RTA-OS

入門ガイド

著作権について

Document: 10671-GS-5.4.0 JP-05-2015

安全に関する注意事項

このETAS製品は標準的な品質管理要件を満たしています。特定の安全基準（IEC 61508、ISO 26262など）を満たす必要がある場合は、それらの要件を明確に定義してETASにご注文いただく必要があります。また製品を使用するユーザーは、使用に先だって安全基準が遵守されていることを確認する必要があります。

目次

1 RTA-OS について

本書では、RTA-OS のインストール方法や RTA-OS の機能概要を紹介し、さらにサンプルアプリケーションを使用して、RTA-OS が正しくインストールされ機能しているかをチェックする方法を説明します。

RTA-OS はコンパクトで高速なリアルタイムオペレーティングシステムで、各種標準規格（AUTOSAR R3.x、 AUTOSAR R4.0、OSEK/VDX OS 2.2.3）に準拠しています。オペレーティングシステムのコンフィギュレーション設定とビルドには PC を使用し、ターゲットとするハードウェアプラットフォームを指定して行います。

RTA-OS は以下のようなメインパーツで構成されます。

1. PC 用ツール

   RTA-OSツールには以下のモジュールが含まれ、すべてのターゲットで使用されます。

   rtaoscfg: RTA-OSのコンフィギュレーションを設定するためのグラフィカルコンフィギュレーションエディタ。

   rtaosgen： コンフィギュレーションに応じたRTA-OSカーネルライブラリを生成するためのコマンドラインツール。全ターゲットにより共有されるRTA-OSカーネルのパーツも含まれます。

   rtaosanvis： グラフィカルスケジューラビリティ分析ツール。構築されたOSのタイミング挙動のモデリング を行い、デッドライン要件が満たされるかどうかをチェックします。

2. ポートプラグイン

   RTA-OSの各ポートプラグイン（RTA-OS Targets）は、特定のターゲット（マイコンとコンパイラの組み合わせ）にRTA-OSを対応させるためのパッケージです。PCツールは複数のターゲットをサポートできるので、コンフィギュレーション設定時にターゲットを切り替えることができます。

   RTA-OSツールをインストールすると、「VRTA」と呼ばれる仮想ターゲット（Virtual Target）用ポートプラグインもインストールされます。組み込みターゲットハードウェアがない場合でも、VRTAを利用することによりPC上で稼働するRTA-OSアプリケーションをビルドすることができます。

1.1 本書の対象ユーザー

本書は、プリエンプティブなオペレーティングシステムを使用するリアルタイムアプリケーションを構築しようとする組み込みシステム開発者を対象としています。C プログラミング言語の知識は必須で、要件に合わせて選択されたツールチェーンを使用して組み込みアプリケーション用 C コードのコンパイル、アセンブル、リンクを行えることが前提条件となります。またターゲットマイクロコントローラについての基礎知識（先頭アドレス、メモリレイアウト、周辺機器のロケーションなど）も不可欠です。

さらに、Microsoft Windows オペレーティングシステムの一般的用法（ソフトウェアのインストール、メニューアイテムの選択、ボタンクリックの操作、ファイルやフォルダのナビゲートなど）も熟知している必要があります。

1.2 表記上の規則

本書では以下の表記を使用しています。

File > Open を選択します。	メニューオプションは**青い太字（bold）**で表記しています。
OK をクリックします。	ボタンのラベルは**太字（bold）**で表記しています。
<ENTER>を押します。	キーボードコマンドは山括弧で括って表記しています。
“Open file”ダイアログボックスが開きます。	GUI エレメント名（ウィンドウのタイトルやフィールドなど）は、二重引用符で括って表記しています。
`Activate(Task1)`	プログラムコード、ヘッダファイル名、C データ型名、C 関数、および API 関数名は、すべて `Courier`（クーリエ）体で表記しています。
1.3 項を参照してください。	文書内の別の箇所にジャンプするハイパーリンクは青字で表記しています。
	RTA-OS の機能のうち、他の AUTOSAR OS 実装に移植できない可能性のあるものには、RTA-OS アイコンを付けています。
	RTA-OS を正しく機能させるために必ず従わなければならない重要な指示には、警告標識を付けています。

1.3 参考資料

OSEK はヨーロッパの自動車産業界の標準規格策定を目標とするプロジェクトです。自動車エレクトロニクス用のオープンシステムインターフェースを作成しています。OSEK 規格についての詳しい情報は以下のサイトを参照してください。

`http://www.osek-vdx.org`

AUTOSAR（AUTomotive Open System ARchitecture）は、標準化されたオープンな自動車用ソフトウェアアーキテクチャです。自動車メーカー、サプライヤ、およびツール開発者についての詳しい情報は以下のサイトを参照してください。

`http://www.autosar.org`

2 RTA-OS のインストール

2.1 インストールの準備

RTA-OS をインストールする前に、すべてのアイテムが納品されていることを確認してください。RTA-OS ツールは必ずインストールする必要があり、 RTA-OS を組み込みターゲットハードウェア上で稼働させるには、そのターゲット用のポートプラグイン（つまり仮想ターゲット以外のポートプラグイン）のインストールパッケージも必要です。仮想ターゲット（VRTA）については 2.2.2 項を参照してください。

RTA-OS ツールは 1 つのインストーラとして CD で提供され、各組み込みターゲット用のポートプラグインはそれぞれ別の独立したインストーラとして提供されています。RTA-OS を使用する際には、ツールと 1 つ以上のポートプラグインをインストールする必要があります。

2.1.1 ハードウェア要件

RTA-OS をインストールして使用するには、以下のハードウェア要件が満たされた PC が必要です。

- PC タイプ： IBM 互換 PC、1GHz Pentium（またはそれ以上）
- メモリ（RAM）： 512MB
- ハードディスク空き容量： 500MB
- ディスクドライブ： CD-ROM または DVD ドライブ
- イーサネットアダプタ（ネットワークカード）

2.1.2 ソフトウェア要件

RTA-OS をインストールして使用するには、PC に以下のいずれかのバージョンの Microsoft Windows がインストールされている必要があります。

- Windows 2000（Service Pack 3 以降）
- Windows XP（Service Pack 2 以降）
- Windows Vista
- Windows 7

統合のためのアドバイス 2.1： RTA-OS 用ツール を使用するシステムには、Microsoft の.NET Framework V2.0 および V4.0 がインストールされている必要があります。RTA-OS をインストールする前に、必ずこれらをインストールしておいてください。
.NET Framework は RTA-OS の製品パッケージには含まれていません。
http://www.microsoft.com/net/Download.aspx から入手してください。

RTA-OS ツールの性能向上および保守上の理由から、今後（時期は未定）、V2.0 から V4.0 への移行が行われる予定です。

RTA-OS を実行するには、インストール後にライセンスキーを登録する必要があります。ライセンスは ETAS ライセンスマネージャ（ ETAS License Manager）により管理されます。詳細は 2.4 項を参照してください。

2.2 インストール

RTA-OS インストーラは以下のインストールを行います。

1. RTA-OS 用の PC ツール
2. RTA-OS のドキュメント（任意）

2.2.1 ツール

VRTA やポートプラグインをインストールするには、前もって RTA-OS の PC ツールをインストールしておく必要があります。これにより所定のフォルダ構造が作成されます。PC ツールは以下の手順でインストールします。

1. 以下のいずれかを行います。
   - 実行イメージを実行します。
   - RTA-OS の製品 CD を PC の CD-ROM または DVD ドライブに挿入します。

     CD をドライブに挿入してもインストールプログラムが自動的に実行されない場合は、CD/DVD ドライブのルートフォルダにある `autostart.exe` をマニュアル操作で実行してください。
2. 画面に表示される指示に従って RTA-OS をインストールします。

RTA-OS のデフォルトのインストール先は `C:\ETAS\RTA-OS` です。このロケーションはインストール処理中に変更できます。

インストールが完了すると、以下の追加コンポーネントをインストールするかどうかが尋ねられます。

1. VRTA 仮想ターゲット用ポートプラグイン： 2.2.2 項を参照してください。
2. ETAS ライセンスマネージャ（ETAS License Management）ソフトウェア： 2.4 項を参照してください。

統合のためのアドバイス 2.2： RTA-OS 開発ツールを使用するには、いずれかのポートプラグインを 1 つ以上インストールする必要があります。ツールのインストール時に VRTA ポートプラグインのインストールについて提示されるので、それをデフォルトターゲットとしてインストールしておくことをお勧めします。

統合のためのアドバイス 2.3： 追加コンポーネントのインストールは、インストール先フォルダのサブフォルダ内にあるインストーラを実行することにより行うことができます。

2.2.2 VRTA

VRTA を使用すると、実際の組み込みハードウェア（マルチコアハードウェアを含む）上で実行されるアプリケーションの挙動を Windows PC 上でエミュレートする AUTOSAR アプリケーションを作成することができます。これにより、組み込みツールやターゲットハードウェ

アがなくても、完全な RTA-OS コンフィギュレーションの開発やテスト、デバッグを行えます。

RTA-OS をインストールすると、デフォルトにおいて `C:\ETAS\RTA-OS\Targets\VRTA_n.n.n` に VRTA 用ポートプラグインがインストールされます。さらに VRTA は、VRTA ベースのアプリケーションのランタイムサポートを提供する一連のライブラリと実行ファイルを `C:\ETAS\RTA-OS\bin` にインストールします。

統合のためのアドバイス 2.4： ポートプラグイン用のデフォルトのインストールパスは、ツールのインスタレーションパスと異なる場合がありますが、任意にパスを変更してツールと同じパスにインストールすることもできます。

ポートプラグインは任意のロケーションにインストールできますが、デフォルト以外のフォルダを使用する場合は、**rtaosgen** と **rtaoscfg** を実行する際に、以下のように `--target_include` 引数を使用してパスを指定する必要があります。

```
rtaosgen --target_include:<target_folder>
```

統合のためのアドバイス 2.5： VRTA のランタイムライブラリと実行ファイルは、VRTA アプリケーション実行用のパスにインストールされます。このパス（`<install dir>\bin`）は、あらかじめ Windows の PATH 環境変数に登録しておくことをお勧めします。また、RTA-OSEK がすでにインストールされている PC に RTA-OS をインストールした場合は、RTA-OS のパスを RTA-OSEK のパスよりも前に定義する必要があります。

VRTA 用コンパイラ

VRTA は、以下のような一般的な PC 用 C/C++コンパイラに対応しています。

- MinGW /gcc
- Microsoft Visual Studio 2005
- Microsoft Visual Studio 2008
- Microsoft Visual Studio 2010

これらのコンパイラは RTA-OS のインストールパッケージには含まれていません。

MinGW（Minimalist GNU for Windows）という C コンパイラは、`http: //www.mingw.org` から入手できます。このコンパイラは、VRTA ベースのアプリケーションのビルドにおいてデフォルトコンパイラとして使用されます。

Microsoft は、Visual Studio C++コンパイラの Express Edition を提供しています。これは、`http://www.microsoft.com/express/ vc` から自由にダウンロードできます。

2.2.3 ポートプラグイン

ポートプラグインのインストールは、ツールと同じ方法で行います。

1. 以下のいずれかを行います。
   - 実行イメージを実行します。
   - RTA-OS の製品 CD を PC の CD-ROM または DVD ドライブに挿入します。

CD をドライブに挿入してもインストールプログラムが自動的に実行されない場合は、CD/DVD ドライブのルートフォルダにある `autostart.exe.` をマニュアル操作で実行してください。

2. 画面に表示される指示に従ってポートプラグインをインストールします。

統合のためのアドバイス 2.6： ポートプラグイン用のデフォルトのインストールパスはツールのインストールパスと異なる場合があります。これは、過去にリリースされたポートプラグインと完全に適合させるためですが、任意にパスを変更してツールと同じパスにインストールすることもできます。

ポートプラグインは任意のロケーションにインストールできますが、デフォルト以外のフォルダを使用する場合は、**rtaosgen** と **rtaoscfg** を実行する際に、`--target_include` 引数を使用してパスを指定する必要があります。

2.3 インストールされる内容

RTA-OS をインストールすると、インストール時に指定されたロケーションの下に所定のフォルダ構造が作成されます。デフォルトロケーションは `C:\ETAS\RTA-OS` です。

作成されるフォルダには、主に以下のようなものがあります。

フォルダ	内容
`Bin`	実行プログラム
`Bin\Licenses`	インストールしたプログラムに必要なライセンスの種類を ETAS ライセンスマネージャに通知するためのライセンスシグネチャファイルが格納されます。ライセンスファイルは別の場所にインストールされます。詳細は 2.4 項を参照してください。
`Bin\plugins`	**rtaoscfg** コンフィギュレーションツール用 GUI プラグイン
`Documents`	ユーザードキュメント
`Targets`	ポートプラグイン ― ターゲットごとに 1 つずつサブフォルダが作成されます。各サブフォルダにはポートの固有パーツ（ポート用の『Target/Compiler Port Guide』を含む）が格納されます。
`VRTA`	VRTA ポートプラグインのインストーラが格納されます。
`LicenseManager`	ETAS ライセンスマネージャのインストーラが格納されます。

各ポートプラグインは Targets の下の専用のサブフォルダにインストールされます。各サブフォルダには、ポートのプラグイン DLL とポート固有のドキュメントが格納されます。

すべてのユーザードキュメントは PDF 形式なので、Adobe Acrobat Reader で読むことができます。Adobe Acrobat Reader は RTA-OS には同梱されていませんが、`http://www.adobe.com` から自由に入手できます。

これらのフォルダの名前は以下の規則に従って付けられています。

`<Target><CompilerVendor>_n.n.n`[1]

サフィックス `n.n.n` はポートプラグインのバージョンです。これにより、1 つのポートプラグインについて複数のバージョンをインストールしておくことができます。

同じポートプラグインについて 2 つ以上のバージョンがインストールされている場合は、コンフィギュレーションで特定のバージョンが指定されていないかぎり、RTA-OS は最も新しいバージョン（つまり最も大きい番号のバージョン）を使用します。

2.4 ライセンス管理

> **注記：**
> ETAS 製品のライセンス管理の形態や運用方法は、機能や利便性の向上のため、適宜変更される場合があります。ライセンス管理についてご不明の点は、ETAS のサポート窓口までお問い合わせください。

RTA-OS は FLEXnet ライセンシングテクノロジーにより保護されています。RTA-OS を使用するには有効なライセンスキーが必要です。

製品のライセンスは ETAS ライセンスマネージャ（ETAS License Manager）により管理されます。ETAS ライセンスマネージャは、現在インストールされているライセンスとそのインストール先を常に把握しています。RTA-OS とプラグインに必要なライセンスの種類は、`<install_folder>\bin\Licenses` というフォルダに格納されたライセンスシグネチャファイルによって示されます。

ETAS ライセンスマネージャには、ライセンスに関する以下のような情報が表示されます。

- インストールされている ETAS 製品
- 製品または機能を使用するために必要なライセンス名
- ライセンスの状態（有効／無効など）
- ライセンスの有効期限
- 使用しているライセンス区分（ローカルライセンスまたはサーバーライセンス）

図 2-1 は、ETAS ライセンスマネージャのユーザーインターフェースです。

[1] VRTAは複数のコンパイラをサポートしているので、VRTAのポートプラグインにはこの命名規則が適用されず、VRTA–n.n.nという名前になります。

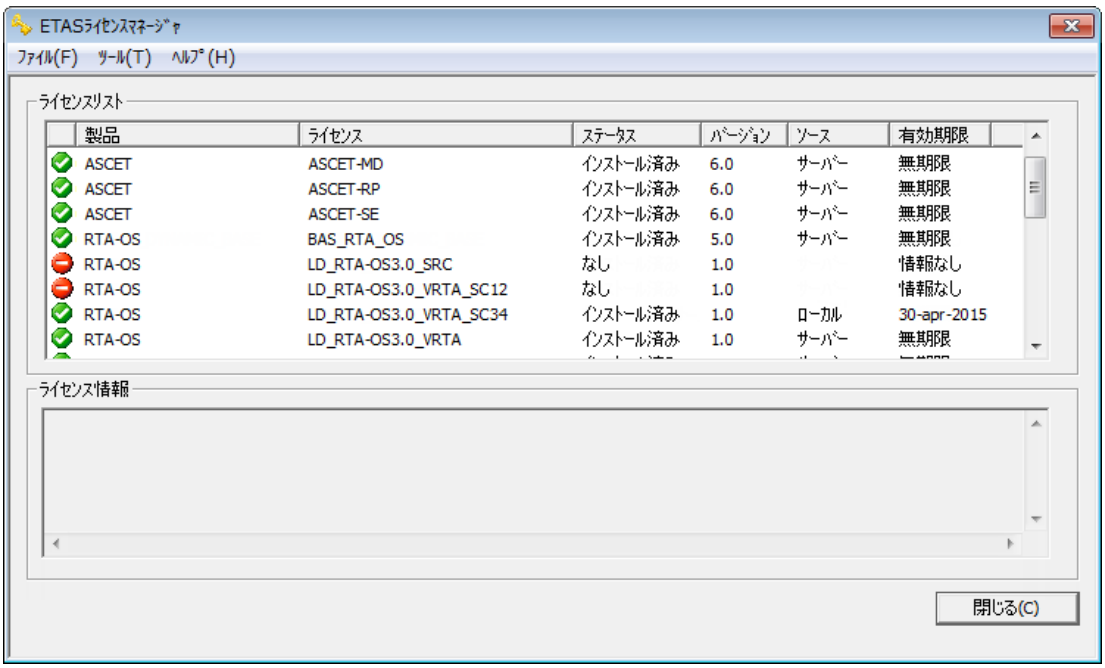

図 2-1　ETAS ライセンスマネージャ

2.4.1　ETAS ライセンスマネージャのインストール

統合のためのアドバイス 2.7：　RTA-OS を稼働させるには、PC に ETAS ライセンスマネージャがインストールされている必要があります。RTA-OS のインストール中に ETAS ライセンスマネージャもインストールするようにしてください。

ETAS ライセンスマネージャのインストーラには、以下の 2 つのコンポーネントが含まれています。

1. ETAS ライセンスマネージャ（メインプログラム）
2. 再配布可能な一連の FLEXnet ユーティリティ。これらのユーティリティには、コンカレントライセンスを使用する際に FLEXnet ライセンスサーバーマネージャをセットアップして実行するために必要なソフトウェアと指示書が含まれています。詳細は 2.4.2 項と 2.4.3 項を参照してください。

RTA-OS のインストール中に ETAS ライセンスマネージャをインストールするかどうかが尋ねられるので、ここでインストールするか、または後で `<install_folder>\LicenseManager\LicensingStandaloneInstallation.exe` を実行することにより個別にインストールすることもできます。

ETAS ライセンスマネージャは、`C:\Program Files\Common Files\ETAS\Licensing` にインストールされます。

インストールを行うと、Windows のスタートメニューに ETAS ライセンスマネージャへのリンク（**すべてのプログラム → ETAS → License Management → ETAS License Manager**）が作成されます。

2.4.2 ライセンスの形態

ライセンスには以下の 3 種類の形態があり、それぞれ適用対象や運用方法が異なります。

マシンネームライセンス（ローカル）：　特定のPCを対象としたライセンスで、有効なライセンスキーが保存されたライセンスファイルをPCにインストールして使用します。対象となるPCにログインできるすべてのユーザーがRTA-OSを使用することができます。

マシンネームライセンスの発行にはPCのホストID（MACアドレス）が必要です。

ユーザーネームライセンス（サーバーベース）：　特定のユーザーを対象としたライセンスで、サーバー情報が保存されたライセンスファイルをPCにインストールして使用します。指定されたユーザーのみがネットワークドメインの任意のPC上でRTA-OSを使用することができます。

ユーザーネームライセンスの発行には、サーバーホストまたはサーバートライアドのホストID、およびユーザー名（Windowsユーザー名）が必要です。

コンカレントライセンス（サーバーベース）：　ユーザー数を指定して発行されるライセンスです。指定された最大ユーザー数を限度とし、複数のユーザーが決められた数のライセンスを共有し、任意のPC上でRTA-OSを使用することができます。この形態は「フローティングライセンス」とも呼ばれます。

コンカレントライセンスの発行には、サーバーホストまたはサーバートライアドのホストIDの情報が必要です。

2.4.3 コンカレントライセンスサーバーのインストール

コンカレントライセンスは、ベンダーデーモンと連携して稼働する FLEXnet ライセンスサーバーマネージャにより、複数のクライアント PC に割り当てられます。ETAS のベンダーデーモンの名前は ETAS.exe です。ETAS ライセンスマネージャをインストールすると、このベンダーデーモンのコピーがディスク上の以下の場所に格納されます。

```
C:\Program Files\Common Files\ETAS\Licensing\Utility
```

ライセンスサーバーが ETAS コンカレントライセンスを取り扱えるようにするには、ライセンスを使用する PC からアクセス可能なライセンスサーバーをセットアップする必要があります。ライセンスサーバーのセットアップには以下のソフトウェアを使用します。

- FLEXnet ライセンスサーバーマネージャ
- ETAS ベンダーデーモン（ETAS.exe）

また、ユーザーのコンカレントライセンスをライセンスサーバーにインストールすることも必要です。

一般的には、組織の IT 部門が 1 つの FLEXnet ライセンスサーバーマネージャを管理するようになっているので、社内の IT 部門の担当の方に、ETAS ベンダーデーモンとコンカレントライセンスのインストールを依頼してください。

FLEXnet ライセンスサーバーがまだ用意されていない場合は、最初にサーバーをインストールする必要があります。以下の場所に、FLEXnet ライセンスサーバーと ETAS ベンダーデーモンのコピー、および説明書（LicensingEndUserGuide.pdf）が格納されています。

```
C:\Program Files\Common Files\ETAS\Licensing\Utility
```

2.4.4 ライセンスのアクティベート（有効化）

RTA-OS を初めてインストールすると、ETAS ライセンスマネージャにより RTA-OS ソフトウェアは、ライセンスがなくても 7 日間だけ使用できる「グレースモード」に設定されます。その後は、グレースモードの期間が終了するまでにライセンスをアクティベート（有効化）してライセンスファイルを取得し、PC に登録する必要があります。

ホスト ID やユーザー名の確認

ライセンスをアクティベートするために必要な情報（PC のホスト ID やユーザー名）は、ETAS ライセンスマネージャから確認することができます。ETAS ライセンスマネージャのメニューから **Tools → Obtain License Info** を選択すると、”Obrain License Info”ダイアログボックス（図 2-2）が開き、カレントユーザーと PC のネットワークアダプタの情報が表示されます。

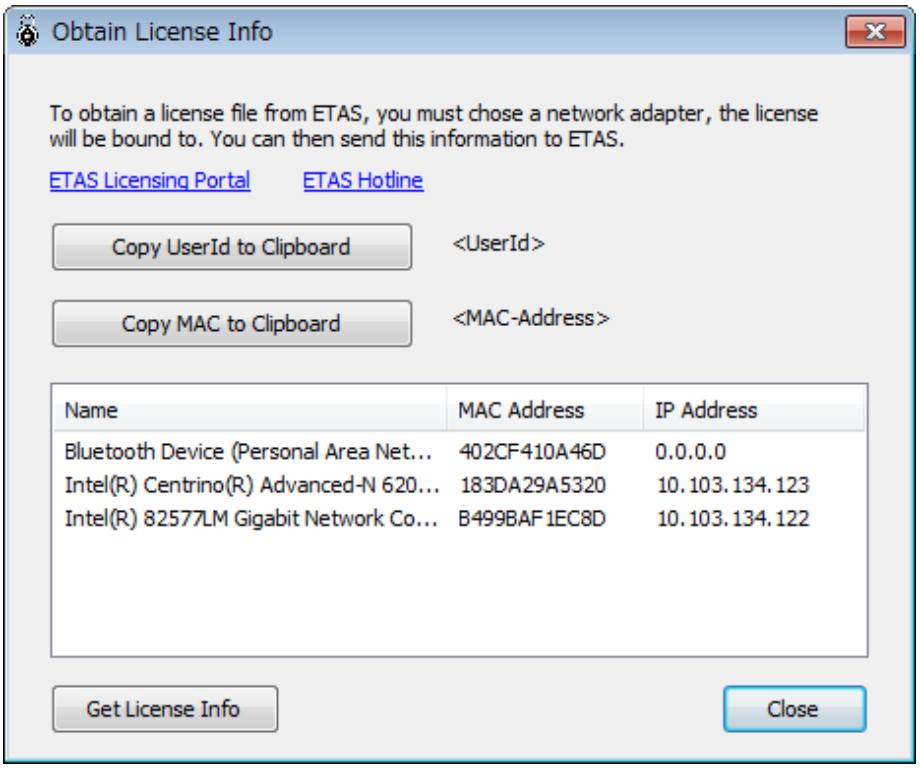

図 2-2　ライセンス情報を確認する

使用するネットワークアダプタを選択して **Copy MAC to Clipboard** ボタンをクリックすると、そのアダプタのホスト ID（MAC アドレス）がクリップボードにコピーされ、**Copy UserId to Clipboard** をクリックすると、カレントユーザーの Windows ユーザー名がクリップボードにコピーされます。

マシンネームライセンスの場合：

ETAS のライセンスポータルサイト（URL は製品に同梱されています）にログインし、PC のホスト ID を指定してライセンスをアクティベートします。ログインには、製品購入時に ETAS から発行された ID（アクティベーション ID またはエンタイトル ID）、または E メールアドレスとパスワードを使用できます。

ライセンスポータルの操作方法についての詳細は、ポータルサイトの ヘルプ というリンクをクリックしてヘルプドキュメントを参照してください。

ライセンスがアクティベートされると、ライセンスキーが含まれる`<name>.Lic` という名前のライセンスファイルが生成されるので、これを ETAS ライセンスマネージャから PC に登録します。登録の方法は 17 ページを参照してください。

コンカレントライセンスの場合：

FLEXnet サーバー上のライセンスを検索するための情報をライセンスファイルに記述します。このファイルは以下の手順で作成します。

1. コンカレントライセンスファイルのコピーを作成します。
2. コンカレントライセンスファイルのコピーを開き、SERVER から始まる行以外のすべての行を削除します。
3. USE_SERVER を含む新しい行を 1 行追加します。
4. 空白行を 1 行追加します。
5. このファイルを保存します。

作成されるファイルの内容は以下のようになります。

```
SERVER <server name> <MAC address> <TCP/IP Port>¶
USE_SERVER¶
¶
```

PC への登録方法はマシンネームライセンスの場合と同じです（17 ページ参照）。

2.4.5 ETAS ライセンスマネージャの使用

ETAS ライセンスマネージャでは、ライセンスステータスの確認やライセンスファイルの登録などが行えます。

有効なライセンスがない状態での RTA-OS 起動時

有効なライセンスがない状態で RTA-OS を実行しようとすると、自動的に ETAS ライセンスマネージャが起動し、有効なライセンスがない製品（または機能）のみが一覧表示されます（図 2-3 参照）。ライセンスステータスの列には「なし」と表示され、有効なライセンスが登録されるまでこれらのものは使用できません。

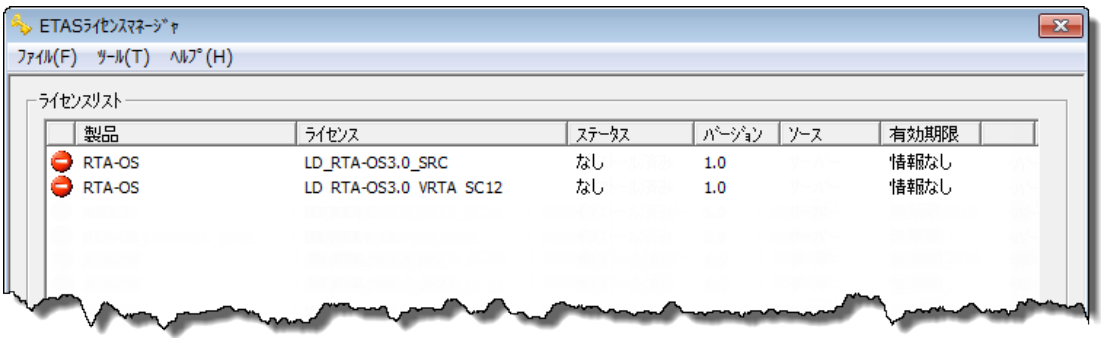

図 2-3　ライセンスが無効になっている RTA-OS モジュール

ライセンスファイルの登録は以下の手順で行います。

ライセンスファイルの登録

ライセンスファイルを PC に登録して製品のライセンスを有効にするには、ETAS ライセンスマネージャのメニューコマンド ファイル → ライセンスファイルの追加 を使用します。ライセンスファイルの取得方法については 16 ページを参照してください。

ライセンスファイルは、ETAS ライセンスマネージャにより以下のフォルダに保存されます。

```
C:\Documents and Settings\All Users\Application Data\ETAS\FlexNet
```

ライセンスステータス

有効なライセンスが登録されると、図 2-4 のようなライセンス情報（ステータス、バージョン、ソース、有効期限）が表示されます。

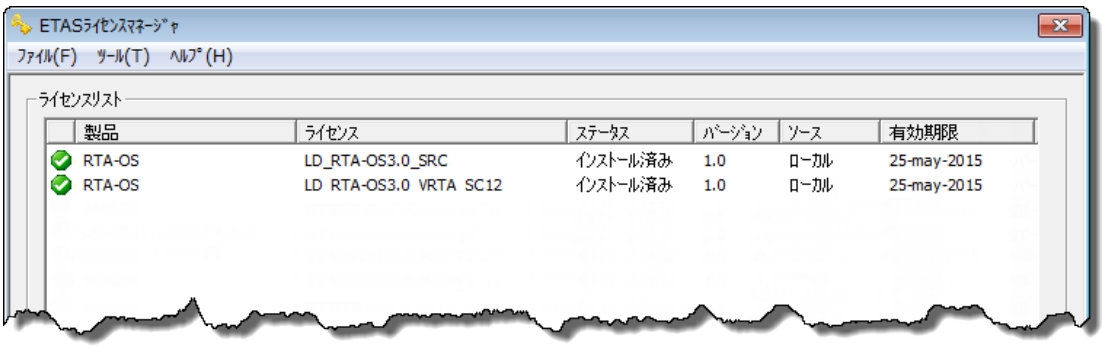

図 2-4　有効なライセンスが登録された RTA-OS モジュール

コンカレントライセンスの借用

コンカレントライセンスを使用している場合、PC をネットワークから切り離すとライセンスは無効になってしまいますが、ネットワークに接続できない環境で RTA-OS を使用する必要がある場合は、ライセンスを一時的にライセンスサーバーから借用することができます。

ライセンスを借用するには、ライセンスマネージャで以下の手順を実行してください。

1. 借用したいライセンスを右クリックします。
2. ショートカットメニューから ライセンスの借用 を選択します。
3. カレンダーが開くので、借用ライセンスの有効期限（借用期限）を指定します。
4. **OK** をクリックします。

借用ライセンスは、借用期間が終了すると自動的に失効します。必要に応じて借用期間の終了前に返却することもできます。ライセンスを返却するには、以下の手順を実行してください。

1. ネットワークに接続します。
2. 借用していたライセンスを右クリックします。
3. ショートカットメニューから **ライセンスの返却** を選択します。

2.4.6 ライセンスのトラブルシューティング

有効なライセンスが見つからない機能をユーザーが使用しようとすると、RTA-OS ツールはエラーを報告します。ライセンスを取得しているはずであるにもかかわらず RTA-OS ツールが動作しない場合は、以下の手順に従ってトラブルシューティングを行ってください。

ETAS ライセンスマネージャがライセンスを認識できていますか？

ETASライセンスマネージャには、ユーザーが使用しようとしている製品や機能に対して有効なライセンスキーが見えていなければなりません。

ETASライセンスマネージャが把握しているすべてのライセンスを調べるには、ETASライセンスマネージャを **rtaoscfg** のメニューコマンド **Help → License Manager...** を選択するか、またはWindowsのスタートメニューから **すべてのプログラム → ETAS → License Management → ETAS License Manager** を選択してライセンスマネージャを起動します。

このような方法で起動すると、ETASライセンスマネージャはPCにインストールされているすべてのETAS製品についてライセンスの状態をチェックし、表示します。有効なライセンスのステータスは”インストール済み”と表示され、無効なライセンスのステータスは”なし”と表示されます。

ライセンスは有効ですか？

有効期限が決められたライセンス（デモ用ライセンスなど）を使用している場合は、その有効期限が過ぎてしまっている可能性があります。ETASライセンスマネージャでライセンスの有効期限を調べ、期限切れになっていないかを確認してください。

有効期限が30日以内になると、ライセンスマネージャは黄色い三角形の警告マークを表示してその旨を通知します。図2-5の例を見ると、`LD_RTA-OS3.0_SRC`と`LD_RTA-OS3.0_VRTA_SC12`の有効期限が近付いていることがわかります。

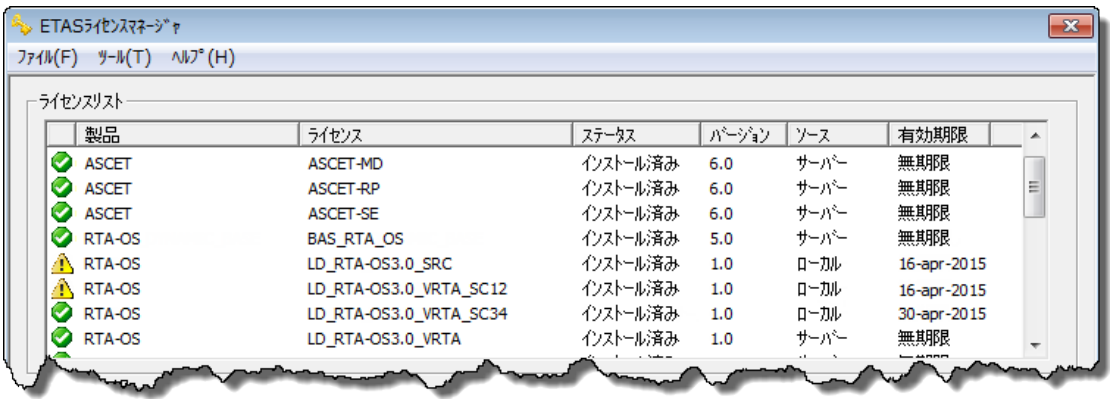

図 2-5　1 か月以内に無効となるライセンス

有効期限後もライセンスを使用する必要がある場合、またはすでに有効期限が切れてしまったライセンスを再度インストールする必要がある場合は、ETAS営業窓口までお問い合わせください。

ライセンスファイルのホスト ID と PC のホスト ID が一致していますか？

マシンネームライセンスは、特定のホストID（MACアドレス）に対してのみ有効です。ご使用のPCのMACアドレスを調べるには、ETASライセンスマネージャを使用する（**T ツール → ライセンス情報の取得**）か、またはWindowsコマンドプロンプトからMicrosoftプログラム **ipconfig /all** を実行してください。

ライセンスファイルに保存されたMACアドレスを確認するには、テキストエディタでライセンスファイルを開き、HOSTIDの値を確認してください。

ライセンスファイル内のHOSTIDの値がPCのMACアドレスと一致していないと、そのライセンスを有効にすることはできません。ご不明の点はETASの営業窓口までお問い合わせください。

ネットワークアダプタは有効になっていますか？

ノートPCを使用していて、ネットワークから切断したときRTA-OSが停止してしまう場合は、PCの省電力設定を調べて、ネットワークに接続されていないときにネットワークアダプタが無効になるように設定されていないかを確認してください。

この確認はデバイスマネージャから行えます。デバイスマネージャを開くにはWindowsのコントロールパネルから **システム → デバイスマネージャ** を選択します。デバイス一覧の中から該当するネットワークアダプタを選択して右クリックし、ショートカットメニューから **プロパティ** を選択してプロパティダイアログボックスを開き、**詳細設定** タブと **電源の管理** タブで省電力に関する設定内容を確認します。

FlexNet ライセンスサーバーにアクセスできていますか？

FlexNetライセンスサーバーから提供されるサーバーベースライセンスを使用している場合、そのライセンスサーバーにアクセスできないと、ETASライセンスマネージャ上にはそのライセンスステータスが"なし"と表示されます。

社内のIT部門に問い合わせて、サーバーが正しく機能しているかどうかを確認してください。

上記の項目がすべて解決されてもまだライセンスが有効になりませんか？

以上の事柄を確認してもまだライセンスが有効にならない場合は、ETASのサポート窓口までお問い合わせください。お問合せ先は6.1項に掲載されています。お問い合わせの際にはライセンスファイルの内容と格納場所、およびPCのイーサネットMACアドレスをお知らせください。

2.5 インストール状態の検証

RTA-OS のインストールが終わり、有効なライセンスをインストールできたところで、作業がうまく進んでいることを確認しましょう。以下のコマンドラインを使用して RTA-OS コードジェネレータを実行することにより、PC ツールのインストールが成功したことを検証できます。

```
rtaosgen --target:?
```

インストールが成功していれば、ツールが実行され、使用可能なターゲットとそのバージョンやバリアントのリストが以下の例のように出力されます。

```
rtaosgen --target:?
RTA-OS Code Generator
Version x.x.x.xxxx, Copyright © ...
Available targets:
   PPCe200GHS_x.x.x [MPC5516z1,MPC5516z0,...]
   TriCoreHighTec_x.x.x [TC1732,TC1736,...]
   VRTA_x.x.x [MinGW,VS2005,VS2008,VS2010]

Errors: 0, Warnings: 0.
```

ターゲットのポートプラグインがデフォルト以外のロケーションにインストールされている場合は、`--target_include` 引数を使用する必要があります。上の例のコマンドラインの代わりに以下のコマンドラインを使用してください。

```
rtaosgen --target_include:<tgtDir> --target:?
```

2.6 ネットワークドライブからの RTA-OS の実行

統合のためのアドバイス 2.8： RTA-OS ツールは Windows .NET アプリケーションです。Windowsセキュリティポリシーにより、デフォルトでは.NETアセンブリ（アプリケーション）をネットワークドライブから実行することはできません。RTA-OS ツールをネットワークドライブから実行するには、特別なコンフィギュレーション設定が必要になります。

RTA-OS をネットワークから実行できるようにするには、以下のいずれかを用いて Windows セキュリティポリシーを変更します。

- Windows コントロールパネル
- Microsoft コマンドラインツール **CasPol.exe**。これは、Microsoft .NET Framework Control Panel とともに配布されます。

コントロールパネルを使用する方法はシンプルですが、コマンドラインツールの方がより詳細な設定が可能です。作業環境に適したセキュリティレベルについては、社内の IT 部門にご相談ください。

2.6.1 コントロールパネルによる設定

コントロールパネルからセキュリティポリシーを変更するには、**コントロールパネル → 管理ツール → Microsoft .NET Framework <x.x> 構成 → ランタイムセキュリティポリシー → ゾーンセキュリティの変更** の順にナビゲートし、“ローカルイントラネット”の値を“Full”にしてください。

2.6.2 コマンドラインツールによる設定

コマンドラインツール **CasPol.exe** 使用して、アプリケーションを実行するネットワークドライブのセキュリティポリシーを変更することができます。このプログラムは `C:\WINDOWS\Microsoft.NET\Framework\vn.n.nnnnn\CasPol.exe` に格納されています。変更を行うには以下の手順を実行します。

1. **CasPol.exe -pp off** を実行して、ポリシー変更プロンプトをオフにします。
2. **CasPol.exe -m -ag 1.2 –url "file:<share_name>/*" FullTrust** を使用してネットワークドライブのポリシーを変更します。各引数の意味は下の表のとおりです。

引数	説明
`-m`	マシンレベルで動作します。
`-ag 1.2`	コードグループをグループ 1.2 に追加します。デフォルトポリシーでは、Group 1.2 は LocalIntranet（ローカルイントラネット）グループなので、ユーザーが作成する新規コードグループはファイルがイントラネットから得られた場合にかぎりチェックされます。
`-url:...`	指定された URL 内の任意のファイルとマッチします。`<share_name>` を実際のドライブ名（`\networkdrive\ETAS\RTA-OS` など）で置き換えて使用してください。
`FullTrust`	コードグループにマッチするアセンブリを許可するように設定されるパーミッション。この場合は FullTrust。

3. **CasPol.exe -pp on** を使用して、ポリシー変更プロンプトをオンに戻します。

3 RTA-OS を使用するアプリケーションの開発

ユーザーアプリケーションに RTA-OS を組み込む作業は、おおまかに以下の 3 ステップで行います。

1. OS の機能のうち、使用するものについて設定します。
2. カスタマイズされた RTA-OS カーネルライブラリを生成します。
3. OS をユーザーのアプリケーション内で使用します。

これらのステップについて、以下に詳しく説明します。これらのステップを省略してすぐに RTA-OS を使用してみたい場合は、本章をスキップして第 4 章に進んでください。第 4 章では、各ポートプラグインとともに提供されているサンプルアプリケーションをビルドして実行する方法について説明しています。

3.1 コンフィギュレーション

RTA-OS のコンフィギュレーション設定は静的に行います。つまり、実行プログラムをビルドする前にコンフィギュレーションの設定（ユーザーアプリケーションに必要なタスクと割り込み、クリティカルセクション、同期化ポイント、カウンタなどの定義）を行っておく必要があります。

コンフィギュレーションデータは、AUTOSAR ECU Configuration Definition 規格に準拠する XML ファイルに格納されます。

RTA-OS にはグラフィカルコンフィギュレーションエディタ **rtaoscfg**（図 3-1 参照）が含まれていて、これを使用して RTA-OS コンフィギュレーションファイルを作成することができます。**rtaoscfg** では AUTOSAR XML ファイルを入力として読み取り、コンフィギュレーション内の OS 固有の部分を編集することができます。OS 固有のコンフィギュレーションと OS 固有でないコンフィギュレーションの両方が入力ファイルに含まれている場合は、OS コンフィギュレーションだけが変更されます。**rtaoscfg** の使用方法は、本製品のユーザーズガイドに詳しく記載されています。

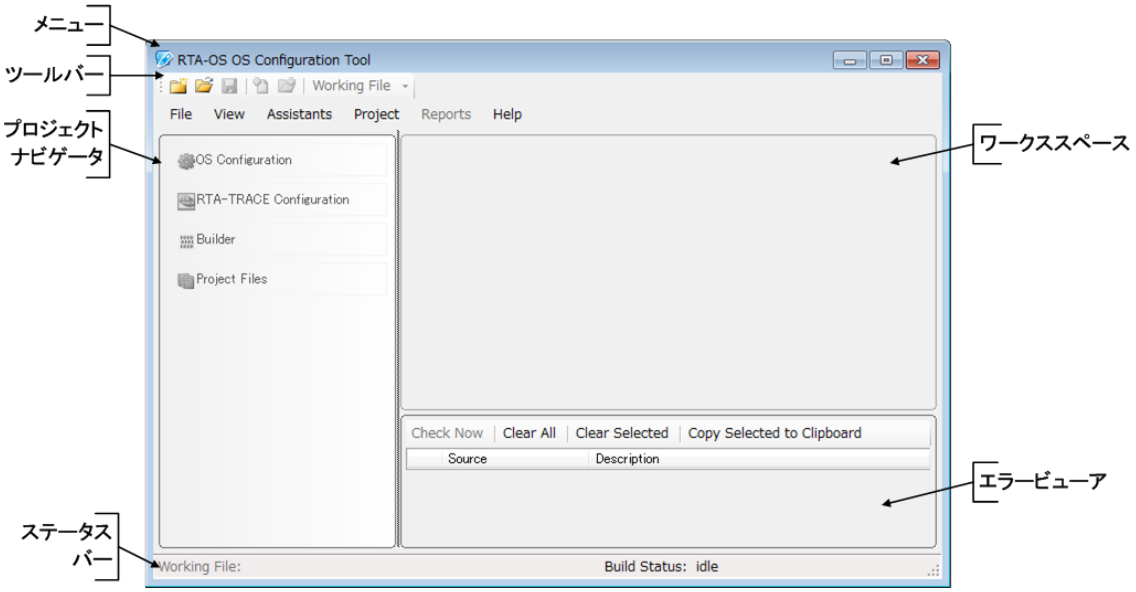

図 3-1　rtaoscfg コンフィギュレーションツール

3.1.1 OSEK OIL ファイルをコンフィギュレーションに使用する

RTA-OS に含まれる OIL インポートアシスタント（OIL Import Assistant）というツールを使用すると、既存の RTA-OSEK アプリケーションからの移行を円滑に行うことができます。GUI の **Assistants** メニューから図 3-2 のようにしてコマンドを選択すると、OIL インポートアシスタントのダイアログボックス（図 3-3）が開きます。

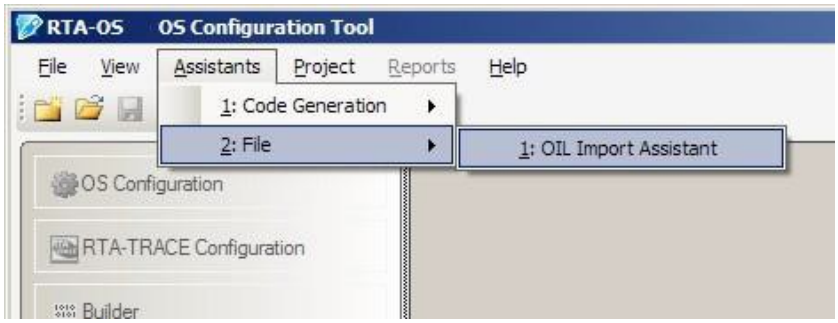

図 3-2　OIL インポートアシスタントを開くためのメニューコマンド

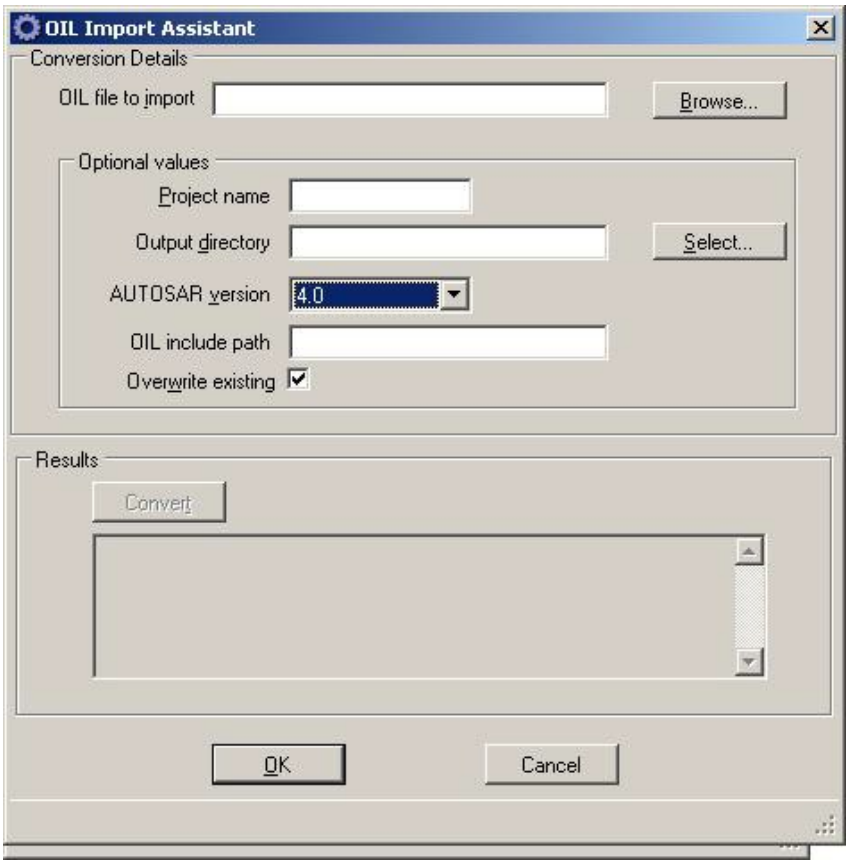

図 3-3　OIL インポートアシスタント

OIL インポートアシスタントは、従来の RTA-OSEK アプリケーションの OIL ファイルを AUTOSAR XML に変換します。

ダイアログボックスに必ず入力しなければならない項目は、入力用 OIL ファイルだけです。以下の項目は必要に応じて設定します。

Output directory（出力ディレクトリ）

出力ディレクトリを指定しない場合、出力ファイルは入力用OILファイルと同じディレクトリに生成されます。

Project name（プロジェクト名）

デフォルトでは、生成されるファイルの名前は入力ファイル名から導き出されます。他の名前を使用したい場合はここに名前を入力します。

AUTOSAR version（AUTOSAR のバージョン）

特定バージョンのXMLに準拠する必要がある場合は、ここでAUTOSAR OSのバージョンを指定します。

OIL include path（OIL インクルードパス）

入力用OILファイルは、他のディレクトリに格納されているOILファイルを参照するインクルード文を使用している場合があります。そのような場合は、被参照ファイルを含むディレクトリのパスをここに設定します。複数のディレクトリを指定するには、各ディレクトリの間をセミコロンで区切ってください。

OIL ファイルを変換するには、**Convert** ボタンをクリックします。変換結果は"Results"エリアに表示されます。その後、**OK** ボタンをクリックすると変換済みファイルが GUI にロードされ、**Cancel** をクリックするとインポート処理は中止されます。

3.2 ライブラリの生成

アプリケーション内で RTA-OS を使用するには、RTA-OS のカーネルライブラリとヘッダファイルを生成する必要があります。カーネルライブラリの生成には、コマンドラインツール **rtaosgen** を使用します。このツールは以下の処理を順に行います。

1. XML コンフィギュレーションを分析し、使用される機能だけが含まれるように RTA-OS カーネルを最適化します。これにより RTA-OS のサイズが最小化され、効率的に実行できるようになります。
2. ターゲットプラグインにより提供される情報を使用して、最適化されたカーネルをターゲット用にカスタマイズします。
3. アプリケーションが使用するものと同じサードパーティ製ツールチェーンを使用して、カーネルライブラリをビルドします。これにより RTA-OS カーネルライブラリとアプリケーションコードの間でツールチェーンの適合性が保証されます。

3.2.1 ツールチェーンの準備

カーネルライブラリをビルドするには、**rtaosgen** からターゲットツールチェーンにアクセスする必要があります。**rtaosgen** は、ターゲット用ツール（コンパイラ、アセンブラ、リンカ、ライブラリアン）の実行方法や、渡すべき引数を把握しているので、ユーザーが注意しなければならないのは以下の 2 点だけです。

1. ツールチェーンのパスをユーザーのパス（PATH 環境変数）に割り当てておく必要があります。またその代りに--env というコマンドライン引数を使用することもできます。
2. ツールチェーンは、RTA-OS に適合している必要があります。

環境変数 PATH の確認と設定は、Windows コマンドプロンプトから簡単に行えます。コマンドプロンプトに echo %PATH%と入力して実行すると、現在定義されているパスの一覧が開くので、ここにコンパイラの実行ファイルのパスが含まれていない場合は、以下のように入力してパスを追加します。

```
set PATH=%PATH%;<PathToCompilerExecutable>
```

ツールチェーンが RTA-OS に適合しているかどうかを調べるには、各ポート用の『Target/Compiler Port Guide』を参照してください。

3.2.2 rtaosgen の実行

rtaosgen は、Windows の実行ファイルを呼び出すことのできるすべての場所（Windows コマンドプロンプト、バッチファイル、makefile、Ant スクリプトなど）から実行することができます。また、グラフィカル環境 **rtaoscfg** のビルダペインからも実行できます。

3.2.3 ライブラリのビルド

RTA-OS は、AUTOSAR の基本ソフトウェアモジュール（Basic Software Module）であるため、標準の AUTOSAR ヘッダファイルにアクセスします。そのため、**rtaosgen** を呼び出す際には AUTOSAR 標準ヘッダファイルへのパスを指定する必要があります。たとえば、Hello World という RTA-OS ターゲット用サンプルアプリケーションのためのライブラリをビルトするには、以下のように入力します。

```
rtaosgen --include:<PathToAutosarHeaderFiles> HelloWorld.xml
```

ユーザーが AUTOSAR ヘッダファイルにアクセスできない場合（たとえば、AUTOSAR システムとは無関係のアプリケーション開発のために RTA-OS を使用している場合など）は、**rtaosgen** で AUTOSAR ヘッダファイルを自動生成することができます。その際には、生成されるヘッダの格納場所を RTA-OS に対して指示する必要があります。一般的な格納場所は Samples\Includes で、コマンドは以下のようになります。

```
rtaosgen --samples:[Includes] --include:Samples\Includes
     HelloWorld.xml
```

ビルド処理実行中に **rtaosgen** は、プレビルド処理とポストビルド処理を実行する 2 つのファイル（<projectname>_prebuild.bat および<projectname>_postbuild.bat）が存在するかどうかを調べます。これらのファイルがプロジェクトディレクトリ（以下の RTA_PROJECT_BASE を参照）に存在する場合は、ビルドの前後に実行されます。ユーザーが **rtaosgen** をどのディレクトリから起動した場合でも、これらのファイルはプロジェクトディレクトリから実行されます。一般的なポストビルド処理では、生成されたファイルの名前を変更したり、生成されたファイルをソースコントロールにコミットしたりします。

これらの.bat ファイル内では、**rtaosgen** が提供する以下の環境変数が使用できます。

- RTA_PROJECT_BASE：プロジェクトディレクトリ。通常、.rtaos ファイルはここに格納されます。
- RTA_PROJECT_NAME：プロジェクト名。通常は、.rtaos ファイルの名前から拡張子を除いたものです。

- RTA_WORKING_FOLDER： ビルド処理を実行するディレクトリ。通常、ユーザーはこのディレクトリから **rtaosgen** を実行します。

3.2.4 ビルドメッセージ

rtaosgen は実行中に以下の 4 種類のメッセージを発行します。

Information（情報）

コンフィギュレーションに関する有益な事柄（設定済みタスクの数など）を通知します。

Warning（警告）

コンフィギュレーションの挙動がユーザーが期待するものと異なるものになる可能性がある旨を警告します。

Error（エラー）

コンフィギュレーション内に何か間違いがあることを知らせます。**rtaosgen**は適切な時点においてコンフィギュレーション処理を停止します。出力ファイルは生成されません。

Fatal（フェイタル）

コンフィギュレーションまたは**rtaosgen**のいずれかに根本的な間違いがあることを知らせます。**rtaosgen**は直ちに停止するので、このタイプのメッセージが2つ以上連続して発生することはありません。

3.2.5 生成されるファイル

エラーもフェイタルメッセージも発行せずに **rtaosgen** の実行が完了すると、以下のファイルが生成されます。

ファイル名	内容
Os.h	OS 用のメインインクルードファイル。
Os_Cfg.h	ユーザーが設定したオブジェクトの宣言。Os.h にインクルードされます。
Os_MemMap.h	AUTOSAR メモリマッピングコンフィギュレーション。RTA-OS によりシステムワイドの MemMap.h ファイルにマージされます。
RTAOS.<lib>	ユーザーアプリケーションのための RTA-OS ライブラリ。拡張子<lib>はターゲットにより異なります。
RTAOS.<lib>.sig	ユーザーアプリケーション用ライブラリのためのシグネチャファイル。これは、コンフィギュレーションが変更された場合に、カーネルライブラリのどの部分をリビルドする必要があるかを把握するために **rtaosgen** が使用するファイルです。拡張子<lib>はターゲットにより異なります。
<projectname>.log	ビルド処理実行中にツールとコンパイラが画面に出力したテキストのコピーが格納されたログファイルです。

ポートによっては、これ以外のファイルも生成される可能性があります。詳細は各ポートの『Target/Compiler Port Guide』を参照してください。

3.3 統合

3.3.1 ユーザーのソースコード内で OS にアクセスする

RTA-OS の呼び出しを行うすべての C ソースファイルは、`Os.h` をインクルードする必要があります。このヘッダファイルは多重インクルードから保護されています。

RTA-OS を使用するにあたり、ソースコードの構成に対する要件はありません。1 つのソースファイルに 2 つ以上のタスクや割り込みを記述することも可能です。

3.3.2 タスクと ISR の実装

タスク

コンフィギュレーション設定時にユーザーが宣言する各タスクについては、ユーザーがそのタスクを実装する必要があり、各タスクは `TASK(x)`マクロでマークします。一般的にタスクは以下のような構造になります。

```
#include <Os.h>
TASK(MyTask){
  /* Do something */
  TerminateTask();
}
```

カテゴリ 2 の ISR

カテゴリ 2 の ISR も、ユーザーが実装する必要があります。ISR は以下のように `ISR(x)`マクロでマークします。

```
#include <Os.h>
ISR(MyISR){
  /* Do something */
}
```

統合のためのアドバイス 3.1： カテゴリ 2 の ISR ハンドラには、割り込みコールからのリターンは必要ありません。リターン処理は RTA-OS が自動的に行います。ターゲットハードウェア上での割り込みソースの挙動によっては、ユーザーが割り込み保留フラグをクリアしなければならない場合があります。詳細については、チップベンダーが提供するハードウェアマニュアルを参照してください。

カテゴリ 1 の ISR

カテゴリ 1 の ISR も、ユーザーが実装する必要があります。各コンパイラは、C 関数を割り込み処理としてマークするための特別な表記を使用し、RTA-OS は、コンパイラ用の正しい命令に展開されるマクロを提供します。カテゴリ 1 のハンドラは以下のようになります。

```
#include <Os.h>
CAT1_ISR(MyCat1ISR) {
  /* Do something */
}
```

OS の始動

RTA-OS はハードウェアを制御しないので、リセットから抜け出した後、ユーザーアプリケーションは、オペレーティングシステムを使用しない組み込みアプリケーションの場合と同じように、必要なすべての初期化を OS に妨げられることなく実行することができます。OS の始動は、ハードウェアの初期化が完了した後に StartOS()を呼び出して行います。この呼び出しは一般的に main()関数内で行います。

RTA-OS は OS_MAIN()マクロを提供しています。このマクロを使用して、main()関数を以下の例のような一般的な形でマークすることができます [2]。

```
#include <Os.h>
OS_MAIN(){
  /* Initialize target hardware */
  ...
  /* Do any mode management, pre-OS functions etc. */
  ...
  /* Use RTA-OS to initialize interrupts */
  Os_InitializeVectorTable();
  StartOS();
  /* Call does not return so you never reach here */
}
```

3.3.3 RTA-OS カーネルとのインタラクション

RTA-OS へのアクセスはカーネル API 関数により行います。各コールの詳しい情報はリファレンスガイドに掲載されています。

3.3.4 コンパイルとリンク

アプリケーションをコンパイルするときには、Os.h と Os_Cfg.h がコンパイラのインクルードパス上に存在している必要があり、アプリケーションをリンクするときには、RTAOS.<lib>がユーザーのリンカライブラリパス上に存在している必要があります。

また、ユーザーが使用するすべてのツールオプションが、RTA-OS ライブラリに適合している必要があります。必須のツールチェーンオプションと使用できないツールチェーンオプションについては、各ポートの『Target/Compiler Port Guide』に記載されています。

[2] ほとんどのポートでは、OS_MAINはvoid main(void)に展開されます。しかし、一部の組み込みコンパイラはmainプログラムがvoidを返すことを許していないので、異なる展開が必要になります。

3.3.5 よくある問題

コンパイラが RTA-OS 関連のエラーを発行した場合は、以下の条件が満たされているかどうかを確認してください。

- `Os.h` と `Os_Cfg.h` がコンパイラのインクルードパス上に存在するかどうか
- コンフィギュレーションに変更を施した後、カーネルライブラリが再ビルドされているかどうか
- `RTAOS.<lib>`がリンカのライブラリパス上に存在するかどうか

3.3.6 ターゲットへのダウンロード

一般的なリンカの出力は、a.out、COFF、ELF、IEEE695 などのバイナリファイルです。これらのファイルは、デバッガやインサーキットエミュレータ、インサーキットプログラマなどで読み取ることができますが、場合によっては、この出力をシリアルリンクでターゲットに転送できるフォーマット（S レコードや Intel HEX など）に変換する必要があります。この変換を行うツールは、一般的に開発環境に含まれています。アプリケーションを不揮発性メモリにプログラムする方法については、ターゲットプラットフォームや開発ツールチェーンの資料を参照してください。

4 サンプルアプリケーション

各ポートプラグインには、すぐに実行して RTA-OS の主要機能をデモンストレーションできるサンプルアプリケーションが付属しています。これらのサンプルアプリケーションを使用すると、OS のコンフィギュレーションを設定して、ライブラリを生成し、アプリケーションコードをライブラリとリンクする一連の手順を試すことができます。

各ポートプラグインには以下のサンプルアプリケーションが付属しています[3]。

アプリケーション	内容
Hello World	基本タスク、割り込み、カウンタ、アラーム
Clive Devices	基本タスク、割り込み、カウンタ、アラーム、スケジュールテーブル
Pizza Pronto	拡張タスク、割り込み、アラーム、カウンタ

統合のためのアドバイス 4.1： RTA-OS がユーザーのコンパイラツールチェーンを使用して OS カーネルを生成できること、およびそのカーネルとともにシンプルなアプリケーションが稼働できることを検証するため、少なくとも Hello World アプリケーションをビルドして実行しておくことをお勧めします。

4.1 サンプルアプリケーションの準備

サンプルアプリケーションと **rtaosgen** は、ユーザーのコンパイラツールチェーンを使用する必要があり、コンパイラツールチェーンが Windows パス上に存在することを想定しています。 所定のパス上にない場合は、以下のように Windows コマンドプロンプトを使用して、ツールのパスをユーザーパスに追加することができます。

```
set PATH=<PathToCompilerExecutable>;%PATH%
```

また、RTA-OS ツールもユーザーパスに追加しておくことをお勧めします。

```
set PATH=<PathToRTAOS>\bin;%PATH%
```

サンプルアプリケーションを使用可能な状態にするには、ある特定のターゲットおよびターゲットバリアントを対象としてアプリケーションを展開する必要があります。展開は、コマンドライン、または **rtaoscfg** に含まれる専用ツールを使用して行います。

4.1.1 コマンドラインからサンプルアプリケーションを展開する

コマンドラインからサンプルアプリケーションを展開するには、以下の手順を実行します。

1. サンプルアプリケーションのビルドに使用する新しい作業フォルダを作成します。
2. 作成した新規フォルダ内で Windows コマンドプロンプトを開きます。
3. 以下のコマンドを実行します。

```
rtaosgen --samples:[Applications]
  --target:[<variant>]<target>
```

[3] ポートによってはこれ以外のアプリケーションも提供されています。

2.5 項で説明されている以下のコマンドを実行すると、インストールされているターゲットとバリアントを確認することができます。

```
rtaosgen --target:?
```

引数<variant>は必要に応じて指定します。これを省略すると、各サンプルアプリケーションはデフォルトターゲット用に展開されます。

上記のコマンドにより、Samples\Applications という名前の新しいサブフォルダが作成されます。このフォルダの中には、選択されたターゲットとバリアントのデフォルト設定を使用する各サンプルアプリケーション用サブフォルダが作成されます。クリーンなサンプルアプリケーションを作成する必要がある場合は、新しい作業フォルダでこの処理を再度実行してください。

たとえば、VS2008 バリアントを使用する VTRA ターゲット用のサンプルアプリケーションを展開するには、以下のコマンドを実行してください。

```
rtaosgen --samples:[Applications] --target:[VS2008]VRTA
```

各サンプルアプリケーションを個別に展開することもできます。たとえば以下のコマンドを実行すると、VRTA の MinGW バリアント用の PizzaPronto アプリケーションが展開されます。

```
rtaosgen --samples:[Applications\PizzaPronto]
  --target:[MinGW]VRTA
```

4.1.2 GUI からサンプルアプリケーションを展開する

サンプルアプリケーションは、RTA-OS GUI に含まれるサンプルアプリケーション生成アシスタント（Example Application Generator Assistant）を使用して展開することもできます。

GUI の **Assistants** メニューから図 4-1 のようにしてコマンド（**Code Generation → Example Application Generator**）を選択すると、インストールされているターゲットの情報が検索され、図 4-2 の”Example Application Generator”ダイアログボックスが開きます。

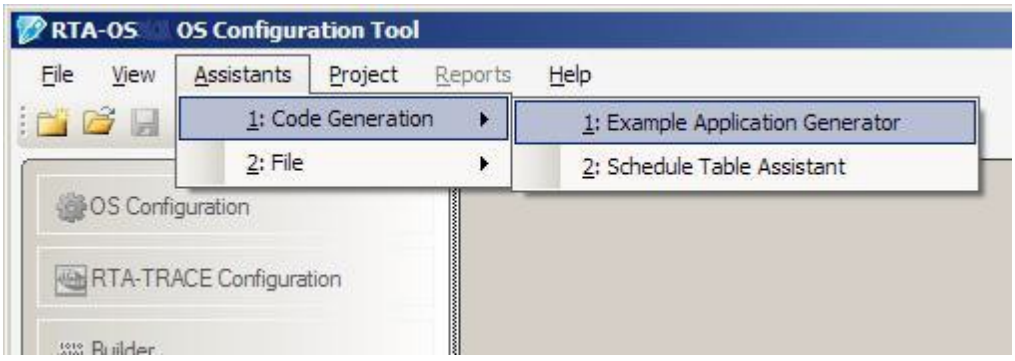

図 4-1　RTA-OS　Assistants メニュー

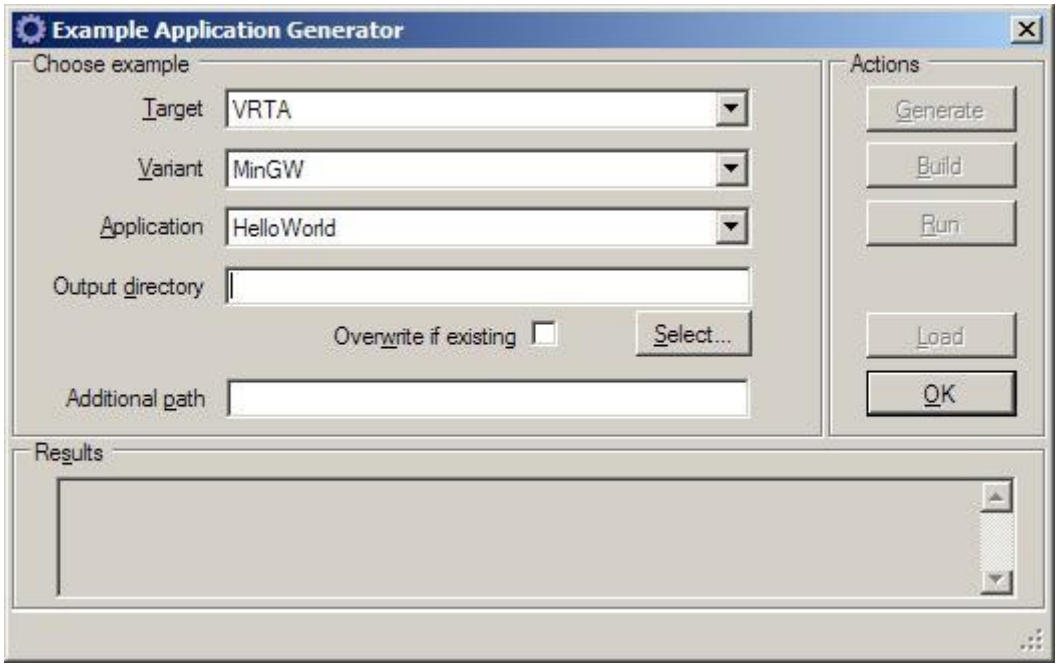

図 4-2　RTA-OS のサンプルアプリケーション生成アシスタント

ここでターゲット、バリアント、サンプルアプリケーションを選択し、出力ディレクトリを指定すると、**Generate** ボタンが有効になり、アプリケーションを展開できるようになります。アプリケーションを展開した後は、それをビルドして実行したり、GUI にロードしたりすることができ、そのままディスクに保存しておくこともできます。

アプリケーションをビルドするときにツールチェーンが見つからない場合は、このダイアログボックスの Additional Path フィールドにパスを追加することができます。

4.2　サンプルアプリケーションの構造

たとえば `C:\GettingStarted` という名前の作業フォルダを作成して、そこに VRTA ポート用のサンプルアプリケーションを展開すると、図 4-3 のようなフォルダ構造が作成されます。実際に展開されるサンプルアプリケーションは、ポートに応じて異なる場合があります。

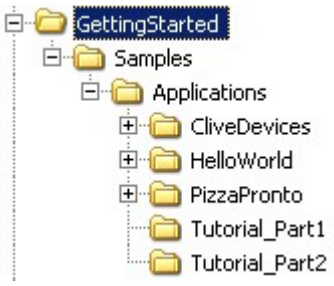

図 4-3　サンプルアプリケーションのフォルダ構造

サンプルアプリケーションは各ターゲットに共通なデフォルトオプションを使用しますが、必要に応じてターゲット固有のオプションを使用するサンプルアプリケーションを生成することもできます。詳細は 4.7 項で説明します。

サンプルアプリケーションはそのサブフォルダ内ですべて自己完結しています。つまり、ビルドと実行に必要なすべてのファイルが各アプリケーションのサブフォルダ内に含まれています。以降の項で、標準的なファイルについて簡単に説明します。

4.2.1 コンフィギュレーション

一般的に、RTA-OS コンフィギュレーションは 以下のファイルに含まれています。

ファイル名	説明
`<ApplicationName>.rtaos`	コンフィギュレーションが定義された AUTOSAR OS XML ファイルを参照する RTA-OS プロジェクトファイル
`<ApplicationName>.xml`	ターゲットに依存しないコンフィギュレーション。このファイルにはサンプルアプリケーションが使用するすべてのタスク、割り込み、アラームなどが定義されます。
`<ApplicationName>_Target.xml`	ターゲット固有のコンフィギュレーション。このファイルにはすべての割り込みの優先度とベクタ、ターゲット固有のコンフィギュレーションオプション、スタックサイズなど、ターゲットに固有なコンフィギュレーションが定義されます。

rtaoscfg で`<ApplicationName>.rtaos` を開くと、RTA-OS コンフィギュレーションの設定内容を見ることができます。

4.2.2 アプリケーションコード

`*.h` ファイルと`*.c` ファイルには、アプリケーションのソースコードが格納されています。ソースコードはターゲット固有の部分とターゲットに依存しない部分とに分かれます。ターゲット固有の部分については 4.2.3 項で説明します。

4.2.3 ターゲットサポート

RTA-OS はマイクロコントローラペリフェラル（タイマ、カウンタ、I/O ポートなど）の制御を一切行いません。これらのデバイスを使用して RTA-OS 内のスケジューリングを行うには、ユーザーがターゲット上のペリフェラルを適切に設定する必要があります。

統合のためのアドバイス 4.2： ターゲットマイクロコントローラの設定と使用は、ユーザーの責任において行ってください。

ただし、サンプルアプリケーションにはマイクロコントローラを初期化するために最小限のハードウェアコンフィギュレーションが必要であるため、ターゲット固有のソースファイル内にその処理が提供されています。たとえば、標準的なサンプルアプリケーションではスケジューリング駆動のために 1ms タイマ割り込みを使用しています。

ターゲットの設定についての説明や、設定を行うためのソースコードは、以下のファイルに提供されています。

ファイル名	説明
README.txt	サンプルアプリケーションに関する低レベル情報(ターゲットのリファレンスプラットフォーム、使用するペリフェラル、メモリ配置、アプリケーションのダウンロードに使用できるデバッガなど）についての詳細な説明
Target.c	ハードウェアを初期化し､割り込みソースをプログラムするためのターゲット用ソースコード
Target.h	ターゲット用マクロ（保留状態の割り込みをクリアするマクロなど）

ポートによってはこれ以外のファイルも提供される場合があります。

サンプルアプリケーションが使用しているものと異なるターゲットハードウェアやコンパイラを使用する場合は、ターゲット用ファイルの修正が必要となる場合があります。

ビルドと実行

サンプルアプリケーションとともに提供される以下のファイルを利用して、アプリケーションをビルドして実行することができます。ターゲットハードウェアを使用する場合はサンプルをターゲットハードウェアにダウンロードして実行します。

ファイル名	説明
build.bat	RTA-OS カーネルライブラリを生成して、サンプルをコンパイルし、さらにサンプルコードをライブラリとリンクしてダウンロード可能な実行イメージを生成するための基本スクリプト
run.bat	サンプルアプリケーションの実行方法を示すサンプルスクリプト

アプリケーションをビルドするには、Windows コマンドプロンプトを開いて以下のいずれかを実行します。

- `build.bat` ― 組み込みターゲットを使用する場合
- `build.bat <variant>` ― VRTA を使用する場合。ここで `variant` はサンプルアプリケーションのビルドに使用するターゲットバリアント（つまりコンパイラ）の名前です。

`build.bat` が正常に終了すると、`<ApplicationName>.<target extension>`という実行プログラムがフォルダ内に作成されます。

アプリケーションのダウンロードや実行には `run.bat` スクリプトを使用します。

以降の項では、各サンプルアプリケーションが実行する処理、およびそれらが正しく機能していることを検証するための方法について説明します。

4.3 Hello World

サンプルアプリケーション Hello World は、一連のビルド処理がすべて正しく機能するかどうかを検証するためのものです。これにより、RTA-OS がユーザーのツールチェーンでカーネルライブラリをビルドできることを確認でき、さらに、ライブラリを使用した必要最小限の規模のサンプルアプリケーションを実行することによりカーネルが正しく機能することを実証することができます。

このサンプルアプリケーション用に生成されるソースコードは、コアが 2 個以上あるターゲットバリアント用に作成された場合はマルチコア処理を実行します。

4.3.1 Hello World アプリケーションが実行する処理

シングルコアターゲットにおいて Hello World を実行すると、`HighPriority` および `LowPriority` という 2 つのタスクの間でプリエンプションが行われるようすを確認することができます。実行に要する時間はどちらのタスクも 2ms ですが、タスク `HighPriority` の方が優先度が高く、50ms の周期で実行され、タスク `LowPriority` は 25ms の周期で実行されます。

マルチコアターゲットにおいては、2 つのタスクとその周期はシングルコアターゲットの場合と同じですが、タスク `HighPriority` はコア 1、他のすべての OS オブジェクトはコア 0 で実行されるように設定されます。

タスクの周期的実行は、2 つのアラーム（`Alarm50` と `Alarm25`）を使用して実現されます。これらのアラームは `MillisecondCounter` というカウンタにアタッチされていて、このカウンタは、`MillisecondInterruptHandler` という割り込みサービスルーチン（ISR）によって処理される 1ms タイマ割り込みによりチックされます。どちらのアラームも自動的に始動されますが、`Alarm50` は `Alarm25` より 1ms 遅れて始動されます。

アプリケーションが実行されると、`Alarm25` は 1ms 後に満了してタスク `LowPriority` を起動します。タスク `LowPriority` は、実行中は `IO_PIN1` を HIGH に保ち、1ms にわたって実行された後にタスク `HighPriority` にプリエンプトされます。

タスク `HighPriority` は、2ms にわたって実行された後にターミネートします。処理開始時に `IO_PIN1` の状態を保存してからそれを LOW にし、`IO_PIN2` を HIGH にします。そしてターミネートする直前に `IO_PIN2` を LOW にして、`IO_PIN1` の状態を復元します。`HighPriority` がターミネートすると、タスク `LowPriority` は、プリエンプトされたポイントから処理を再開し、残りの所要時間 1ms を実行します。

この実行パターンは図 4-4 のようになります。

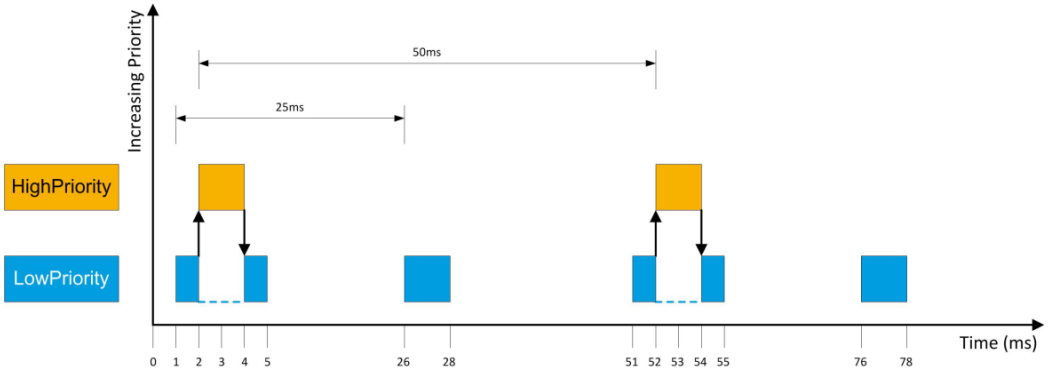

図 4-4　Hello World のタスクの実行

4.3.2　プログラム実行状態の検証

`Target.h` 内に `IO_PIN1` および `IO_PIN2` として定義されている I/O ピンにオシロスコープのプローブを接続することにより、2 つのタスクが実行されるようすをモニタすることができます。

図 4-5 では、プログラム実行時の I/O ピンの状態が、オシロスコープ上に 2 本の信号トレース（軌跡）で表されています。ここでは横軸の 1 目盛が 10ms に相当します。

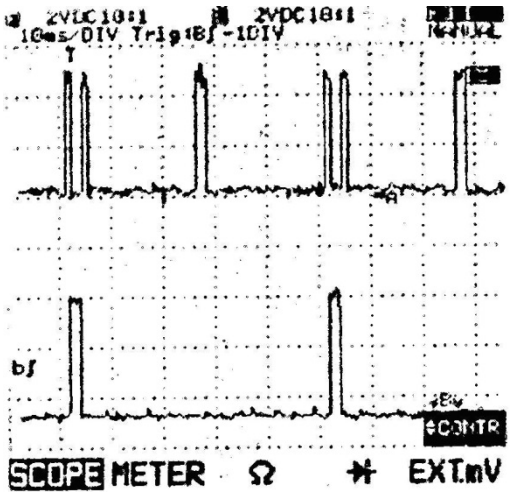

図 4-5　Hello World のオシロスコープトレース

4.3.3　トラブルシューティング

アプリケーションが正常に実行されていないように見える場合は、デバッガを使用することにより、ISR が正しく呼び出されているかどうかを確認することができます。ISR（`Os_Entry_MillisecondISR`）内の最初の命令にブレイクポイントを設定し、アプリケーションがそこに達するかどうかを調べてください。

ISR が実行されていれば、少なくともカウンタはチックされていることがわかります。続いて 2 つのタスク（`Os_Entry_HighPriority` と `Os_Entry_LowPriority`）が実行されているかどうかを調べるには、各タスク内にブレイクポイントをセットします。タスクが実行されているにもかかわらず出力が変化しない場合は、ハードウェアの初期設定を確認してください。

オシロスコープ上にトレースが表示されない場合は、Target.h 内の IO_PIN1 と IO_PIN2 の設定を調べてください。また、IO_PIN 制御マクロを調べて、各マクロがターゲットハードウェア上の同じ I/O ポートを参照しているかどうかを確認してください。トレースが示すタイミング挙動が想定されるタイミング挙動と異なっている場合は、タイマハードウェアが正しく設定されているかどうか、またターゲットハードウェアのインストラクションレートがコンフィギュレーションに設定されている値と一致しているかどうかを確認してください。

4.4 Clive Devices

サンプルアプリケーション Clive Devices には、さらに高度なコンフィギュレーションが使用されています。Hello World の場合と同様に、マルチコアターゲットが選択された場合は 2 個のコアがサンプルアプリケーションの処理を分担し、Eating タスクと OnThePhone タスクがコア 1 で実行されます。

このアプリケーションは Clive Devices の一生をモデリングしたものです。Clive の 1 日の長さは 24 時間で、その時間の中で Clive は以下の作業（タスク）を優先度の高いものから順に行います。

タスク	所要時間	いつ実行するか
OnThePhone	15 分	午前 7 時から午後 7 時までの毎正時
Eating	1 時間	朝食は午前 7 時、昼食は正午、夕食は午後 8 時
Working	8 時間	1 日に 1 回、午前 8 時に開始
Sleeping	8 時間	1 日に 1 回、午後 10 時に開始

Clive の自由時間はアイドルタスク Os_Cbk_Idle を使用してモデリングされています。

タスクの Eating、Sleeping、Working は DailyRoutine というスケジュールテーブルから起動されます。このスケジュールテーブルは 24 チック（24 時間に相当します）ごとに先頭に戻ります。

タスク OnThePhone はアラーム PhoneRings により駆動されます。このアラームは毎正時（ただし、Clive が夜間に起きないですむように午前 7 時から午後 7 時までのみ）に駆動されます。

スケジュールテーブルもアラームも、 MillisecondInterruptHandler という ISR により処理される 1ms タイマ割り込みでチックされるソフトウェアカウンタにより駆動されます。つまり、実時間の 1 ミリ秒が Clive の生涯の 1 時間に相当することになります。

Clive は Eating（食事）と OnThePhone（電話）を同時に行うことはできません。なぜなら、食べながら話をすることはできないからです。この「排他処理」を実現するため、Mouth という標準リソースを使用し、このリソースを Eating および OnThePhone の両タスクで共用するようにします。ただし、Eating タスクにおいては、全体の 1 時間のうち、最初の 15 分間で食事を準備し、次の 30 分間でそれを食べ、その後 15 分間は消化のために休みます。つまり Mouth を使用するのは 30 分間のみです。

Clive の一生は RTA-OS の始動時に始まります。Clive は永遠に生き続けるわけではないので、GrimReaper という名前のアラームが用意されていて、これは Clive が生まれてから 365 日

後に満了するようにセットされています。このアラームは TimeToDie というコールバックを実行します。このコールバックは、次の自由時間が来たとき（つまり次に Os_Cbk_Idle がレジュームされた時）に Clive が死ぬことを示すフラグをセットします。

これらのアラームとスケジュールテーブルは、すべて RTA-OS が始動されてから 1ms 後に自動始動されます。

4.5 Pizza Pronto

サンプルアプリケーション Pizza Pronto では拡張タスクが使用され、システム上で拡張タスクを実行するために必要なスタック使用量の設定例も含まれています。拡張タスク用のスタック設定についてはユーザーズガイドを参照してください。

Pizza Pronto 用に生成されるコードは、EatPizza タスクをマルチコアターゲットのコア 1 に割り当てます。

このアプリケーションは複数のプロジェクトに取り組んでいるソフトウェア開発者をモデリングしたものです。各プロジェクトの開発には 100 分かかり、200 分ごとに新規プロジェクトが到来します。

開発者は常にお腹を空かせていて、各プロジェクトの作業が 20 分経ったところで空腹を満たすためにデリバリーピザを注文する必要があります。ピザを注文するタスクには 5 分かかります。

注文してからピザが届くまでにかかる時間は、ピザ屋がすでに処理している注文の数や配達にかかる時間に応じて変わりますが、最短で 1 分、最長で 61 分です。

開発者はピザの注文を終えるとプロジェクトの作業を再開します。その後、ピザが到着するとドアベルが鳴るので、ピザを受け取って食べます。これにかかる時間は 20 分です。

ピザを食べ終えたら、プロジェクトの作業を再開し、終了させます。

タスクは、優先度の高いものから順に、以下のように定義されています。

タスク	所要時間	タイプ	周期
OrderPizza	5 分	基本	WriteCode にトリガされる
EatPizza	20 分	拡張	
WriteCode	100 分	基本	200 分

Clive Devices の場合と同様に、時間はミリ秒単位に量子化され、実時間の 1 ミリ秒が開発者の時間の 1 分を表します。また同様に、MillisecondInterruptHandler という割り込みサービスルーチン（ISR）により処理される 1ms タイマ割り込みが使用されます。

タスク WriteCode は、自動始動される Project というアラームにより 200ms ごとに起動されます。WriteCode は、実行開始から 20ms 経過するとタスク OrderPizza を起動します。OrderPizza は WriteCode よりも優先度が高いので、WriteCode をプリエンプトします。ピザの注文が終わると WriteCode はレジュームし、残りの処理時間 80ms を実行します。

タスク `OrderPizza` は 5ms にわたって実行され、処理の最後に `Doorbell` というシングルショットアラームをセットアップします。このアラームの満了時期は 1ms から 61ms までの乱数としてプログラムされています。このアラームは、満了するたびにピザの到着を示す `Delivery` というイベントをセットします。

タスク `EatPizza` は、ピザの `Delivery` を待ち、20ms にわたって実行された後、再び次の `Delivery` を待ちます。この挙動は「`WaitEvent()`呼び出しを含む無限ループ」という一般的な拡張タスクとして実装されています。このタスクは RTA-OS が始動されると自動始動されるように設定されています。

図 4-6 に示すタスクの実行例では、初回のピザが届くまでに 40ms かかり、2 回目は 10ms で届いています。

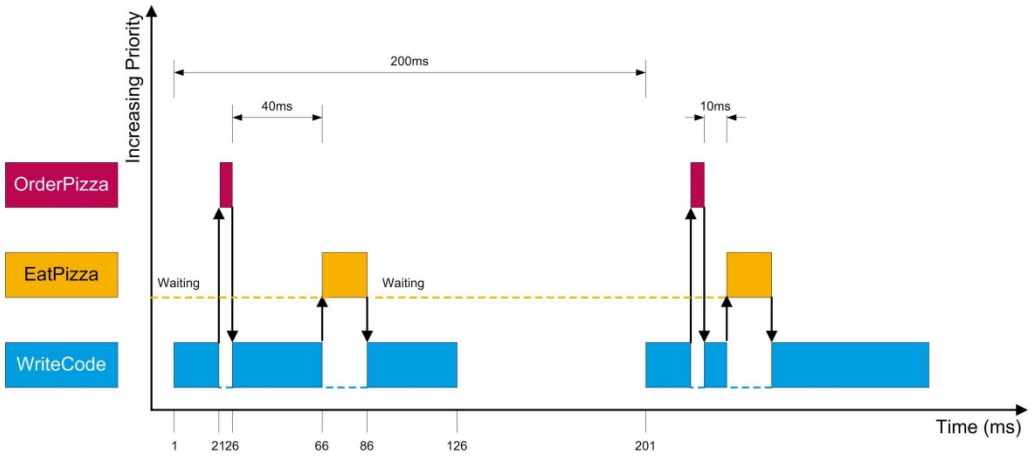

図 4-6　PizzaPronto のタスクの実行例

このサンプルアプリケーションでは拡張タスク（`EatPizza`）を使用しているので、全体のスタックサイズと各タスク／ISR のスタックサイズを定義する必要があり、その情報に基づいて RTA-OS はランタイムにスタックを管理します。**rtaoscfg** の **General → Target** を選択して **Target Specific** タブを開くと、現在のターゲット用の設定値を確認できます。

4.6　RTA-TRACE との統合

各サンプルアプリケーション（Hello World、CliveDevices、PizzaPronto）は、コンフィギュレーション設定により RTA-TRACE を有効にするだけで、RTA-TRACE の機能を使用できるようになっています。デフォルトにおいて RTA-TRACE は無効になっていますが、`RTA-TRACEIntegration.xml` というファイルには、RTA-TRACE を有効にするサンプルコンフィギュレーションが定義されています。

組み込みターゲットの場合、RTA-TRACE は、デバッガまたはシリアル（RS232）リンクを使用してターゲットのトレースデータを PC に転送します。**rtaosgen** では、ユーザーのハードウェアとともに使用できるサンプルのシリアルリンクを生成することができます。

VRTA ターゲットの場合は、バーチャル ECU と RTA-TRACE との間で直接データを転送する高速 TCP/IP データリンクを使用し ます。

RTA-TRACE をサンプルアプリケーションに統合するには、以下のように操作します。

1. **rtaoscfg** で `.rtaos` プロジェクトファイルを開きます。

2. RTA-TRACE Configuration ワークスペースに切り替えます。
3. **Enable Tracing** を TRUE にします。
4. プロジェクトを保存します。
5. `build.bat` を実行してサンプルアプリケーションをリビルドします。
6. `run.bat` を使用してサンプルアプリケーションを実行します。

トレースデータを参照するには、以下のように操作します。

1. RTA-TRACE を起動し、メインメニューから **File → New Connection** を選択します。
2. ターゲットのロケーションを入力するダイアログボックスで、localhost のデフォルトを適用します。
3. OSインターフェースを選択するダイアログボックスで、以下のいずれかを選択します。
   - RTAOSEK-Debugger または RTAOSEK-RS232（組み込みターゲットの場合）
   - RTAOSEK-VRTA（VRTA の場合）
4. ファイル選択ダイアログボックスで、サンプルアプリケーションのフォルダからファイル`<ApplicationName>.rta` を選択します。

VRTA ターゲットの場合は自動的にアプリケーションが起動され、RTA-TRACE ウィンドウにデータが表示されます。組み込みターゲットの場合はマニュアル操作でアプリケーションを起動します。

4.7 ターゲット固有オプションの使用

ターゲットのタイプに応じて、メモリモデルや実行するメモリ領域、デバッグ機能などに関するターゲット固有のオプションがあります。

適切なオプションをコマンドライン引数として渡すことにより、これらのオプションを使用するサンプルアプリケーションを生成することができます。

以下のコマンドを実行すると、ターゲット用オプションのリストが表示されます。

```
rtaosgen --target:<target> --target_option:?
```

リストの中から必要なオプションを選び、サンプルアプリケーションを生成するコマンドに以下のように追加します。

```
rtaosgen --samples:[Applications] --target:[<variant>]<target>
     --target_option:option1=value1
     --target_option:option2=value2 ...
```

統合のためのアドバイス 4.3： ターゲットオプション名にスペースが含まれている場合（たとえば `Stack used for C-startup` など）には、以下の例のようにそのオプションを二重引用符で囲みます。

```
rtaosgen --target:[MinGW]VRTA --target_option:"Stack used for
     C-startup"=64 HelloWorld.rtaos --include:Samples\Includes
```

5 参考資料

5.1 RTA-OS のユーザードキュメント

RTA-OS をインストールすると、以下のユーザードキュメントが所定のフォルダにコピーされます。

> **注記：**
> **Getting Started**（入門ガイド- 本書）と **User Guide**（ユーザーズガイド）につきましては、V5.4 より日本語版が用意されています。詳しくはサポート窓口までお問い合わせください。

<install dir>\Documents

Getting Started（入門ガイド）： 本書です。製品のインストール方法やオペレーティングシステムの基本原理について説明し、サンプルアプリケーションの扱い方についても説明されています。

Release Note（リリースノート）： 従来のリリースからの変更点のリストや既知の問題のリストなど、リリースに関する情報が掲載されています。

User Guide（ユーザーズガイド）： AUTOSAR OSの背景や概念について説明し、RTA-OSツールを使用してOSを設定し、アプリケーションに統合する方法を示しています。V5.4より日本語版も用意されています。

Reference Guide（リファレンスガイド）： RTA-OS用のAPIとプログラミング規則についての詳細情報が記載された参照ガイドです。

Analysis Visualizer User Guide（Analysis Visualizerユーザーズガイド）： スケジューラビリティ分析の背景や概念について説明し、RTA-OSベースのシステムをビルドして分析する方法を説明しています。

<install dir>\Targets\VRTA_n.n.n

VRTA Port Guide（VRTAポートガイド）： VRTAポートプラグインに関する詳細な実装情報について説明しています。

VRTA Release Note（VRTAリリースノート）： 従来のリリースからの変更点のリストや既知の問題のリストなど、VRTAポートプラグインのリリースに関する情報 が掲載されています。

Virtual ECU User Guide（バーチャルECUユーザーズガイド）： VRTAポートプラグインに同梱されているバーチャルECU環境の使用方法について説明しています。

<install dir>\Targets\<TargetCompiler>_n.n.n

Target/Compiler Port Guide（ターゲット／コンパイラ ポートガイド）： RTA-OSの各ポートに同梱されている「ポートガイド」には、RTA-OSとユーザーのツールチェーンやターゲットハードウェアとのインタラクションに関する具体的な情報（有効なコンパイラオプション、レジスタ設定、割り込み処理など）が記載されています。また、OSのパフォーマンスやリソース使用に関する情報も記載されています。

Target/Compiler Release Note（ターゲット／コンパイラリリースノート）： 従来のリリースからの変更点や既知の問題点など、ポートプラグインのリリースに関する情報が掲載されています。

5.2 関連資料

5.2.1 OSEK

OSEK/VDX Operating System（バージョン2.2.3、2005年2月17日）は、OSEK OSの仕様書です。OSEK OSの挙動について詳しく記述されています。

OSEK/VDX Binding Specification（バージョン1.4.2、2004年7月15日）には、OSEK OSと他のOSEK規格との相互運用の方法が記述されています。

OSEK Run-Time Interface (ORTI)（パートA：Language Specification　バージョン2.2、2005年11月14日、およびパートB：OSEK Objects and Attributes バージョン2.2、2005年11月25日）には、OSEK OSにサポートされているコアORTIデバッグ言語および標準属性が定義されています。

5.2.2 AUTOSAR

RTA-OS は AUTOSAR の R3.0 から R4.0 Rev1 までの一連のリリースに適合しています。以下の資料は、各リリース別に用意されています。

AUTOSAR SWS OS： AUTOSAR OSの仕様書です。 AUTOSARのOSEK OSエクステンションについて詳しく記述されています。

AUTOSAR SWS GPT Driver： OSが使用するAUTOSAR “General Purpose Timer”（Gpt）ドライバについて説明しています[4]。

AUTOSAR SWS RTE：AUTOSAR RTEについて記述されています。AUTOSAR RTEとAUTOSAR OSとのインタラクションの方法に関する情報が必要な場合は、この文書をお読みください。

AUTOSAR SRS General： すべてのAUTOSAR基本ソフトウェアモジュールに適用される一般的性質について記述されています。AUTOSAR OS文書には、この一般的なSRSから逸脱している事柄のうち、このOSに適用されるものについて具体的に記載されています。

AUTOSAR SWS StandardTypes： OSのような基本ソフトウェアモジュールのためのAUTOSARの標準データ型およびインクルード構造が定義されています。

AUTOSAR SWS PlatformTypes： ターゲットプラットフォームの特性がAUTOSARでどのように抽象化されるかが記述されています。

[4]注 － AUTOSAR OS仕様ではGptを使用することをOSに要求していますが、実際のところ、Gptは実用に適するほど強力ではありません。そのためRTA-OSはGptドライバを直接は使用しません。詳細については『リリースノート』を参照してください。

AUTOSAR SWS MemoryMapping： AUTOSARにおけるソフトウェアのメモリマッピングの扱いについて説明されています。RTA-OSはこの概念を使用して、ユーザーが指定したセクションにOSのパーツを配置します。

AUTOSAR SWS CompilerAbstraction： コンパイラのプロパティ（型定義、ポインタ、near／farアドレス空間へのアクセスなど）がAUTOSARでどのように抽象化されるかが記述されています。RTA-OSはこの概念を使用してすべての内部OSコードをAUTOSAR規格に適合させます。

6 お問合せ先

6.1 テクニカルサポート

サポート契約を結ばれたお客様は、テクニカルサポートをご利用いただけます。まだサポート契約を結ばれていないお客様は、担当の営業窓口までお問い合わせください（6.2.2 項を参照してください）。

テクニカルサポートをご利用いただく際には、E メールでご連絡していただくのが最も確実な方法です。製品の使用について問題やご不明な点がございましたら、下記アドレス宛の E メールでお気軽にお問い合わせください。

ec.hotline.jp@etas.com　（日本国内のホットライン窓口）

テクニカルサポートチームに直接お話しになりたい場合は、ホットライン窓口の下記番号までお電話ください。

045-222-0951　（日本国内のホットライン窓口）

お電話での受付時間は、平日の営業時間内（9:00～18:00）です。

どちらの場合も、以下の情報をご提供いただけると問題解決に役立ちます。

- お客様のサポート契約番号
- ユーザーデータ（`.xml`、`.arxml`、`.rtaos`、`.stc` ファイル）
- エラーの原因となったコマンドライン
- ご使用の ETAS ツールとそのバージョン
- ご使用のコンパイラツールチェーンとそのバージョン
- システムからエラーメッセージが出された場合は、そのメッセージ
- `Diagnostic.dmp` というファイルが生成された場合は、そのファイル

6.2 全般的なお問合せ

6.2.1 ETAS 本社

ETAS GmbH

Borsigstrasse 14	Phone:	+49 711 3423-0
70469 Stuttgart	Fax:	+49 711 3423-2106
Germany	WWW:	www.etas.com

6.2.2 セールスとテクニカルサポートの窓口

各地域の事業所（セールスオフィス）とテクニカルサポートの連絡先は、ETAS のウェブサイト（下記）でご案内しています。

ETAS 事業所（一般的なお問合せ）	www.etas.com/ja/contact.php
ホットライン窓口（テクニカルサポート）	www.etas.com/ja/hotlines.php