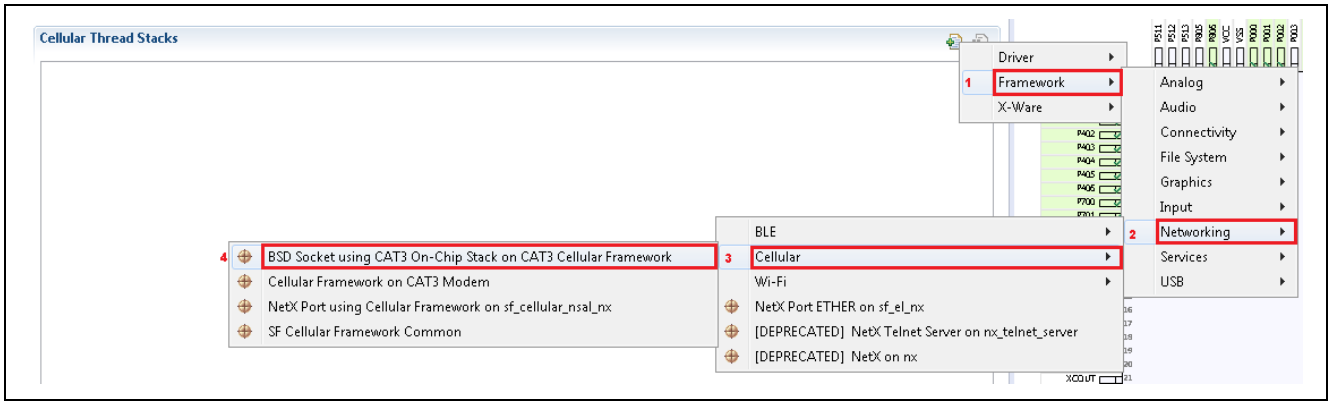

図11　CAT3にオンチップスタックを使用するセルラー フレームワークモジュール

表4　　オンチップスタックによるCAT1のセルラー フレームワークモジュール選択シーケンス

リソース	ISDEタブ	スタック選択シーケンス
g_sf_cellular_socket0 (CAT1セルラー フレームワークのオンチップスタックを使用するBSDソケット)	スレッド	「Framework」 -> 「Networking」 -> 「Cellular」 -> 「BSD Socket using On-Chip Stack on CAT1 Cellular Framework」

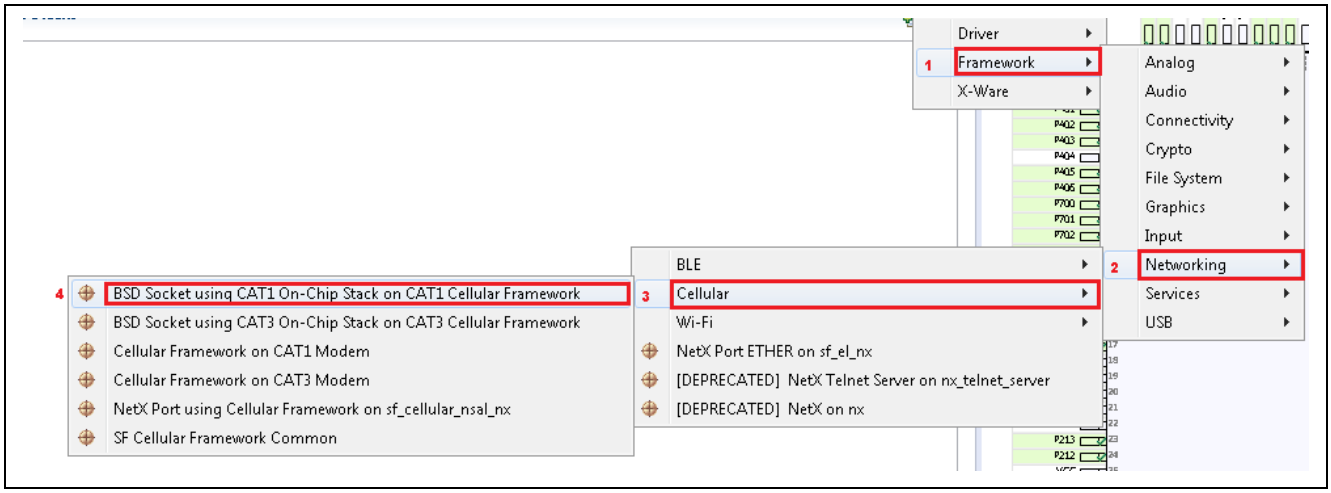

図12　CAT1にオンチップスタックを使用するセルラー フレームワークモジュール

5. セルラーフレームワークモジュールの設定(Configuring the Cellular Framework Module)

前項では、セルラーフレームワークをアプリケーションに組み込む各種方法について説明しました。本項では、フレームワークモジュールとその依存するモジュールの設定について説明します。個別の設定パラメータ、その推奨値、デフォルト値について詳細に解説しています。このため、ユーザはこれらの項目を必要に応じて自分のアプリケーションに適用することが可能です。

5.1 セルラー フレームワークをNetXでTCP/IPスタックとして設定 (Configuring Cellular Framework with NetX as TCP/IP Stack)

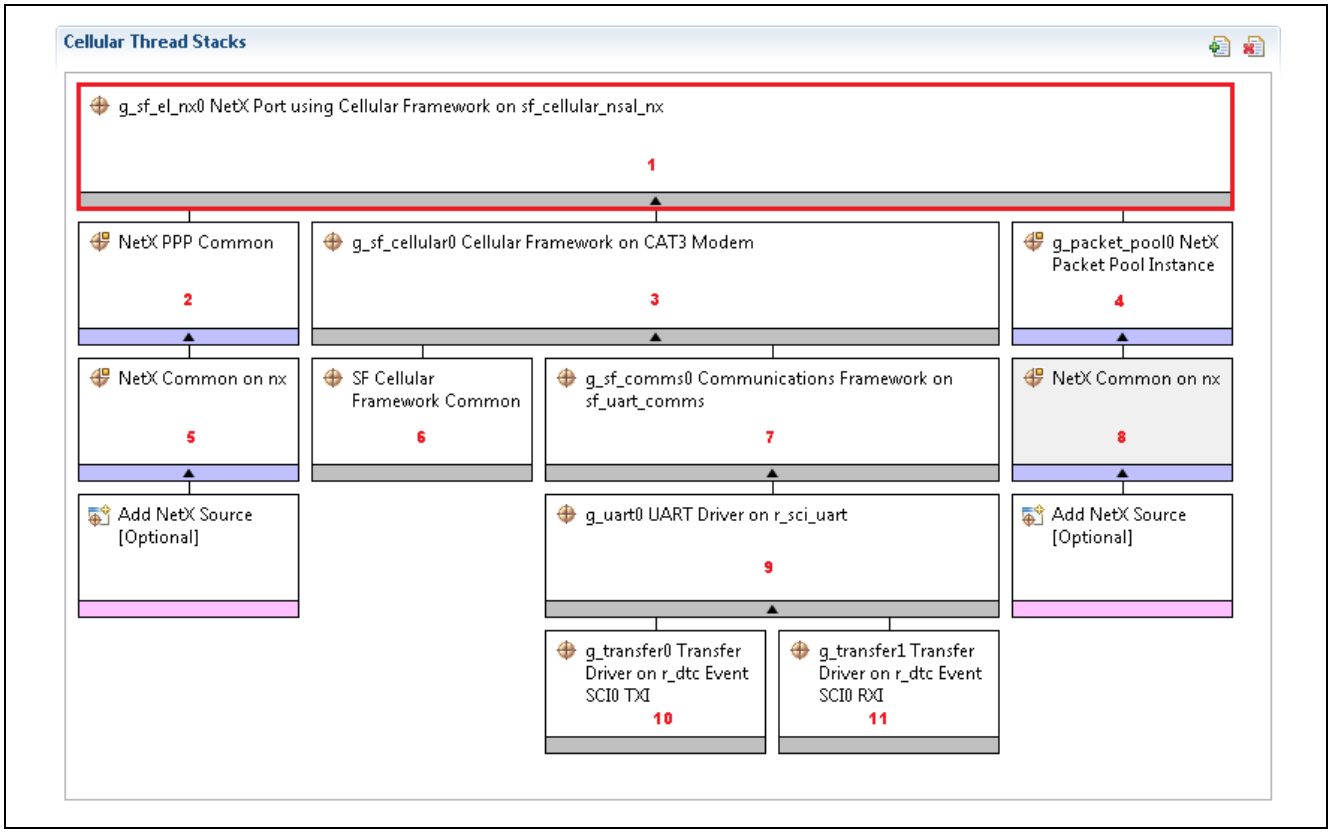

図13　NetXをTCP/IPスタックとして使用するセルラーフレームワークモジュール

上記の図13に記載されている（1）の設定プロパティ（sf_cellular_nsal_nxでセルラーフレームワークを使用するg_sf_el_nx0 NetXポート）を以下に示します。

ISDEプロパティ	デフォルト値	内容
共通		
パラメータ確認 (Parameter Checking)	BSP、有効(Enabled)、 無効(Disabled) デフォルト：BSP	これらは、SSPコード内のAPIから渡されるパラメータを確認するオプションのSSP機能です。アプリケーションでSSPコードの追加確認が不要な場合、ユーザはこれを無効にすることが可能です。
sf_cellular_nsal_nxでセルラー フレームワークを使用するg_sf_el_nx0 NetXポート		
名前(Name)	g_sf_el_nx0	NetXポートインスタンス(Port instaance)の名前
PPPスタックサイズ（バイト単位） (PPP Stack Size in Bytes)	2048	これは、PPPのスタックサイズです。デフォルトでは、2048に設定されます。ユーザは最適な動作のためにこの値を保持する必要があります。ユーザは最小スタックサイズを2048として保持する必要があります。
名前(Name)	g_nx_ppp0	PPPインスタンス(PPP instance)の名前
PPPスレッドの優先度を表す数値（IP Helperスレッドよりも低い優先度にする必要があります）。有効な値は0～（TX_MAX_PRIORITES-1）の範囲です。値0が最も高い優先度を表します。	3	これは、フレームワークで使用される内部PPPスレッドの優先度です。ユーザはアプリケーションスレッドをこの優先度と同じレベルに維持することが推奨されます。または、アプリケーションスレッドを低い優先度にすることも可能です。
認証方法(Authentication Method)	None	これは、PPPの認証を選択するためのフィールドです。PPPはPAPまたはCHAPの認証、または認証なし（None）で動作します。
無効なパケットハンドラコールバック(Invalid Packet Hadler Callback)	NULL	これは、無効なパケット(Invalid Packet)を処理するためのコールバック(Callback)です。ユーザは無効なパケットをコールバックで処理することが可能です。
PAPログインコールバック (PAP Login Callback)	ppp_link_down_callback	これは、フレームワークから提供されるコールバックです。ユーザは、各自のアプリケーションでPPPリンクダウンイベント(PPP link down event)を処理するように、このコールバックをカスタマイズすることが可能です。たとえば、PPPリンクがダウンしたときに、アプリケーションは利用可能なネットワーク通信インタフェースに切り替わることが可能です。または、適切な措置を取ってください。
PAP認証ログインコールバック (PAP verify Login Callback)	ppp_link_up_callback	これは、フレームワークから提供されるコールバックです。ユーザは、PPPリンクアップイベントを処理するように、このコールバックをカスタマイズすることが可能です。ユーザはアプリケーション要件ごとにコールバック処理をカスタマイズすることが可能で

ISDEプロパティ	デフォルト値	内容
共通		
		す。
チャレンジ値コールバック取得 (Get Challenge Values Callback)	NULL	これは、認証CHAP(Authentication CHAP)のコールバックです。ユーザが認証CHAPの使用を希望する場合、チャレンジ値を処理するようにコールバックをコーディングする必要があります。
レスポンダ値コールバック取得 (Get Response Values Callback)	NULL	これは、認証CHAPのコールバックです。ユーザが認証CHAPの使用を希望する場合、レスポンダ値を処理するようにコールバックをコーディングする必要があります。
認証コールバック取得 (Get Verification Callback)	NULL	これは、認証CHAPのコールバックです。ユーザが認証CHAPの使用を希望する場合、認証を処理するようにコールバックをコーディングする必要があります。
ローカルIPv4アドレス（区切り記号にカンマを使用します)	0,0,0,0	PPPはポイントツーポイントプロトコル(point to point protocol)です。これは、ローカルIPアドレスが設定される場所です。ただし、PPPは、このフィールドに0,0,0,0が選択された場合にピア側(peer side)がIPv4アドレスを割り当てるオプションも提供します。
ピアIPv4アドレス（区切り記号にカンマを使用します)	0,0,0,0	これは、ユーザがピア(peer)のIPアドレスを割り当てるプレースホルダ(placeholder)です。セルラーフレームワークの場合、PPPリンクに使用する所定のIPアドレスがサービス事業者から提供された場合、そのIPアドレスをここで割り当てることが可能です。

上記の図13に記載されている（2）の設定プロパティ(configuration property)（NetX PPP共通）を以下に示します。本モジュールが追加されるため、ユーザは設定に変更を加える必要はありません。

ISDEプロパティ	デフォルト値	内容
モジュールNetX PPP共通		
名前	g_nx_ppp_common0	PPP共通モジュール(PPP Common Module)の名前

5.1.1 セルラーフレームワークモジュールCAT3/CAT1モデムデバイスドライバ設定(Cellular Framework Module CAT3/CAT1 Modem Device configuration)

上記の図13に記載されている（3）の設定プロパティ（CAT3モデムのg_sf_cellular0セルラーフレームワーク）を以下に示します。本モジュールでは、モデムとフレームワークへのインタフェースに固有の設定です。

ISDEプロパティ	デフォルト値	内容
共通		
パラメータ確認 (Parameter Checking)	BSP、有効(Enabled)、無効(Disabled) デフォルト：BSP	これらは、SSPコードのAPIから渡されるパラメータを確認するオプションのSSP機能です。アプリケーションでSSPコードの追加確認が不要な場合、ユーザはこれを無効にすることが可能です。
オンチップスタックサポート (On-Chip Stack Support)	有効(Enabled)、無効(Disabled) デフォルト：無効	これらのオプションはオンチップスタックの選択に使用されます。NetXスタックが使用される場合、これを無効にする必要があります。
モデム(Modem)	CAT3の場合 - TVSG, TEUG CAT1の場合 - GELS3, WM14	これらは、アプリケーションで使用されるCAT3モジュールとCAT1モジュールに基づくモデム選択です。LTEバンドに基づく各地域（北米、ヨーロッパなど）に応じて、各種のCAT1モデムまたはCAT3モデムが使用されます。
CAT3/CAT1モデムのg_sf_cellular0セルラー フレームワーク		
名前	g_sf_cellular0	セルラーモデムのインスタンス名
SIM PIN（SIMのアンロックに使用されます）	1111	SIMをアンロックするためのSIM PIN。SIMのアンロックが必要な場合、ここで設定することが可能です。
SIM PUK（SIMのアンロックに使用されます）	12345678	SIMをアンロックするSIMパーソナルアンロックキー。SIMのアンロックが必要な場合、ここで設定することが可能です。
優先オペレータ数	0	優先オペレータ総数。ゼロ以外の場合、モデムは1～5の順にオペレータを試行します。
優先オペレータ1の名前	40422	オペレータ1の数値名
優先オペレータ1の名前形式	数値	オペレータの名前形式
優先オペレータ2の名前	40422	オペレータ2の数値名
優先オペレータ2の名前形式	数値	オペレータの名前形式
優先オペレータ3の名前	40424	オペレータ3の数値名
優先オペレータ3の名前形式	数値	オペレータの名前形式
優先オペレータ4の名前	40422	オペレータ4の数値名
優先オペレータ4の名前形式	数値	オペレータの名前形式
優先オペレータ5の名前	40424	オペレータ5の数値名
優先オペレータ5の名前形式	数値	オペレータの名前形式
オペレータ選択モード	自動	自動または手動オペレータ選択モード
オペレータ名（手動モード選択）	40422	手動モードのオペレータ数値名
オペレータ名前形式（手動モード選択）	数値	手動モードのオペレータ名前形式
タイムゾーン更新ポリシー(Time Zo	有効	ネットワークによる時刻同期

ISDEプロパティ	デフォルト値	内容
共通		
ne Update Policy)ne Update Poli		
受信データコールバック (Receive Data Callback)	sf_cellular_nsal_recv_callback	受信者のコールバック関数
プロビジョニングコールバック (Provisioning Callback)	celr_prov_callback	セルラーハードウェアモジュールをプロビジョニングするためのコールバック関数
循環キューサイズ（バイト単位) (Circular Queue Size in Bytes)	256	受信データのデータキューサイズ（バイト単位）。これは、セルラーモデムからデータを受信するためのデータバッファサイズです。256バイトがデフォルトで、必要最低限のサイズです。
SF通信フレームワークスレッドスタックサイズ(SF Communication Framework Stack Size)	512	内部通信フレームワークスレッドのスタックサイズ。セルラー フレームワークはシリアル通信に通信フレームワークを使用します。必要最低限のスタックサイズは512バイトです。
SF通信フレームワークスレッドの優先度を表す数値。有効な値は0～（TX_MAX_PRIORITES-1）の範囲です。値0が最も高い優先度を表します。	5	通信フレームワークスレッドの優先度
セルラーハードウェアモジュールリセット入出力端子(Cellular Hardware Module Reset IO Pin)	IOPORT_PORT_10_PIN_05	リセット端子として使用されるGPIO端子

5.1.2 セルラー フレームワークモジュールパケットプール設定(Cellular Framework Module Packet Pool configuration)

ISDEプロパティ	デフォルト値	内容
モジュールg_packet_pool0 NetXパケットプールインスタンス		
名前	g_packet_pool0	パケットプールのインスタンス名
パケットサイズ（バイト単位) (Packet Size in Bytes)	128	パケットのサイズ（バイト単位）。これは、PPPパケットのサイズです。
プール内のパケット数 (Number of Packet in Pool)	64	プール内のパケット総数
生成された初期化関数の名前 (Name of Packets in Pool) packet_pool_init0	packet_pool_init0	パケットプール初期化関数の名前
自動初期化(Auto Initialization)	有効	パケット初期化の自動初期化

ISDEプロパティ	デフォルト値	内容
nxのモジュールNetX共通		
生成された初期化関数の名前 (Name of generated initialization function)	nx_common_init0	NetX共通の初期化関数
自動初期化(Auto Initialization)	有効(Enabled)	NetX共通の自動初期化

ISDEプロパティ	デフォルト値	内容
共通		
パラメータ確認 (Parameter Checking)	BSP、有効(Enabled)、無効(Desabled) デフォルト：BSP	これらは、SSPコードのAPIから渡されるパラメータを確認するオプションのSSP機能です。アプリケーションでSSPコードの追加確認が不要な場合、ユーザはこれを無効にすることが可能です。
リード入力キューサイズ (4バイトワード) (Read Input Queue Size (4-Byte Words))	15	通信フレームワークがシングルリードで処理できるバイト数
sf_uart_commsのモジュールg_sf_comms0通信フレームワーク		
名前(Name)	g_sf_comms0	通信フレームワークのインスタンス名
生成された初期化関数の名前 (Name of generated initialization function)	sf_comms_init0	コンフィグレータで生成された通信初期化関数の名前
自動初期化(Auto Initialization)	有効(Enabled)	通信初期化関数の自動初期化

5.1.3 セルラーフレームワークモジュールUART設定(Cellular Framework Module UART configuration)

本項では、セルラーハードウェアモジュールと通信するセルラーフレームワークのUART設定について詳しく説明します。

注： これらのモジュール（r_sci_uart, r_dtc）の設定に関する詳細については、本書の参考情報の項に記載されているモジュールガイド(Module Guide)を参照してください。

ISDEプロパティ	デフォルト値	内容
共通		
外部RTS動作 (External RTS Operation)	無効(Disable)	RTS動作の有効化／無効化
受信(Reception)	有効(Enable)	UART受信の有効化／無効化
送信(Transmission)	有効(Enable)	UART送信の有効化／無効化
パラメータ確認 (Parameter Checking)	BSP、有効(Enabled)、無効(Disabled) デフォルト：BSP	これらは、SSPコードのAPIから渡されるパラメータを確認するオプションのSSP機能です。アプリケーションでSSPコードの追加確認が不要な場合、ユーザはこれを無効にすることが可能です。
r_sci_uartのモジュールg_uart0 UARTドライバ		
名前(Name)	g_uart0	UARTドライブインスタンスの名前
チャネル(Channel)	0	セルラー モデムに接続されたSCI UARTのチャネル番号
ボーレート(Baud Rate)	115200	セルラー モデムとのUART通信のボーレート
データビット(Data Bits)	8ビット	UART選択のデータビット数
パリティ(Parity)	なし	パリティビット選択　奇数／偶数またはなし
ストップビット(Stop Bits)	1ビット	ストップビット数
CTS/RTS選択(CTS/RTS SElection)	RTS (CTSは無効です)	RTS/CTS選択
ユーザによって定義されるUARTコールバック関数の名前	NULL	ユーザによって定義されるUARTコールバック関数
クロックソース(Clock Source)	内部クロック(Internal Clock)	クロックソース選択
SCK端子からのボーレートクロック出力(Baud rate Clock Output from SCK pin)	無効(Disable)	SCK端子からのボーレートクロック選択
スタートビット検出(Start bit detection)立ち下がりエッジ(Falling Edge)	立ち下がりエッジ(Falling Edge)	スタートビット検出エッジ (Start bit detection Edge)
ノイズ除去(Noise Cancel)	無効(Disable)	ジッタノイズ除去有効／無効 (Jitter Noise Cancel Enable/Disable)
ビットレートモジュレーション有効 (Bit Rate Modulation Enable)	有効(Enable)	ビットレートモジュレーション有効／無効(Bit Rate Modulation Enable/Disable)
受信割り込み優先度(Receive Interrupt Priority)	優先度2	データ受信割り込み優先度 (Data Receive Interrupt Priority)
送信割り込み優先度(Transmit Interrupt Priority)	優先度2	データ送信割り込み優先度 (Data Transmit Interrupt Priority)
送信終了割り込み優先度 (Transmit End Interrupt Priority)	優先度2	送信終了割り込み優先度 (Transmit End Interrupt Priority)
エラー割り込み優先度(Error Interrupt Priority)	優先度2	エラー割り込み優先度 (Error Interrupt Priority)

ISDEプロパティ	デフォルト値	内容
共通		
パラメータ確認(Parameter Checking)	デフォルト (BSP)	これらは、SSPコードのAPIから渡されるパラメータを確認するオプションのSSP機能です。アプリケーションでSSPコードの追加確認が不要な場合、ユーザはこれを無効にすることが可能です。
ソフトウェア起動(Software Start)	無効(Disabled)	転送関数起動（ソフトウェア起動）有効／無効 (Transfer function start(Software start)) Enable/Disable
DTCベクタテーブルを保持するリンカセクション(Linker section to keep DTC vector table)	.ssp_dtc_vector_table	DTCのベクタテーブル
r_dtcイベントSCI0 TXIのモジュールg_transfer0転送ドライバ		
名前(Name)	g_transfer0	転送関数のインスタンス名
モード(Mode)	通常(Normal)	DTCデータ転送のモード
転送サイズ(Transfer Size)	1バイト	転送サイズ
転送先アドレスモード(Destination Address Mode)	固定(Fixed)	転送先のアドレスモード
転送元アドレスモード(Source Address Mode)	インクリメント(Incremented)	転送元アドレスモード選択
リピート領域(Repeat Area) (通常モードでは使用しません)	転送元(Source)	
割り込み頻度(Interrupt Frequency)	すべての転送後に完了	
転送先ポインタ(Destination Pointer)	NULL	転送先ポインタ
転送元ポインタ(Source Pointer)	NULL	転送元ポインタ
転送数(Number of Transfers)	0	転送数
ブロック数(Number of Blocks) (ブロックモードでのみ有効です)	0	ブロック数
起動要因(Activation Source) (IRQを有効にする必要があります)	イベントSCI0 TXI	起動要因
自動有効化(Auto Enable)	False	自動有効化
コールバック(Callback) (ソフトウェア起動でのみ有効です)	NULL	コールバック関数
ELCソフトウェアイベント割り込み優先度(ELC Software Event Interrupt Priority)	無効(Disabled)	ELCイベント割り込み優先度

5.2 BSDソケットによるセルラー フレームワークの設定(Configuring Cellular Framework with BSD Socket)

5.2.1 セルラー フレームワークモジュールオンチップスタック設定(Cellular framework module on-chip stack configuration)

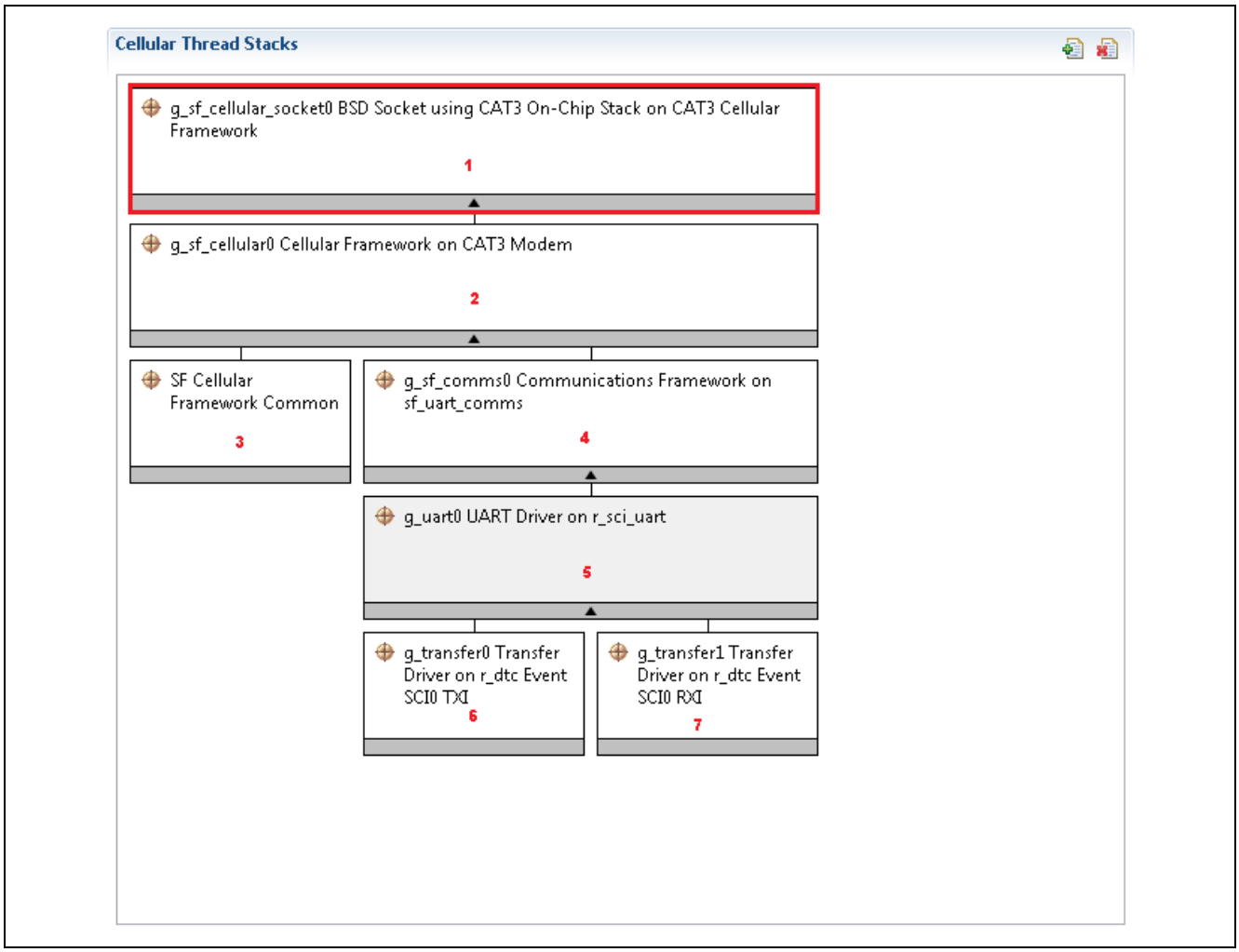

図14　オンチップスタックを使用するセルラー フレームワークモジュールの設定

5.2.2 セルラーフレームワークモジュールCAT3/CAT1モデムデバイス設定(Cellular Framework Module CAT3/CAT1 modem device configuration)

オンチップスタック(On-chip Stack)を使用するセルラーモデム設定(Cellular Modem configuration)は、NetXスタックを使用するセルラーモデム設定に関する詳細な設定と同じです。　詳細については、設定の項5.1.1を参照してください。

5.2.3 セルラーフレームワークモジュールUART設定(Cellular Framework Module UART configuration)

オンチップスタックを使用するセルラーフレームワークのUART設定は、NetXを使用するセルラーフレームワークのUART設定と同じです。　詳細については、設定の5.1.3項を参照してください。

5.2.4 セルラー フレームワークモジュール依存層設定(Cellular Framework Module dependency layer configuration)

SFセルラーフレームワーク共通 (3) 、g_sf_comms0 (4) 、g_uart0 (5) 、g_transfer0 (6) 、g_transfer1 (7) の設定も、NetXを使用するセルラーフレームワークのリストに記載されている設定に似ています。設定とその説明の詳細については、5.1項を参照してください。

6. アプリケーションのセルラーフレームワークモジュールの使用(Using the Cellular Framework Module in an Application)

前項までの内容から、モジュールで利用可能な機能に応じて2種類の方法でセルラーハードウェアモジュールを設定できることが分かりました。

- ネットワーク通信にNetXスタック(NetX stack)をTCP/IPスタックとして使用する方法
- ネットワーク通信にオンチップスタック(on-chip stack)をTCP/IPスタックとして使用する方法

セルラーモデムがNetXアプリケーションプロトコルとともに使用される場合、コンフィグレータはセルラーフレームワークを使用するネットワークポート(Network Port)の選択オプションを提供します。これらに関するコンフィグレータのスナップショットとその詳細は、4項で説明しています。

7. セルラーフレームワークモジュールアプリケーションプロジェクト(The Cellular Framework Module Application Project)

本項では、本アプリケーションガイドに対応するアプリケーションプロジェクトについて説明します。また、アーキテクチャの詳細、そのコンポーネント、設定(configuration)の詳細、コード編成(code organization)、およびユーザアプリケーションコードについても説明します。

以下の図では、アプリケーションのアーキテクチャ概要とそのデータフローを示しています。　右側には、サービスプロバイダーのネットワークへの接続を、セルラー基地局経由のPPP接続(PPP conectivity)とともに示しています。アプリケーションデモの一環として、pingパケット(ping packet)がサービスプロバイダーのネットワークでインターネットに移動してサーバからping応答で戻る様子を示しています。本アプリケーションでは、ユーザがサーバIPアドレスにpingを実行すると、Synergy側で応答が受信されます。正常なping応答（エコー）は、ネットワークIP通信に対してセルラー接続がプロビジョニング(provisioned)されたことを示します。

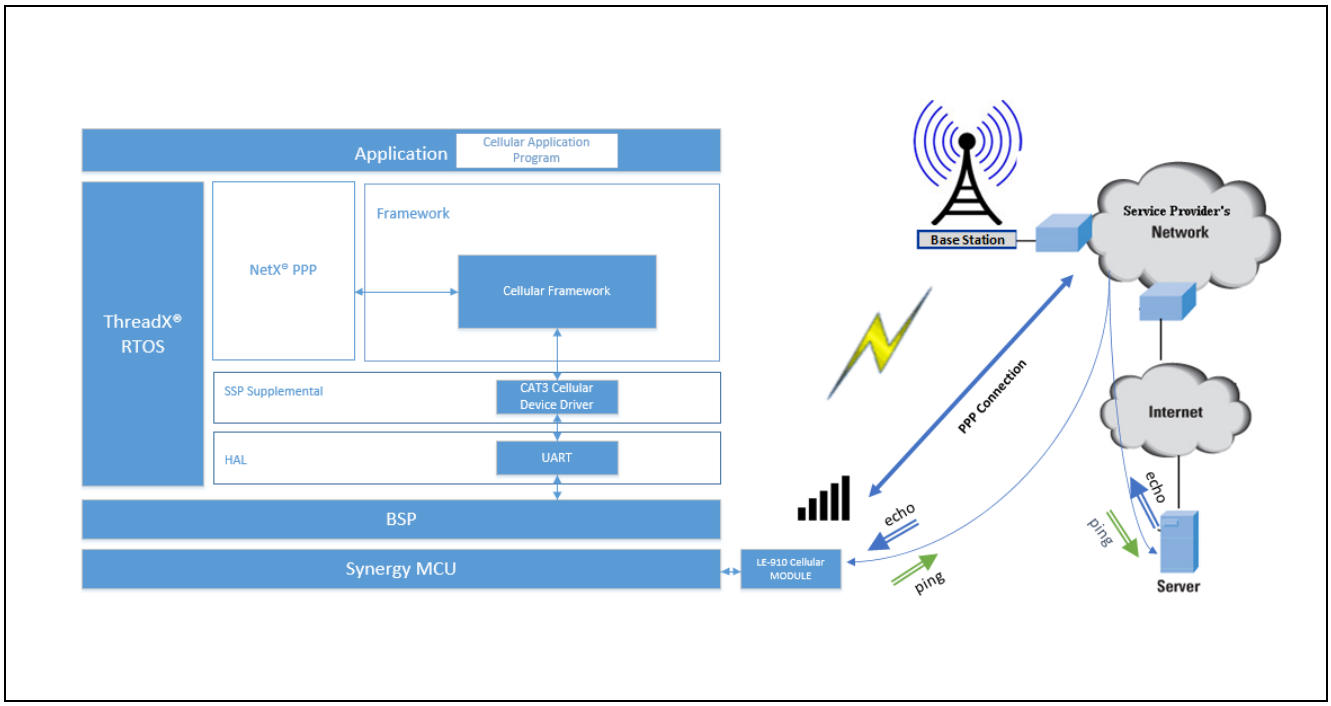

図15　セルラーフレームワークアプリケーションの概要

7.1　セルラー アプリケーションソフトウェアアーキテクチャの概要(The cellular application software architecture overview)

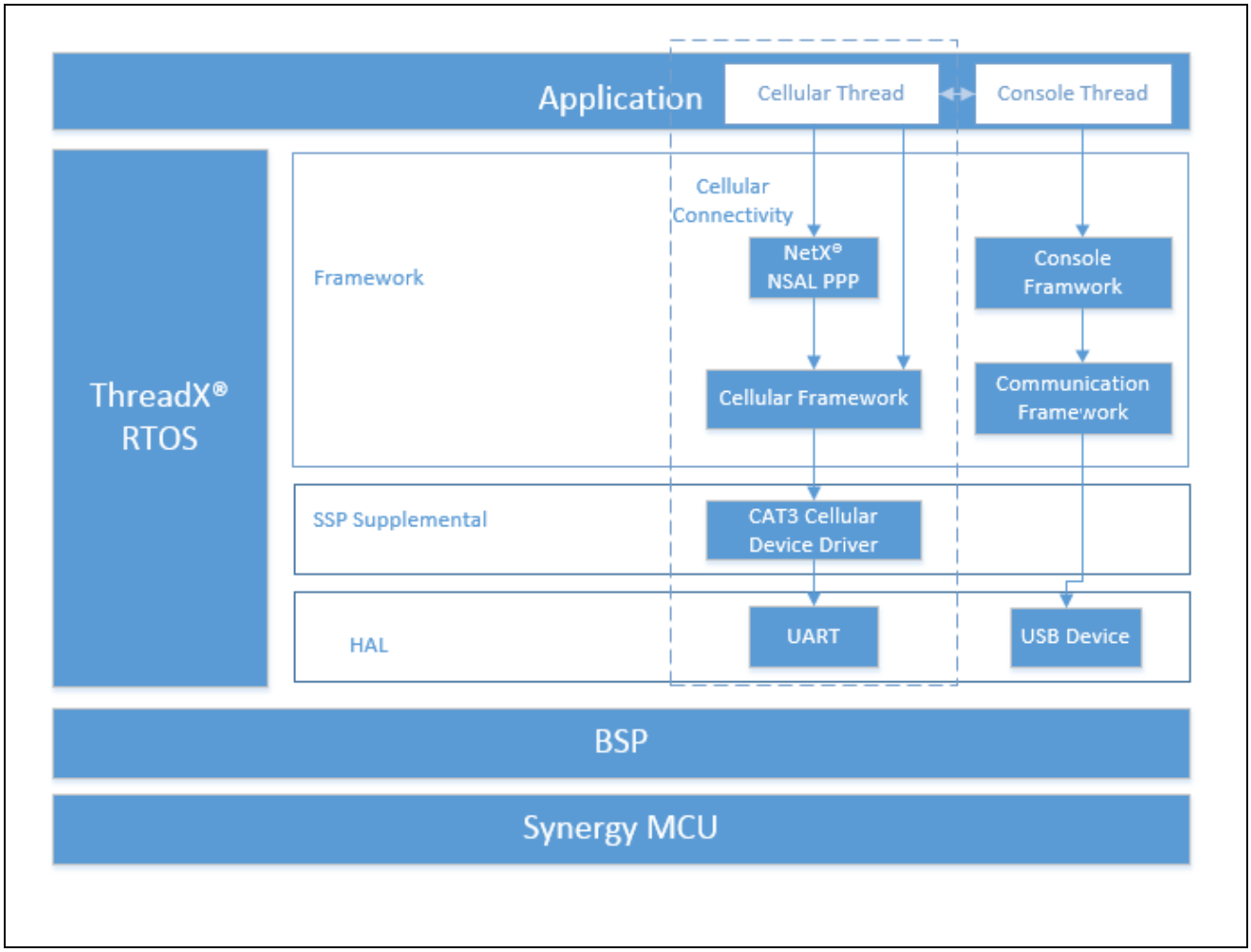

図16　セルラーアプリケーションソフトウェアアーキテクチャ

アプリケーションは以下の2つのユーザ定義スレッド(user defined thread)で構成されます。

1. コンソールスレッド(Console thread)
2. セルラースレッド(Cellular thread)。このスレッドとその機能の詳細は、個々のスレッドの項で説明します。

7.1.1　コンソールスレッド(Console thread)

コンソールスレッドはアプリケーションのユーザインタフェース部分を処理します。このユーザインタフェース部分では、アプリケーションを実行するために、Tera termまたは同等のコンソールアプリケーションを介してアプリケーションと連携することが可能です。コンソールスレッドは、コマンドラインインタフェース（CLI）を提供するコンソールフレームワーク(console framework)を使用します。コンソールフレームワークインフラストラクチャにより、アプリケーションのCLIメニューとコマンドが追加されます。ここでは、セルラー接続で簡単なPingアプリケーションを実証するようにCLIがカスタマイズされています。コンソールスレッドは、コマンドのユーザ入力データ(user entered data)を受信すると、コールバックを呼び出します。各コマンドのコールバックはコマンドデータを処理します。本アプリケーションでは、ここでユーザが「ping 8.8.8.8」（pingはコマンドです）と入力した場合、8.8.8.8はpingコマンドの引数(argument)になります。引数が解析されて処理されると、コンソールスレッドはメッセージキュー(message queue)を介してセルラースレッドにデータを送信します。　ユーザインタフェースおよびコマンドラインインタフェースのスナップショットを以下に示します。PCとの通信はUSB CDCで実行されます。　コンソールフレームワークと連動するコンソールスレッドのスナップショットを以下に示します。

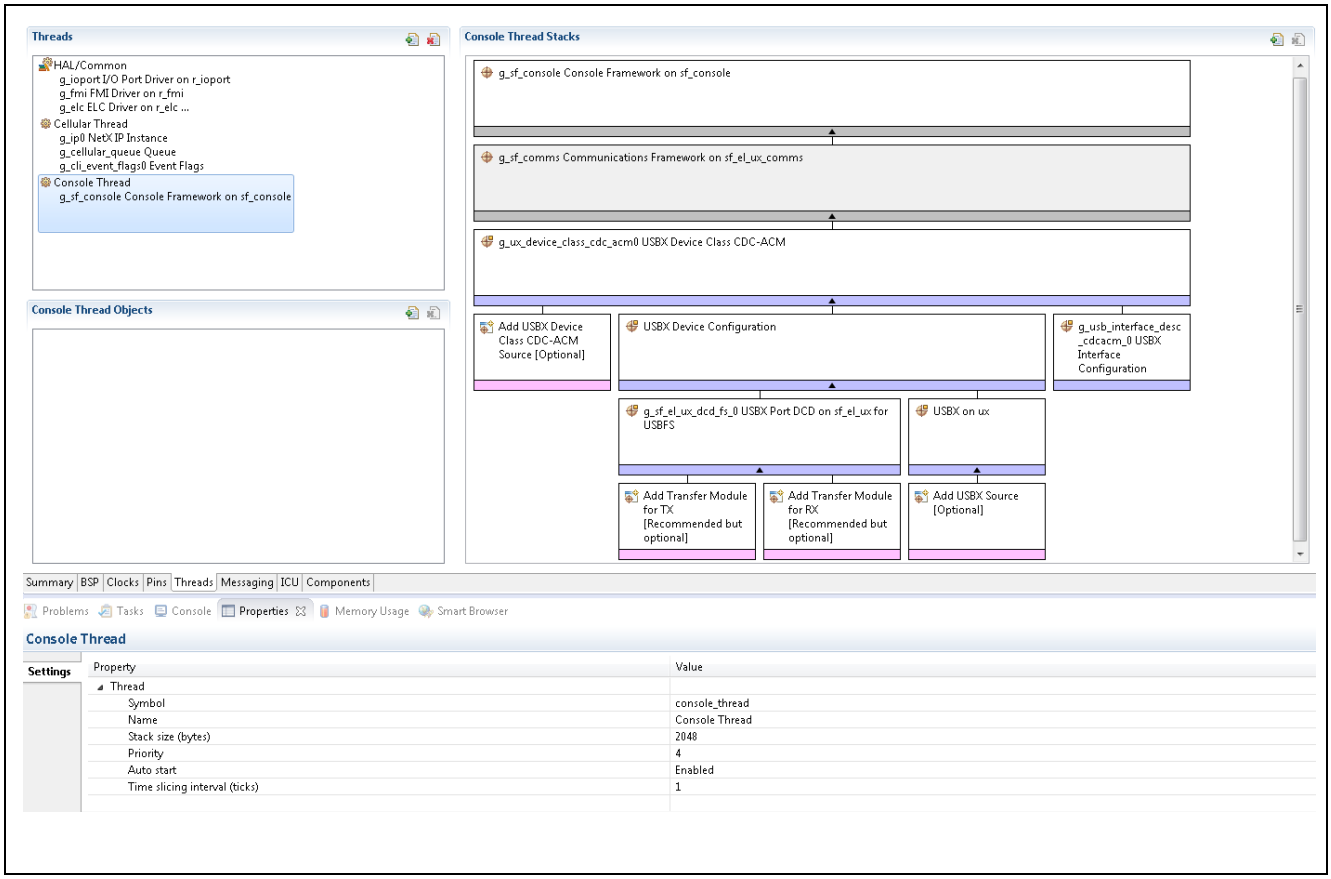

図17　コンソールスレッドおよびコンソールフレームワークコンポーネントのスナップショット

注：　アプリケーションへのコンソールフレームワークの追加およびパラメータの設定に関する詳細については、本書の参考情報の項に記載されているモジュールガイドドキュメントの*Console Framework Module Guide*を参照してください。

コンソールスレッドに対してコンフィグレータが生成したコードおよびコンソールフレームワーク固有のコードはsynergy_gen/console_thread.c/hにあります。ユーザ追加コードはsrc/console_thread_entry.cにあります。コマンドラインインタフェースのコマンドとそのコールバックはconsole_thread_entry.cにあるコードです。

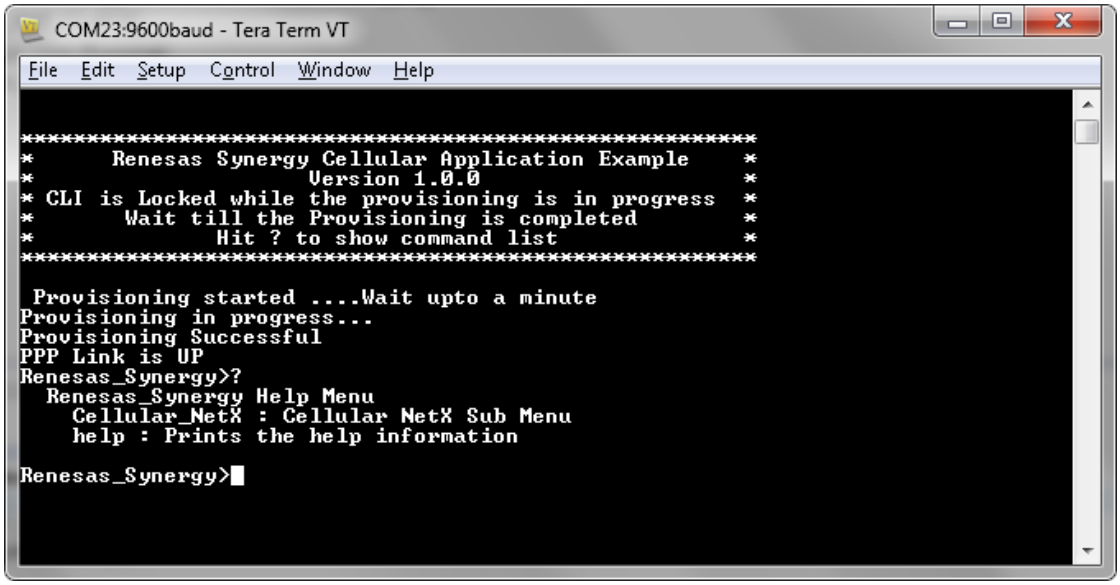

図18　コンソールベースのCLIを使用するユーザインタフェース

7.1.2　セルラーアプリケーションスレッド(Cellular Application Thread)

コンフィグレータによって生成されたコード（cellular_thread.c/h、common_data.c/h、およびフレームワークコード）と、それと連動するセルラーアプリケーションスレッド（cellular_thread_entry.c）は、セルラーアプリケーションを担当します。セルラーフレームワークと連動するIPインスタンスは、コンフィグレータによって追加されると、PPPスタックをフレームワークの一部として組み込みます。また、NSALおよびセルラーデバイスドライバコード(Cellular device driver code)も組み込みます。本スレッドから自動生成されたコードがセルラーの初期化を担当します。（cellular_thread_entry.c）にユーザが追加したコードは、主にデータ接続とICMP pingを担当するため、Ping要求をユーザが入力したパブリックIPアドレスに送信し、Ping応答を確認します。　コンソールスレッドとのスレッド間通信はメッセージキュー(message queue)を介して行われます。メッセージが受信されると、セルラーアプリケーションスレッドはメッセージを処理し、ユーザが望む機能を実行する関数を呼び出します。

セルラースレッドの一環として、メッセージキュー（g_cellular_queue）およびCLIイベントフラグ（g_cli_event_flags）が作成されます。スレッドスタックにある個々のモジュールの設定については、5項で詳しく説明しています。ユーザは設定(configuration)を参照してアプリケーションを再作成することが可能です。一部のコンフィグレータモジュールはオプションであり、作成されたアプリケーションでもオプションのままになっています。

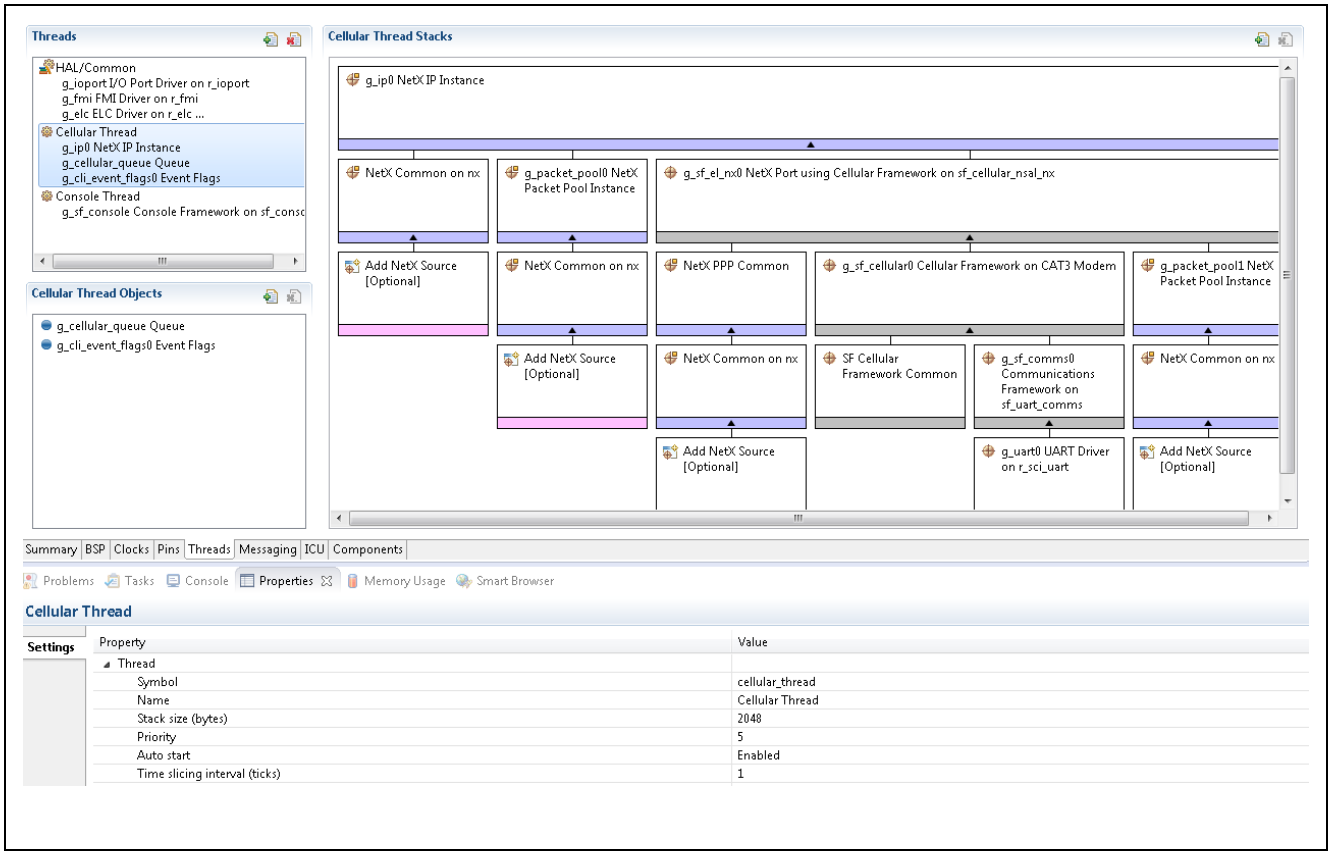

図19　セルラースレッドとその設定

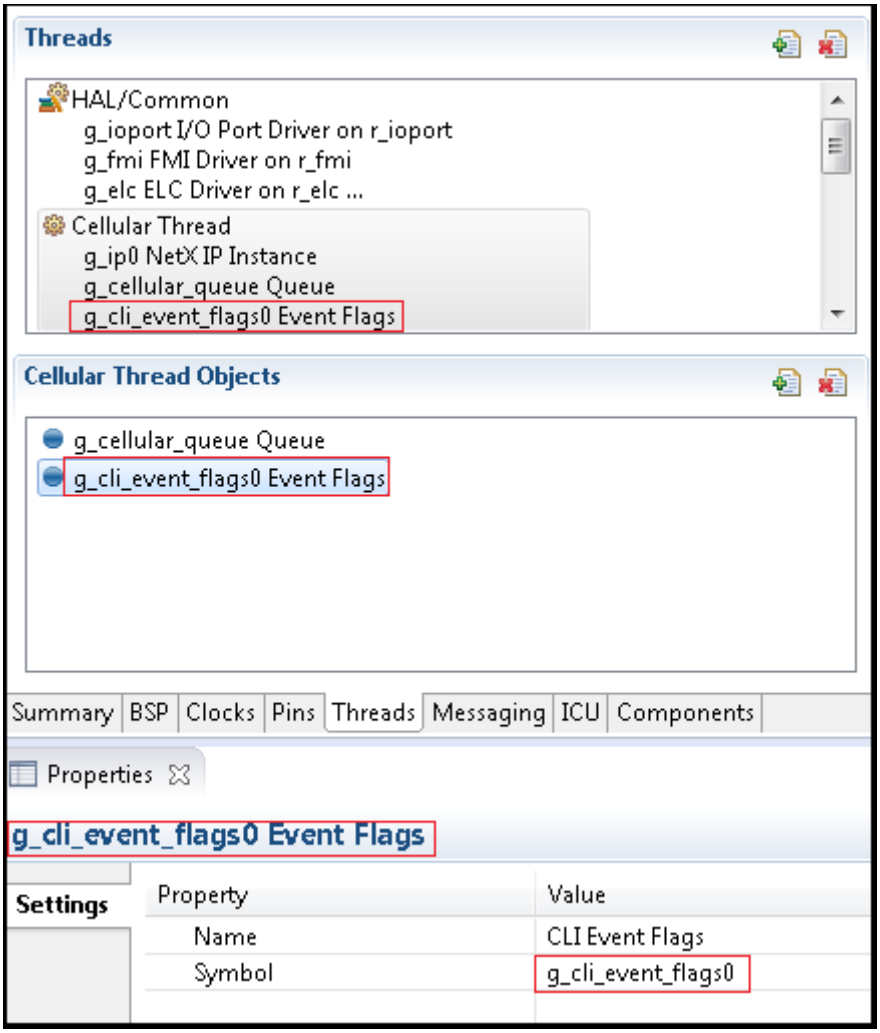

図20　セルラースレッドイベントフラグとその設定

コンソールスレッド(console thread)とセルラースレッド(cellular thread)との間で通信を行うメッセージキューに関する設定の詳細を下図21に示します。メッセージサイズは16バイトに選択されています。

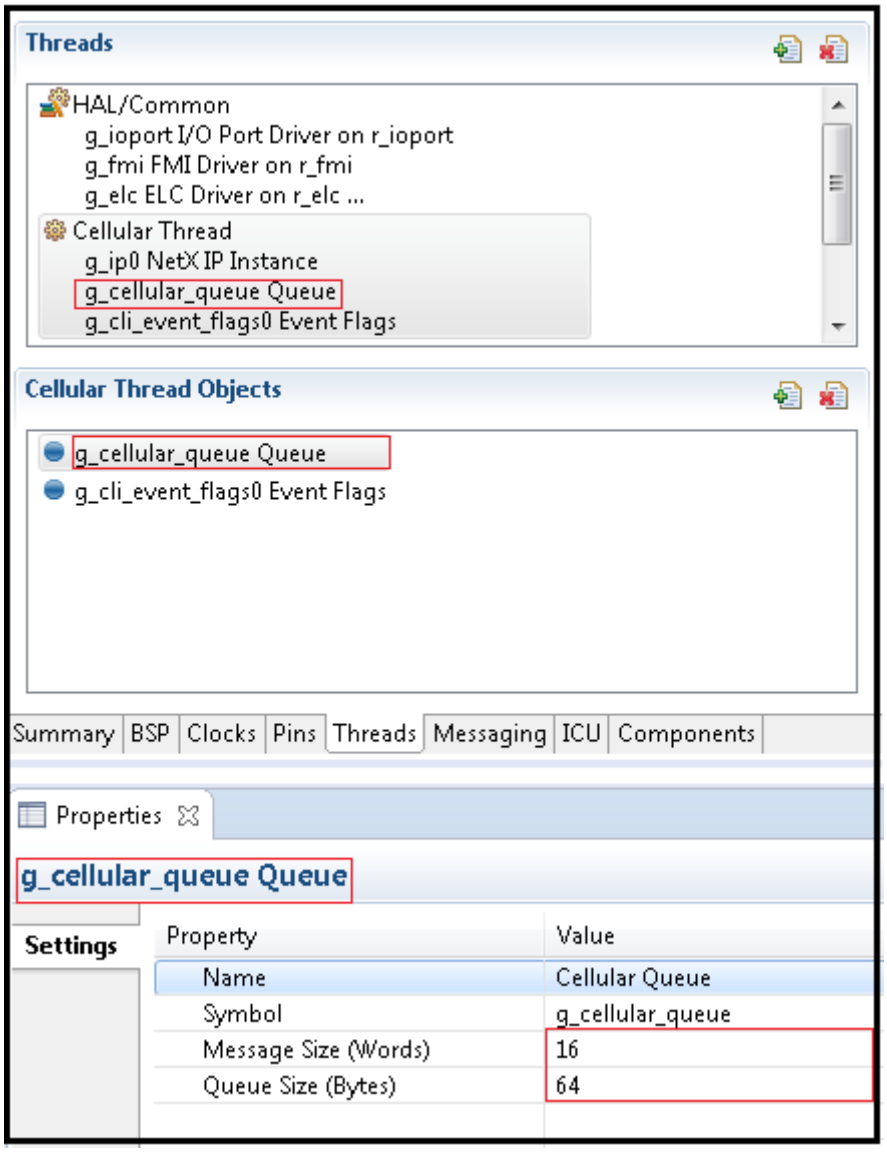

図21　セルラースレッドメッセージキューとその設定

IPインスタンスは、TCP/IPスタック(TCP/IP STACK)をプロジェクトに配置するように設定されます。TCP/IP一式をプロジェクトに配置しても、ICMPのみが有効になります。その他のすべてのプロトコルは、付属のサンプルアプリケーションに必要ない場合は、有効になります。また、IPアドレス192.168.0.2はデフォルトの設定ですが、このIPアドレスは使用されません。PPP接続確立プロセス(PPP establishment process)は、通信のためにピアのIPアドレスを取得します。あらゆるデータ通信は、ピア(peer)から発行されるIPアドレスで発生します。

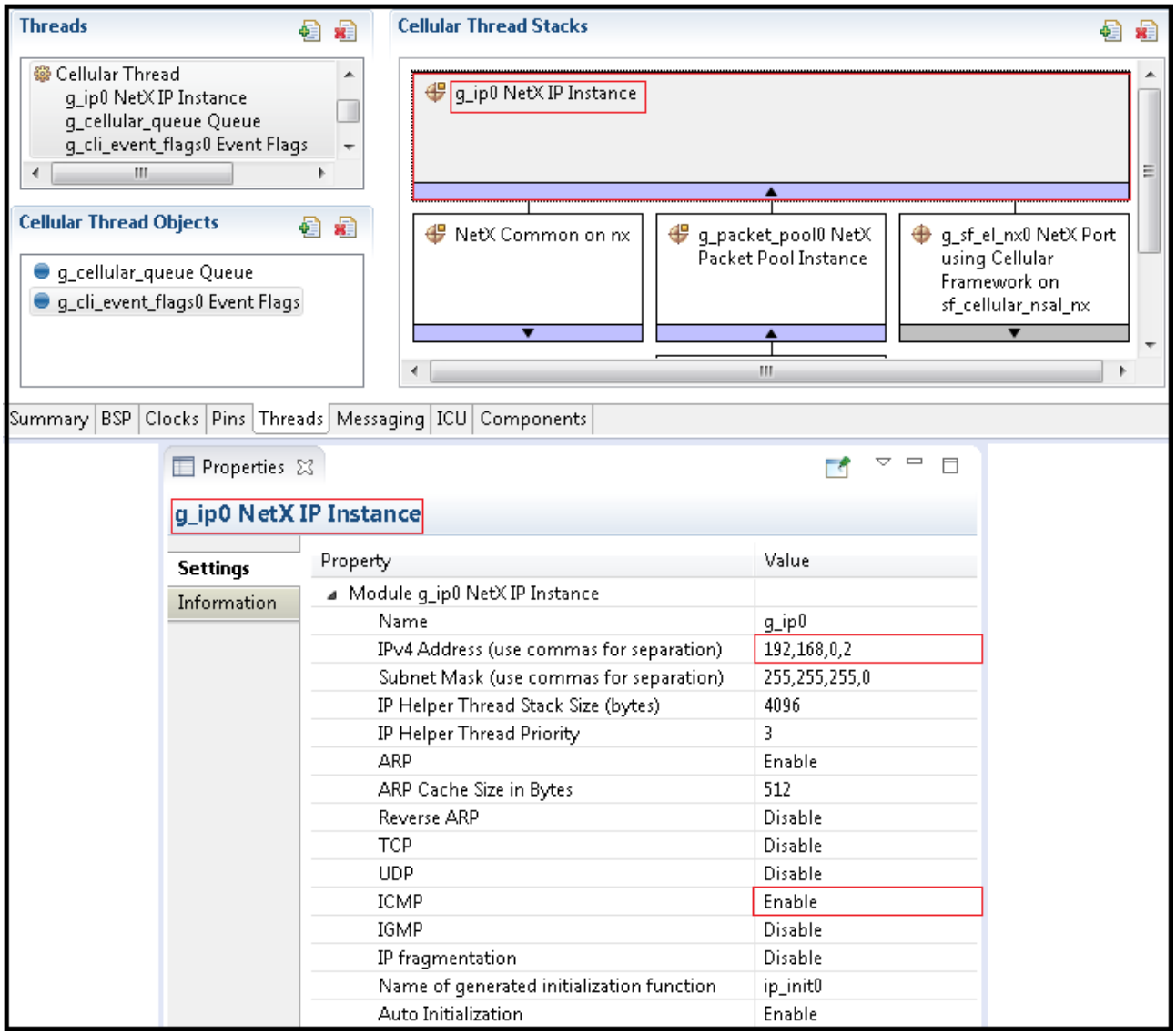

図22　　セルラースレッドのIPインスタンスとその設定

設定されたIPインスタンスは、いくつかの依存層（NetX共通など）をプロジェクトに配置します。ユーザはNetX共通(NetX common)に対して何も設定する必要はありません。

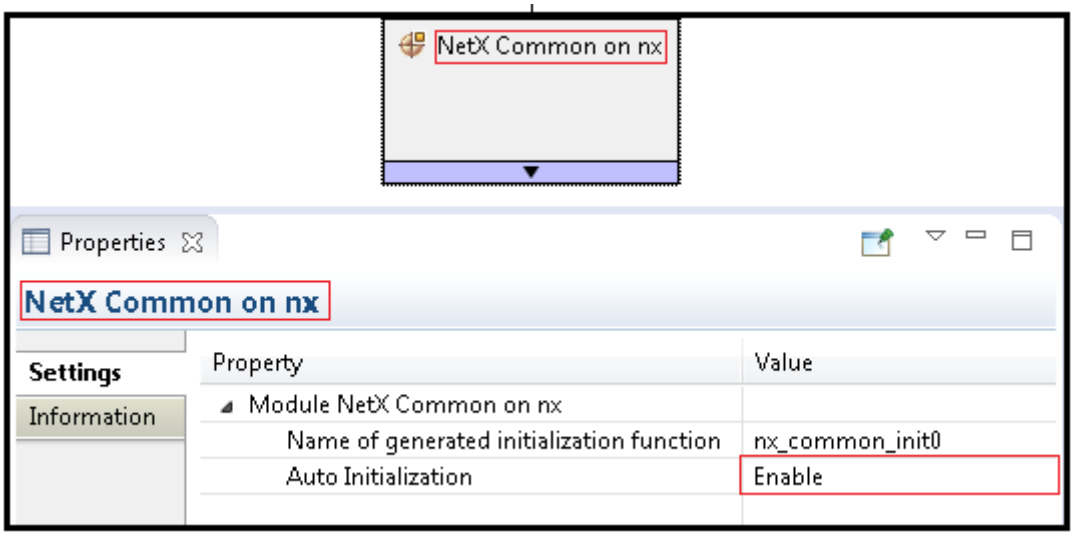

図23　セルラーのNetX共通とその設定

IPインスタンスは依存パケットプールインスタンス(dependency Packet Pool Instance)も配置します。パケットサイズとパケット数はここでアプリケーションに対して設定されます。

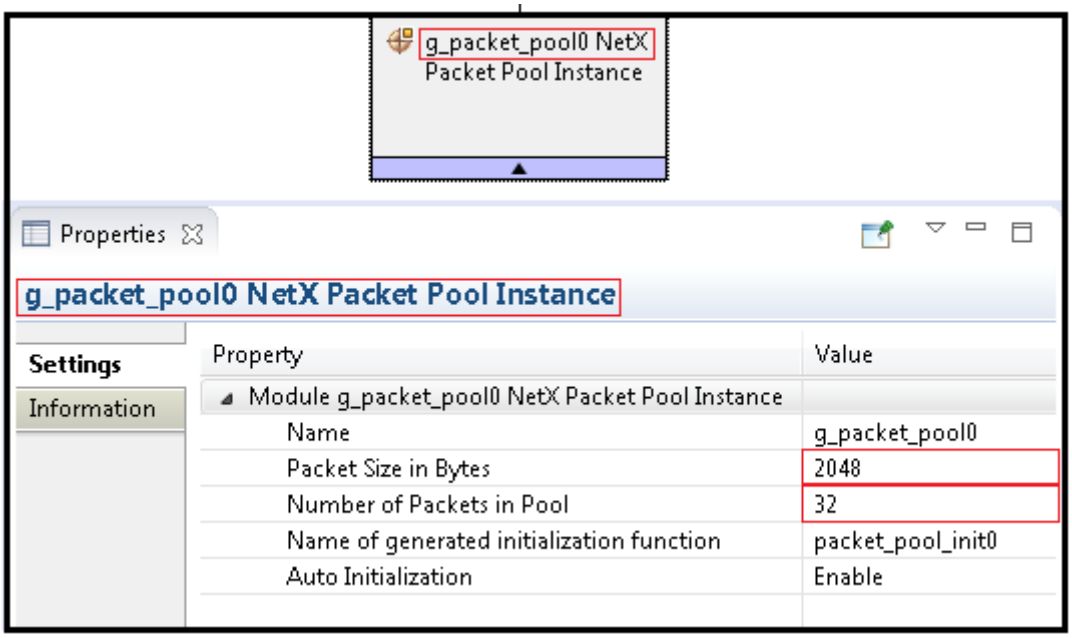

図24　セルラーのNetXパケットプールとその設定

セルラー フレームワークsf_cellular_nsal_nx設定(configuration)をアプリケーションに使用するNetXポート(NetX port)は、以下の図に示すとおりです。PPPのスタックは2048バイトとして設定されています。ユーザは、PPPリンクアップ(Link UP)およびPPPリンクダウン(Link Down)のコールバック関数を設定するオプションを利用できます。これらは、アプリケーションレベルのリンクアップイベント(Link UP event)またはリンクダウンイベント(Link Down Event)をアプリケーションで処理する際に役立ちます。ローカルおよびピアのIPアドレスは0,0,0,0のままになっています。つまり、LCPとIPCPのネゴシエーションが正常に完了すると、ピア(peer)はローカル側のIPアドレスを割り当てます。

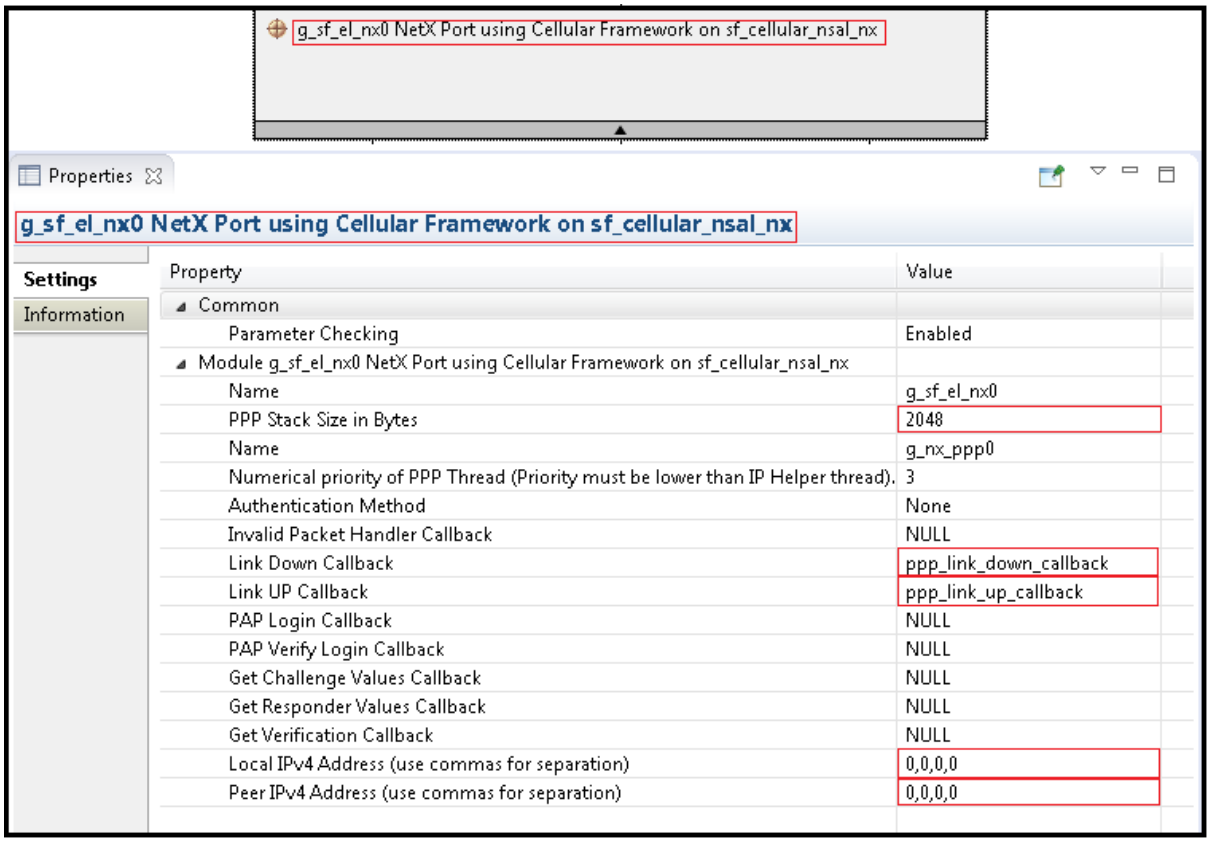

図25　　セルラーのNetXポートとその設定

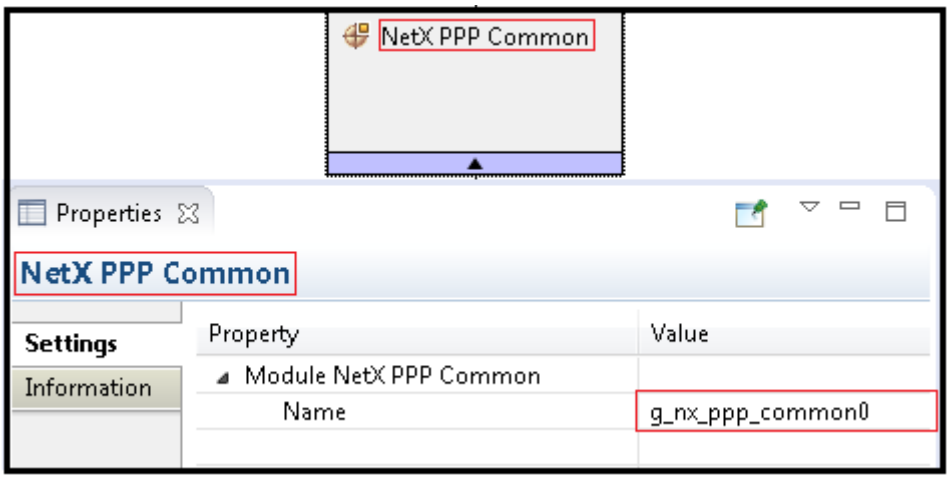

図26　　セルラーのNetX PPP共通とその設定

セルラーモデム固有の設定は、以下の図27および図28に示すように設定されます。本アプリケーションで使用されるモデムは、TSVG（ベライゾンネットワークの場合、CAT3北米市場）、GELS3（CAT1北米ベライゾンネットワーク）です。他のモデムが使用される場合には、それに応じて設定を行う必要があります。さらに、SIM PIN、優先オペレータ名、およびその形式もここで設定が可能です。これらは、優先オペレータ数が2以上の場合に効果があります。

この設定は、アプリケーションがプロビジョニングおよびデータ処理イベントを処理するための受信コールバック関数(Receive Callback function)とプロビジョニングコールバック関数(Provisioning Callback function)も提供します。本アプリケーションの循環バッファ(Circular Buffer)は512バイトとして選択されています。セルラーモデムリセット(Cellular Modem Reset)のリセット端子（GPIO）も提供します。

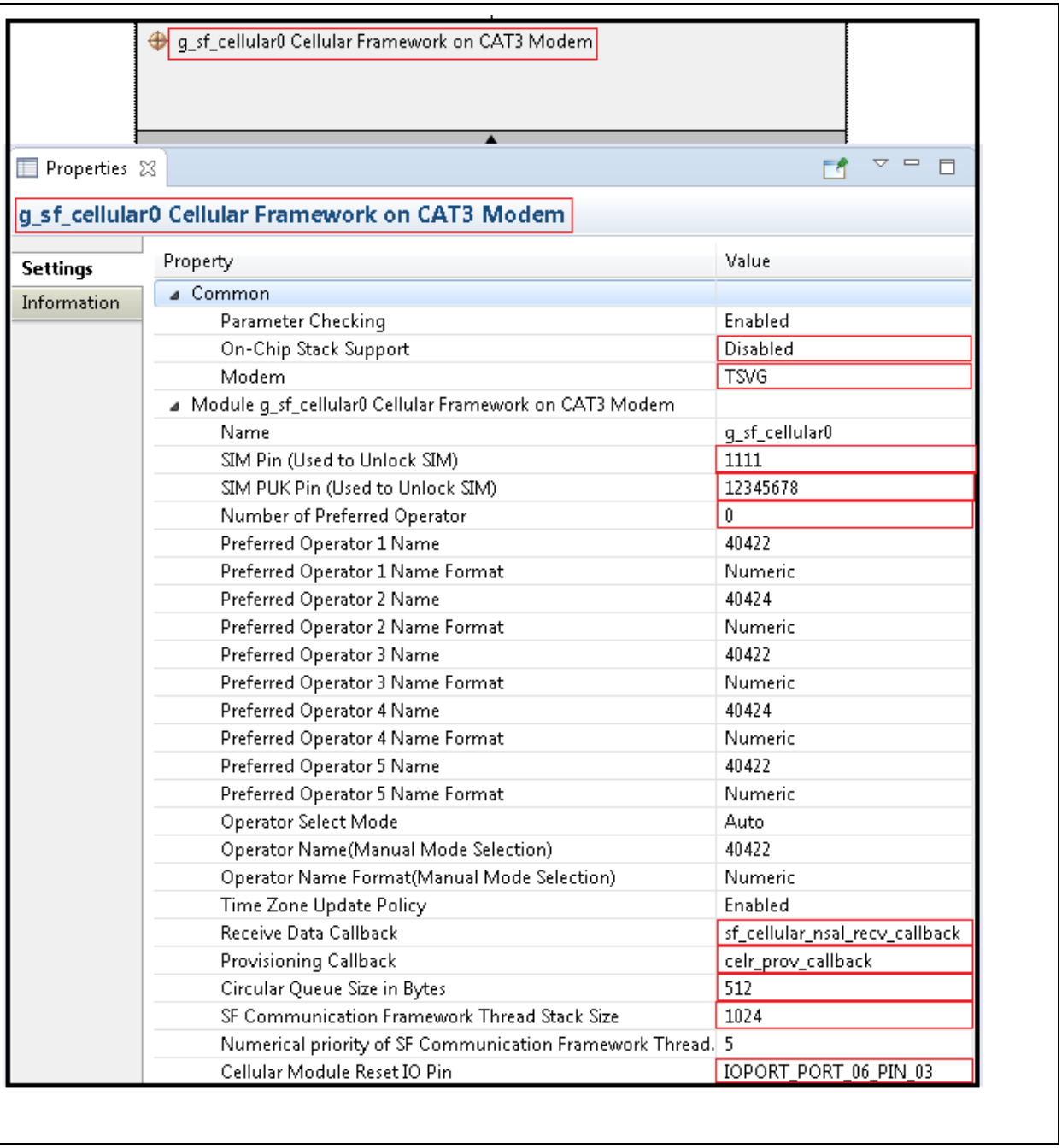

図27　CAT3モデム設定

Property	Value
◢ Common	
Parameter Checking	Default (BSP)
On-Chip Stack Support	Disabled
Modem	GELS3
◢ Module g_sf_cellular0 Cellular Framework on CAT1 Modem	
Name	g_sf_cellular0
SIM Pin (Used to Unlock SIM)	1111
SIM PUK Pin (Used to Unlock SIM)	12345678
Number of Preferred Operator	0
Preferred Operator 1 Name	40422
Preferred Operator 1 Name Format	Numeric
Preferred Operator 2 Name	40424
Preferred Operator 2 Name Format	Numeric
Preferred Operator 3 Name	40422
Preferred Operator 3 Name Format	Numeric
Preferred Operator 4 Name	40424
Preferred Operator 4 Name Format	Numeric
Preferred Operator 5 Name	40422
Preferred Operator 5 Name Format	Numeric
Operator Select Mode	Auto
Operator Name(Manual Mode Selection)	40422
Operator Name Format(Manual Mode Selection)	Numeric
Time Zone Update Policy	Enabled
Receive Data Callback	sf_cellular_nsal_recv_callback
Provisioning Callback	celr_prov_callback
Circular Queue Size in Bytes	512
SF Communication Framework Thread Stack Size	1024
Numerical priority of SF Communication Framework Thread	5
Cellular Module Reset IO Pin	IOPORT_PORT_06_PIN_03

図28　CAT1モデム設定

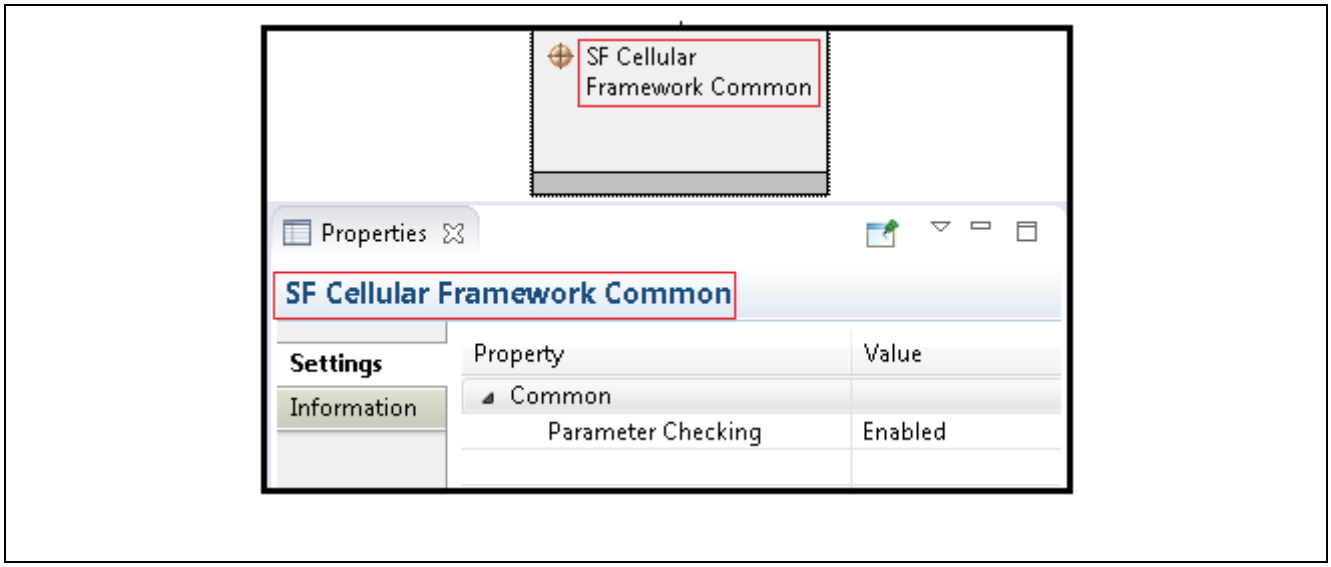

図29　セルラー スレッドのNetX共通とその設定

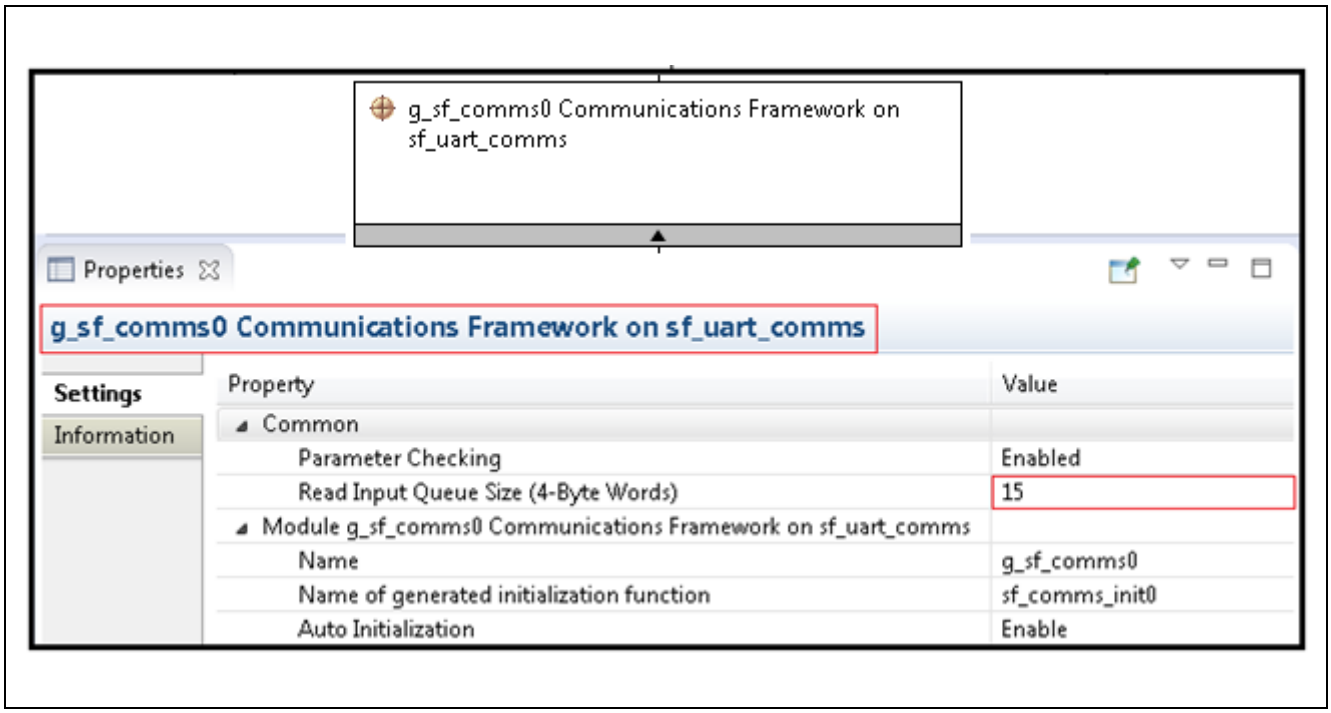

図30　セルラーの通信フレームワークとその設定

セルラーフレームワークの一部として設定されたパケットプール(Packet Pool)は、IPインスタンスの一部として設定されたパケットプールとは異なります。セルラーフレームワークのパケットプール設定は、以下の図31に示すとおりです。鍵記号は、デフォルトとしてシステムで設定されている値であり、ユーザが設定できないことを表しています。

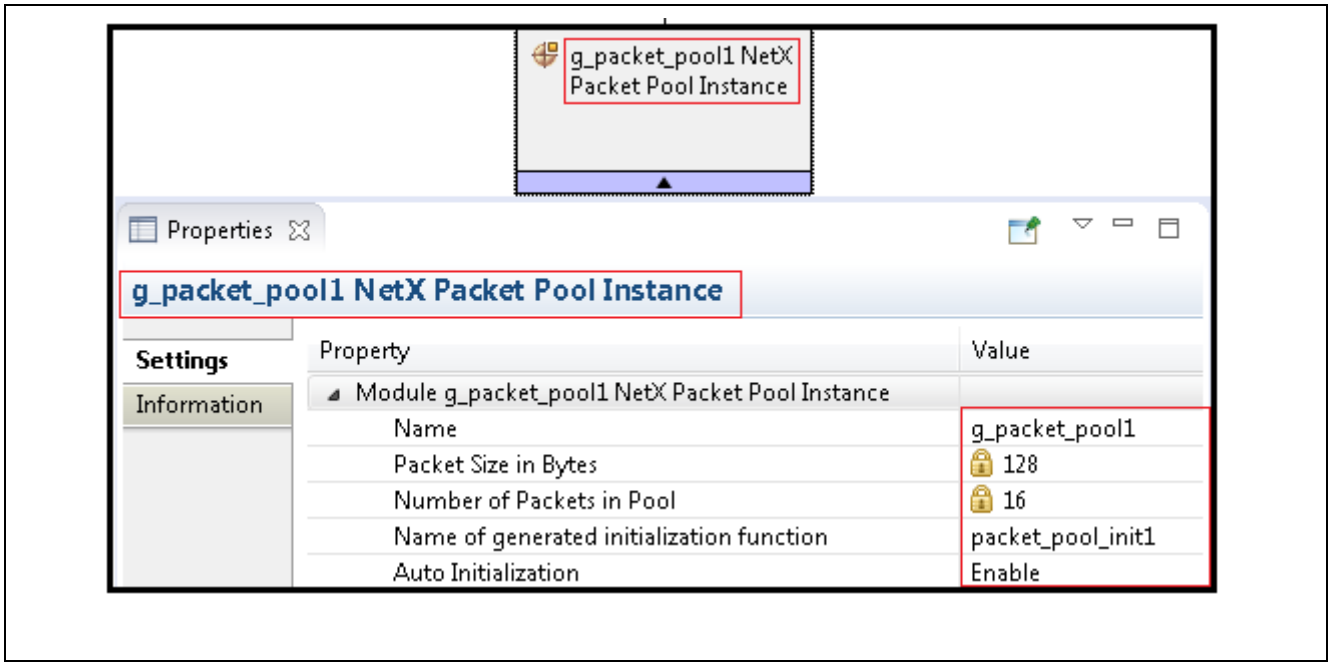

図31　セルラー データを処理するためのパケットプールとその設定

本アプリケーションのセルラーフレームワークはUARTを使用してモデムと通信します。以下の図32は、セルラーアプリケーションに関する設定の詳細を示しています。チャネル0はPK-S5D9ボードに使用されます。ボーレート(Baud rate)のデフォルトは115200です。ボーレートが異なるモデムの場合でも、ここで同じ値を設定することが可能です。UART設定の詳細については、*UART Module Guides*を参照し、r_sci_uartを検索してください。

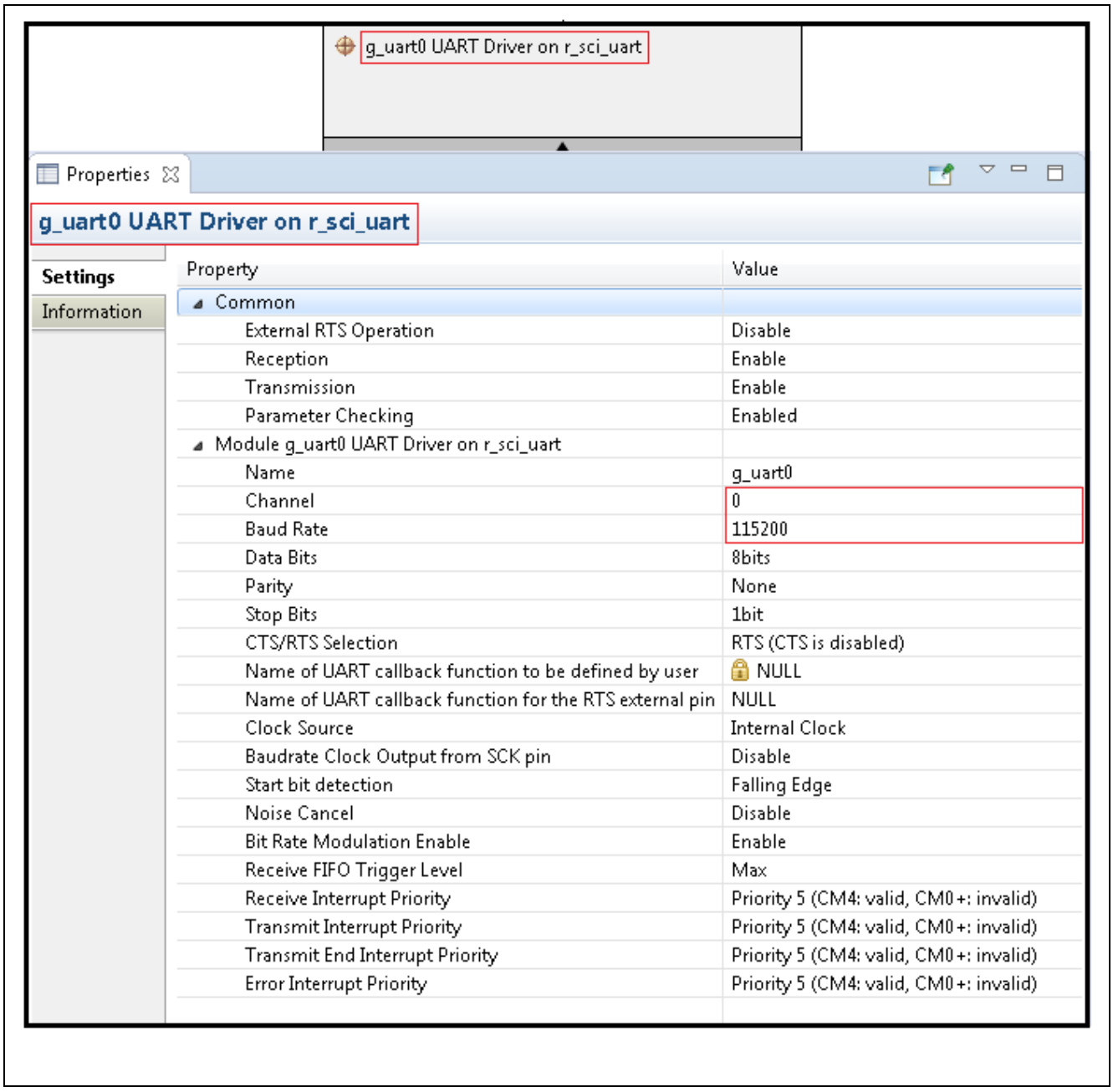

図32　UARTドライバとその設定

TxおよびRx用の転送ドライバ(Transfer driver)は、図33および図34に示すように設定されます。これはシステムによって設定され、ユーザがこれらの設定を変更することはできません。

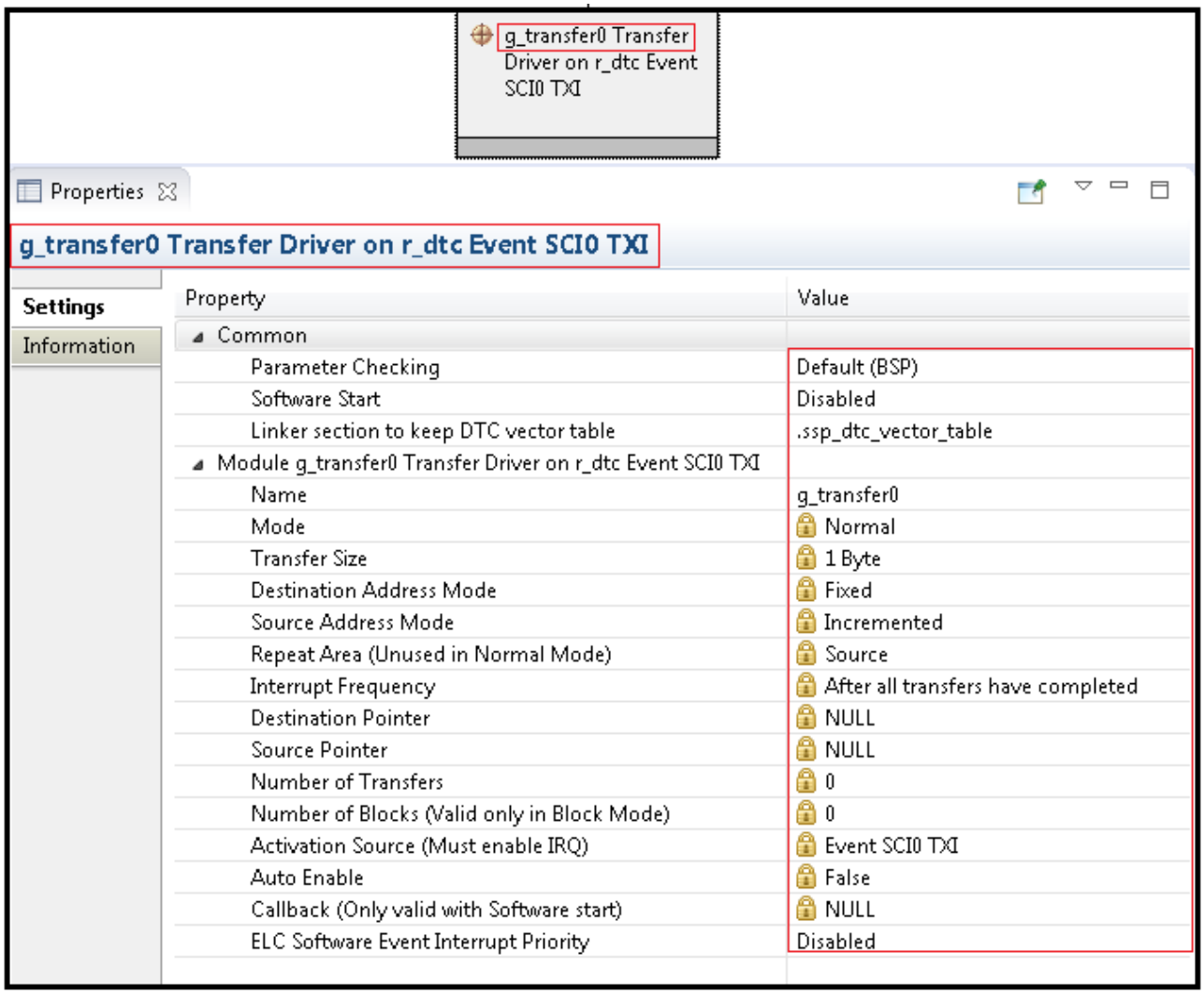

図33　TX転送ドライバとその設定

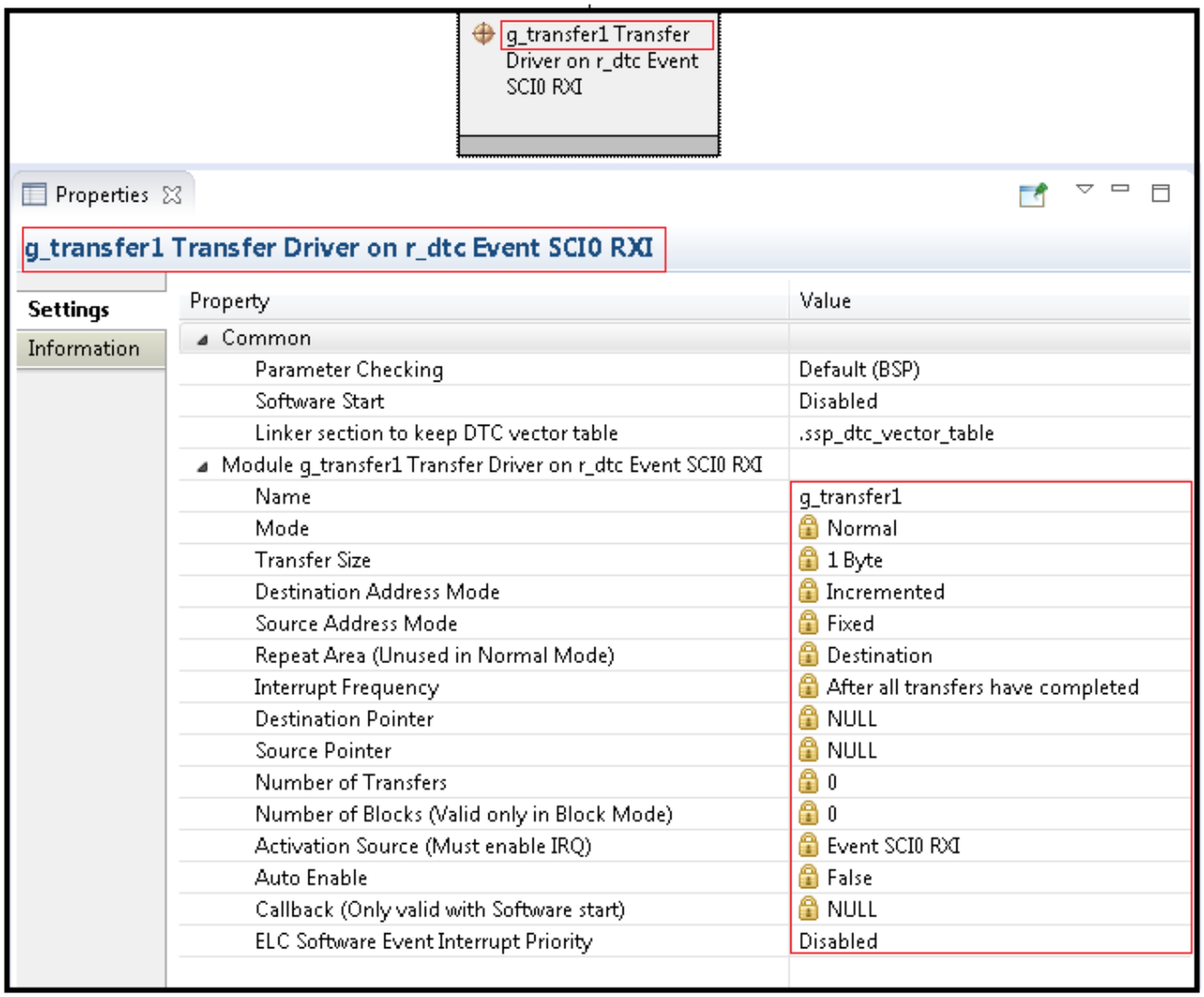

図34　RX転送ドライバとその設定

PMODBに接続されたセルラーモデムのリセット端子設定は、以下の図35に示すとおりです。

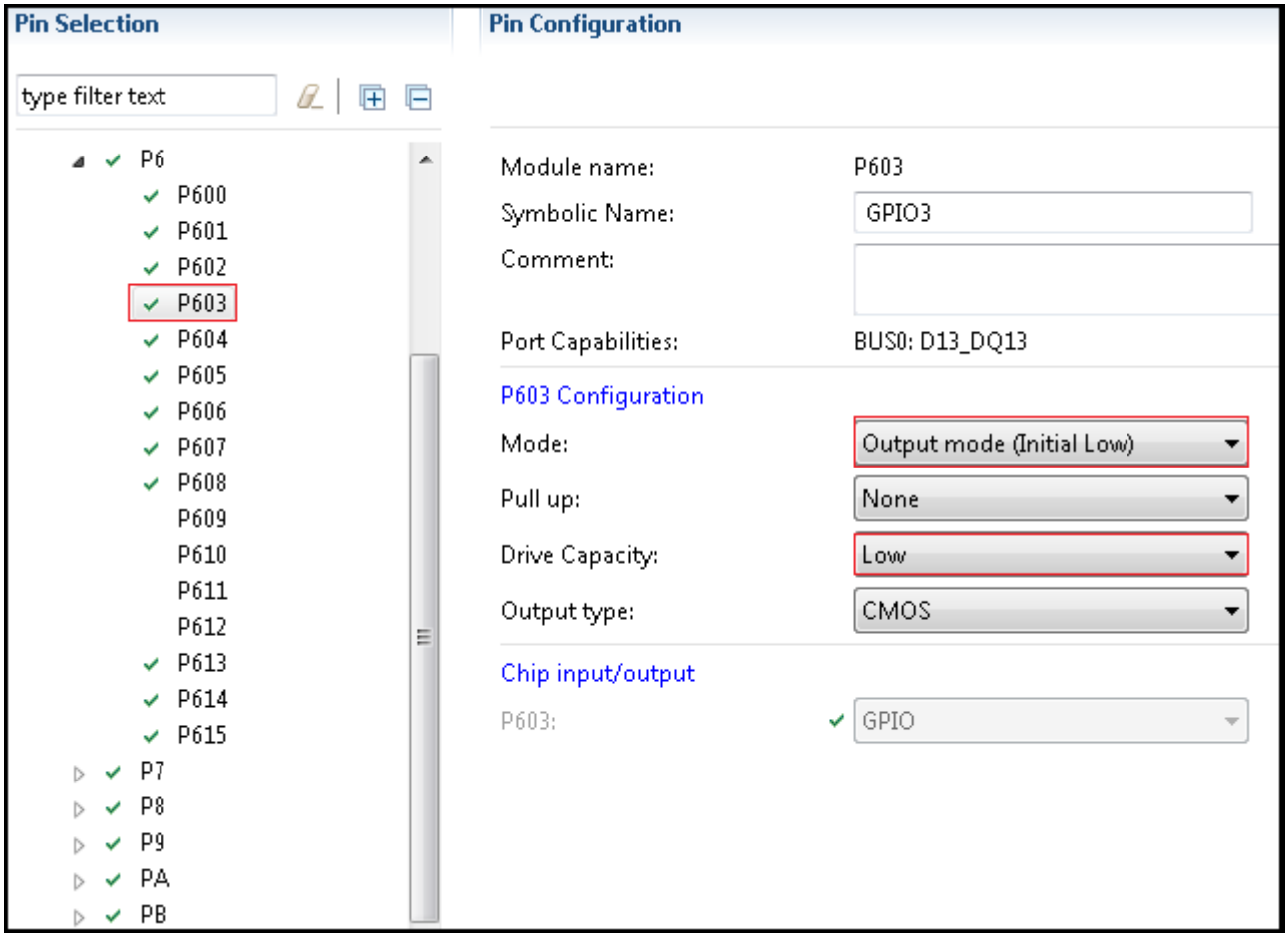

図35　セルラーハードウェアモジュールリセット端子（PMOD端子8）とその設定

7.1.3　セルラーフレームワークモジュールコードの概要(Cellular framework module code overview)

本項では、セルラーフレームワークのファイル構造とそのアプリケーションを以下に示します。プロジェクトがインポートされてプロジェクトの内容が生成されたら、より詳細に理解するためにコードを確認してください。

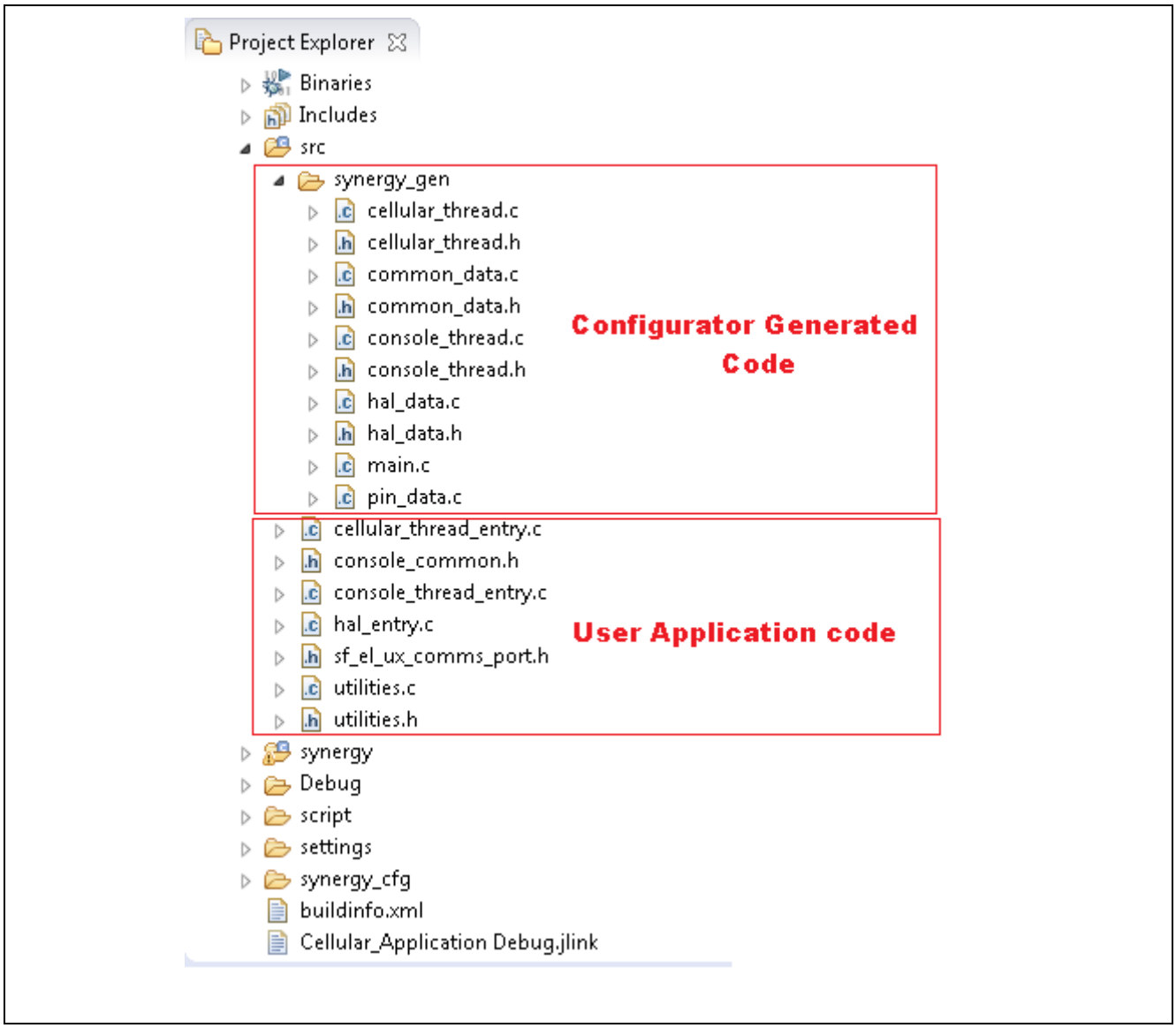

図36　セルラーアプリケーションコード編成の概要

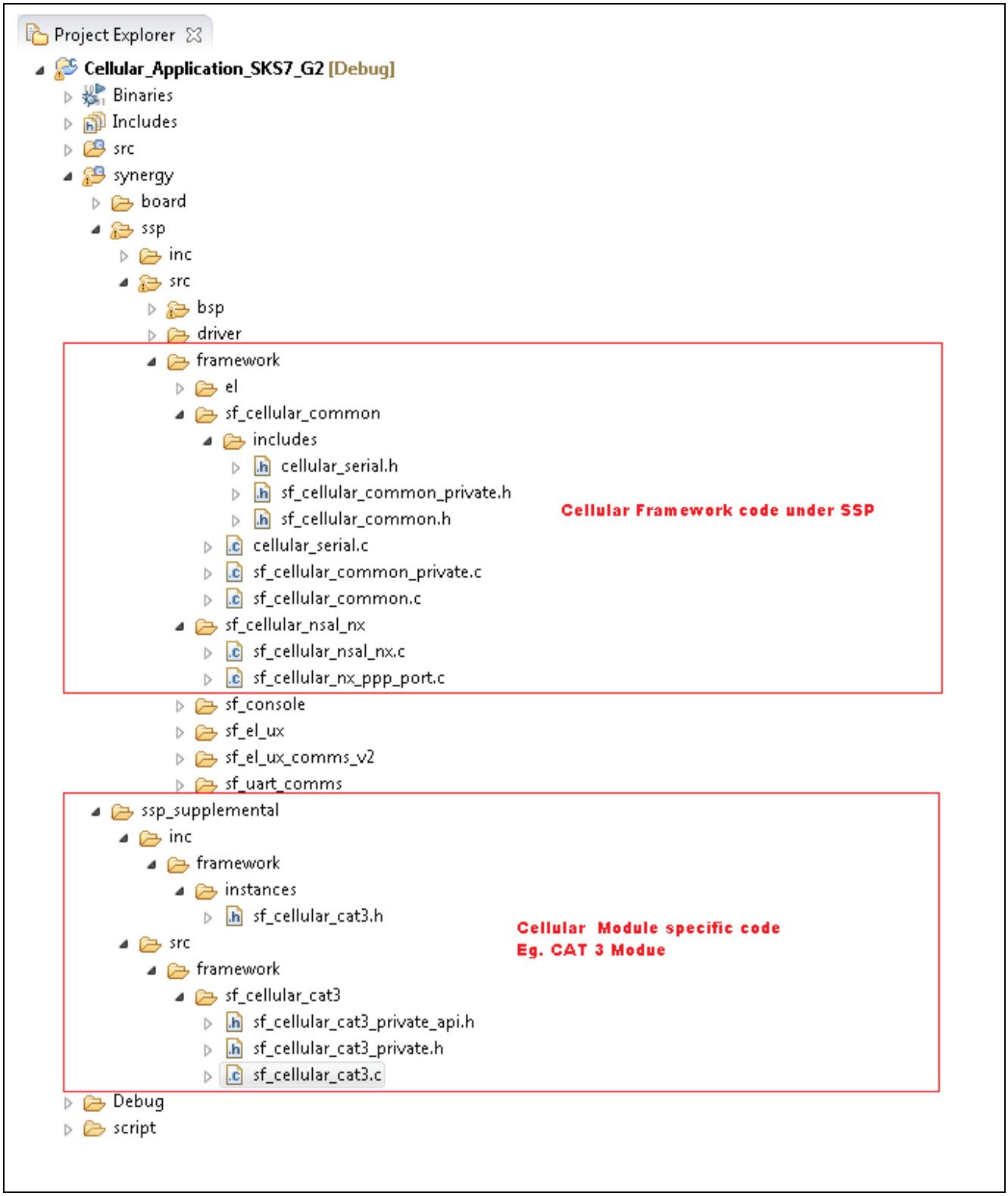

図37　セルラーフレームワークコード編成の概要（CAT3の場合）

図38　セルラーフレームワークコード編成の概要（CAT1の場合）

8. セルラー フレームワークモジュールアプリケーションプロジェクトの実行(Running the Cellular Framework Module Application Project)

セルラーアプリケーションプロジェクトを動作させ、対象キットで実行されていることを確認するには、（CAT1モデムまたはCAT3モデムに応じて）付属のアプリケーションプロジェクトをISDEにインポートするだけです。プロジェクトをe2 studioにインポートしてビルド／実行する手順については、バンドルの一部として付属している*SSP Import Guide*（英文：r11an0023eu0120-synergy-ssp-import-guide.pdf、和文（参考資料）：r11an0023ju0116-synergy-ssp-import-guide.pdf）を参照してください。

8.1 セルラーハードウェアモジュールのアクティベーション(有効化)と設定の詳細(Cellular harware module activation and setup details)

セルラーハードウェアモジュールにはSIMカードのスロットがあります。セルラーハードウェアモジュールがアクティベーションされていない場合には、セルラーハードウェアモジュールのIMEI番号およびSIMカードのSIM ID番号をメモしてください。これらは、セルラーハードウェアモジュールをアクティベーションするために必要になります。

サービス事業者に連絡して、セルラーモデムをM2Mネットワークに追加するよう依頼します。または、ユーザ側からサービス事業者のポータルアカウントを使用して、同じ作業を実行することも可能です。

SIMカードをSIMスロットに挿入します。サービス事業者はモジュールをアクティベーションし、サービス事業者のネットワークにデバイスを追加します。モジュールがネットワークに追加されたら、サービス事業者はAPN（アクセスポイント名）をモジュールに割り当てます。正常にアクティベーションされた後、これらの証明書が提供されます。

ユーザは、USB - TTL（USB- RS-232　3.3Vシリアル）を使用してモデムをPCに接続することで、モデムがアクティベーションしているかどうかを確認することも可能です。詳細なコマンドについては、ATコマンドセットのマニュアルを参照してください。

プロジェクトを実行する前に、上記の図39に示すように、CAT3またはCAT1セルラーハードウェアモジュールをPK-S5D9のPMODBに接続する必要があります。セルラーハードウェアモジュールは、本書の参考情報の項に記載されている参照リンクから購入することが可能です。

注：cellular_thread_entry.c内部のAPN名を変更する必要があります。（DEFAULT_APN_NAME）を探し、アクティベーションされたモジュールのAPN名に置き換えてください。サンプルのAPN名を以下に示します。

```
#define DEFAULT_APN_NAME            "VZWINTERNET"
```

注：ユーザは、アクティベーションされたモデムのコンテキストIDとPDPコンテキストもメモする必要があります。これらは、サービス事業者による割り当てで必要になります。

注：IMEI番号とSIM番号は結合されている場合があるため、SIMを別のセルラー ハードウェアモジュールで交換しても動作しない可能性があります。

注：APN名はサービス事業者によって変わります。「VZWINTERNET」はベライゾン北米向けです。

注：ユーザは、地域およびサービス事業者に適したNimbeLinkのウェブサイトからモデムを確認する必要があります。

8.2　PK-S5D9ボード設定の詳細 (PK-S5D9 Synergy MCU board setup details)

図39に示すように、ジャンパ（J15）を使用するPMOD Bに3.3Vが選択されていることを確認してください。

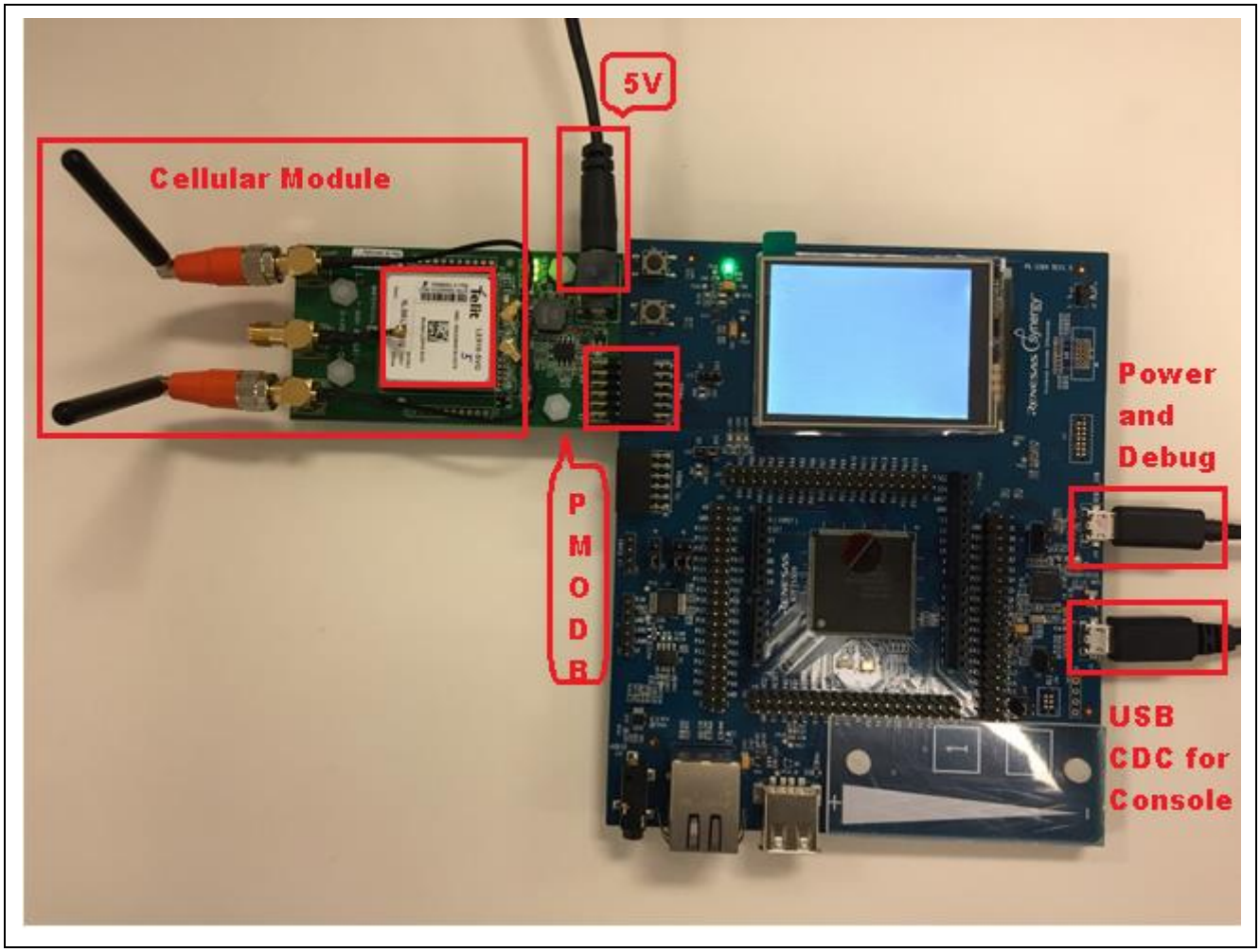

図39　セルラーハードウェアモジュールのハードウェア設定

注：　モジュールに3.3Vを選択することが重要です。そうしないと、モジュールが破損する可能性があります。

指示どおりにジャンパを設定した後：

- セルラー ハードウェアモジュール（CAT1またはCAT3）をPMOD Bコネクタに接続します。
- マイクロUSBケーブルをJ19ポートに接続して、ボードの電源を投入します。
- セルラーハードウェアモジュールには5Vの追加電源が別途必要です（これがないと動作しません）。

J5からPCにUSBデバイスを接続します。

8.3　サンプルアプリケーションの実行(Run the sample application)

セルラーフレームワークアプリケーションプロジェクトを実行するには、以下の手順に従います。

1. インポート(Import)が終了すると、アプリケーションのビルドとイメージのダウンロードが実行されます。
2. アプリケーションのデバッグを開始します。

3. Tera Termコンソールを開きます。PCでUSB CDCが検出された場合、Tera Termコンソールに出力を表示することが可能です。検出されない場合には、適切なドライバをインストールする必要があります（8.4項を参照してください）。
4. ユーザは目的のIPアドレスにpingを実行するコマンドを選択する必要があります。
5. ユーザインタフェースおよびPingコマンドの実行については、サンプルスナップショットを参照してください。

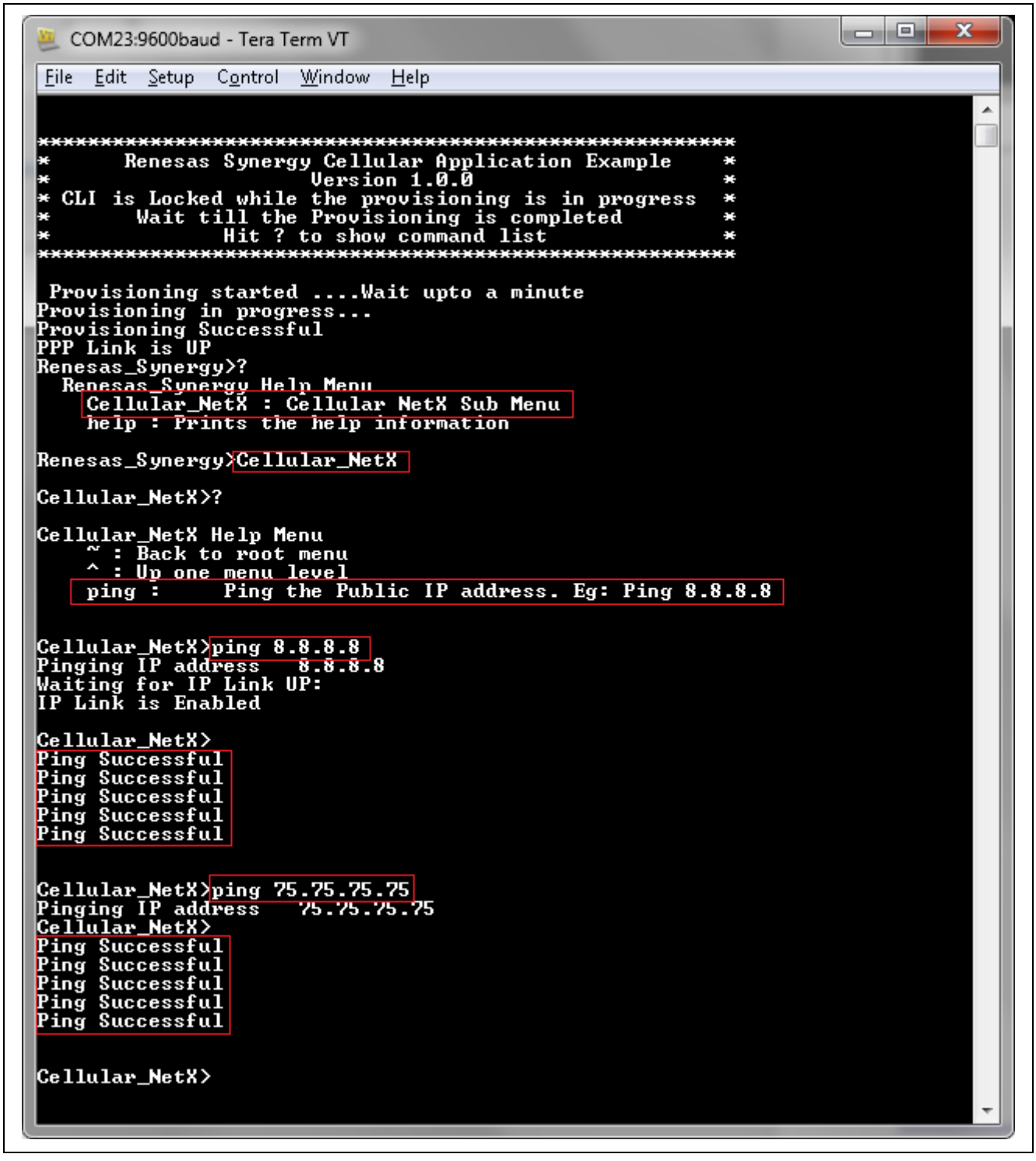

図40　セルラーフレームワークアプリケーションプロジェクトのサンプル出力

注： Windows 10でテストする場合、USBXデバイス設定クラスコードをCommunicationsからMiscellaneous に変更する必要があります。そのためには、「Threads」タブの「Console Thread」に移動し、USBX デバイス設定のクラスコードプロパティ(Class Code property)を変更します。続いて、イメージを 再ビルドします。

8.4 USB CDCデバイスドライバのインストール(Install the USB CDC device driver)

本アプリケーションプロジェクトのコンソールフレームワーク(console framework)は、USB CDCデバイスの通信フレームワーク(communication framework)を使用します。このためには、USB CDCデバイスドライバがPCにインストールされている必要があります。

Windows 10 PCの場合、以下の図41に示すようにPK-S5D9をUSBシリアルデバイスとして検出できるため、USB CDCデバイスドライバ(USB CDC device driver)をインストールする必要はありません。

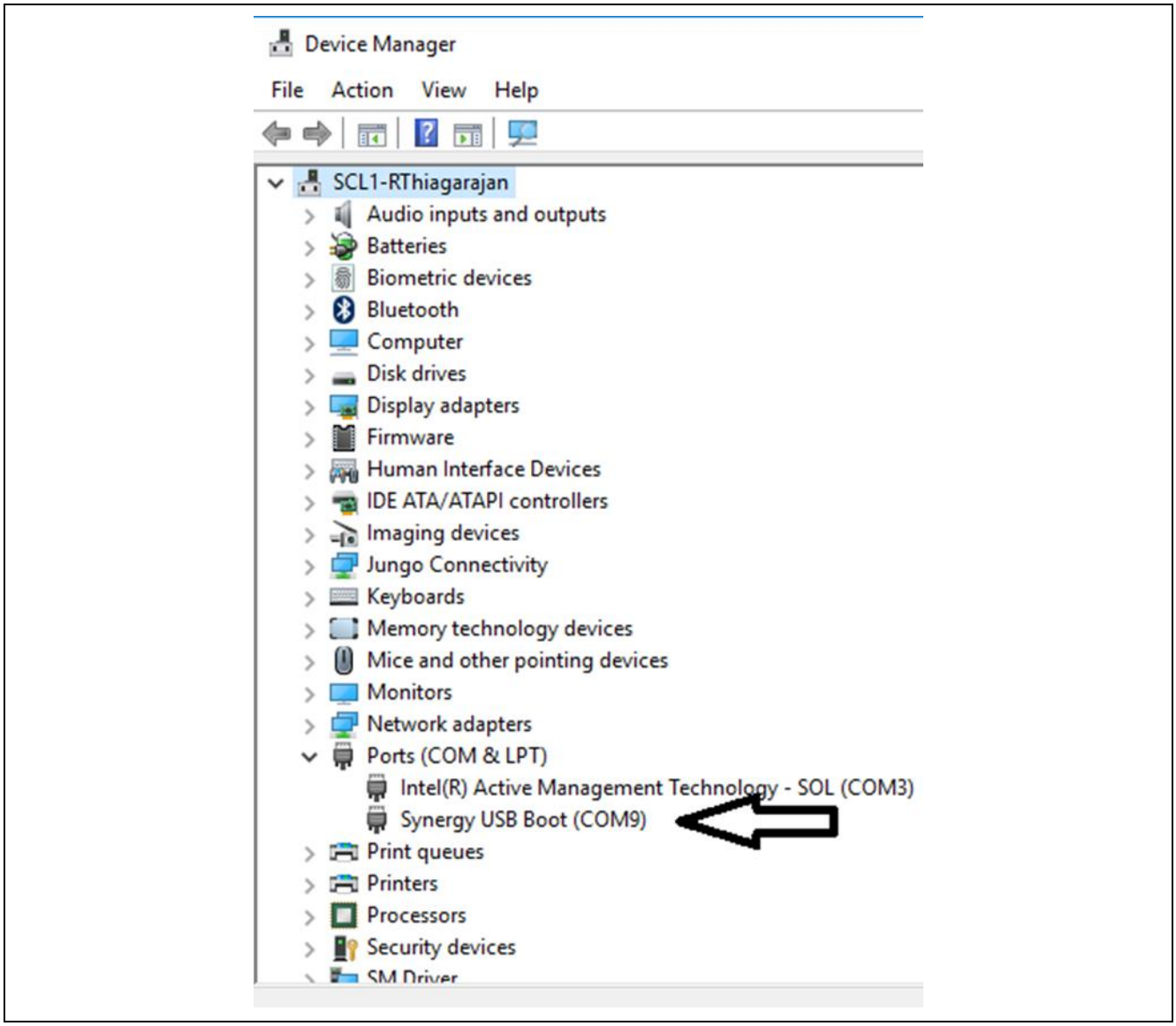

図41　Windows 10のUSB CDCポートの一覧

Windows 7の場合、SK-S7G2 USBデバイスポートはPCに接続された後に**不明なデバイス(Unknown Device)**として最初に検出されます。次に、このデバイスを右クリックし、「**Update Driver software**」を選択します。

ドライバの場所の入力を求められたら、本アプリケーションプロジェクトの一環として提供されるWindows USBシリアルドライバ(USB serial driver)の場所を参照します。ドライバが更新された後、Windows 10にあるような新しいCOMデバイスがデバイスマネージャーに表示されます。

9. セルラーフレームワークモジュールの終了(Cellular Framework Module Conclusion)

本アプリケーションノートで、サンプルプロジェクトでセルラーフレームワークモジュールを選択、追加、設定、および使用するために必要な情報を提供しました。これらの手順の多くは、従来の組み込みシステム

では、時間がかかり間違いを犯しやすい作業でした。Renesas Synergy™プラットフォームにより、これらの作業に要する時間が大幅に短縮され、一般的なエラー（設定の競合、ロウレベルドライバの誤選択など）が取り除かれます。本アプリケーションノートで提示するようなハイレベルAPIを使用することで、高いレベルで作業を開始できるようになり、従来の開発環境を使ったり、ロウレベルドライバを作成するのに要した時間を無くし、開発時間がさらに節減されます。

10. セルラーフレームワークモジュールの次の手順(Cellular Framework Module Next Steps)

簡単なセルラーフレームワークプロジェクトに習熟した後は、より複雑なサンプルを検討してみるとよいでしょう。他のセルラーベースのアプリケーションについては、Renesas Synergy™ウェブ（https://www.renesas.com/ja-jp/products/synergy.html）にアクセスしてください。

11. 参考情報(Reference Information)

*SSPユーザーズマニュアル：*SSPディストリビューションパッケージのhtml形式で利用可能であり、Renesas Synergyギャラリーからpdfとしても利用可能です。最新の参考資料とその場所を確認するには、Synergyナレッジベースにアクセスしてモジュール名を検索し、**module guide references**を検索に含めます。

たとえば、r_docモジュールの参考情報を探している場合、https://en-us.knowledgebase.renesas.com/English_Content/Renesas_Synergy%E2%84%A2_Platform/Renesas_Synergy_Knowledge_Baseにアクセスし、検索バーに**r_doc module guide references**と入力します。検索により結果のリストが表示され、先頭はモジュールガイドのリファレンスページになります。以下のURLでは、この例の検索結果が直接表示されます。

https://en-us.knowledgebase.renesas.com/Special:Search?fpid=230&search=r_doc%20module%20guide&path=&limit=55&page=1&q=r_doc%20module%20guide%20references&tags

アプリケーションサンプルを実装するための実践的な手法として（単に学習用の参考情報としてではなく）本ガイドを利用したいユーザは、ISDE（適切なバージョンのSSPを含むe² studio ISDEまたはIAR Embedded Workbench® for Renesas Synergy™）をPCにインストールして運用している必要があります。この前提条件を満たすには、*Getting Started Guide for the Renesas Synergy Platform*に従います。

- サンプルプロジェクトがSynergy MCUで動作させるには、ISDEおよびSSPに加えて、ハードウェアターゲットも必要になります。このためには、Renesas認定ディストリビュータからキットを購入します。ディストリビュータのウェブサイトでキットを探すには、ディストリビュータの検索ウィンドウでキット名を検索するだけです。アプリケーションサンプルプロジェクトの対象となるキットは以下のとおりです。
  — PK-S5D9キット

北米市場（ベライゾンサービス事業者）の場合、ユーザはSynergy MCUキットに加えてNimbeLink PMODアダプタ、セルラー ハードウェアモジュール、およびモジュールでサポートされているSIMも購入する必要があります。これらは以下のリンクから購入することが可能です。

https://www.digikey.com/product-detail/en/NL-SW-LTE-TSVG/1477-1011-ND/4977073

https://www.digikey.com/product-detail/en/nimbelink-llc/NL-AB-PMOD-SYN/1477-1038-ND/5825469

https://www.digikey.com/products/en?keywords=ANT%20EXT%20GPS%2FLTE%20SMAM%20HINGED%2072MM

https://www.digikey.com/products/en?keywords=SIM%204G%20VERIZON%203FF

注：各地域でサポートされているモデムの詳細については、http://nimbelink.com/も参照してください。

本ガイドは、プロジェクトの作成、スレッドの追加、スタックの作成、モジュールの設定、実行、およびデバッグにSynergy ISDEとSSPを使用する方法を熟知しているユーザを前提としています。スタートアップガイド（e² studio ISDE、IAR Embedded Workbench® for Renesas Synergy™、およびSSP向け）を読んだり、入門ビデオを見たり、個別指導の実習を受けたりすることで、この前提条件を満たすことが可能です。

Synergy ウェブ

https://www.renesas.com/ja-jp/products/synergy.html

はじめてのSynergy開発 導入支援サイト

https://www.renesas.com/ja-jp/promotions/implementation/synergy.html

Synergyナレッジベース：

https://ja-jp.knowledgebase.renesas.com/Japanese_Content/Renesas_Synergy%E2%84%A2_%E3%83%97%E3%83%A9%E3%83%83%E3%83%88%E3%83%95%E3%82%A9%E3%83%BC%E3%83%A0

ホームページとサポート窓口

サポート：　https://synergygallery.renesas.com/support

テクニカルサポート：

- アメリカ：　https://www.renesas.com/en-us/support/contact.html
- ヨーロッパ：　https://www.renesas.com/en-eu/support/contact.html
- 日本：　https://www.renesas.com/ja-jp/support/contact.html

改訂履歴

Rev.	発行日	改訂内容	
		ページ	ポイント
1.03	2018.08.07		第1.03版 発行 英文版（資料番号r30an0311eu0103-synergy-cellular-framework、Rev1.03、発効日2018年4月2日）を翻訳 Renesas Synergy ウェブを日本語版に変更 アプリケーションノート参考例リンク先に日本語版を追加

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/