TOPPERS 活用アイデア・アプリケーション開発コンテスト

部門　：　(活用アイデア部門)　アプリケーション開発部門

作品のタイトル　：　.NET Micro Framework for TOPPERS

作成者　：　株式会社デバイスドライバーズ（代表：日高 亜友）

対象者　：　組込みシステム初心者～業務システム開発者

使用する開発成果物　：　TOPPERS/ASP カーネル

目的・狙い

.NET Micro Framework（NETMF）を TOPPERS 上に移植することで、これまでの組み込みシステムでは実現できなかった、①リアルタイム性能の確保　②容易なシステム開発　③豊富なミドルウェアサポート　をマルチプラットフォームで実現する。この実装の NETMF のハードウェアに依存しないコア・コード部分は全ての TOPPERS で利用可能になるため、組み込みシステム初心者から業務システム開発者まで、幅広いユーザーへの普及が可能になり、次世代組み込み OS 実装事例として研究用にコードを開示し、NETMF 開発コミュニティにもフィードバックする。

アイデア/アプリケーションの概要

TOPPERS/ASP+TECS 上に.NET Micro Framework を OS on OS の形態で移植する。TINET が稼働する移植済みの TOPPERS 環境に、TECS を使用して NETMF の OS Facilities 層を構築する。TOPPERS 上と NETMF 上の両方のユーザーアプリケーション連係動作をサポートするために TOPPERS には NETMF 拡張サービスコールを設け、NETMF には.NET アプリケーションから制御可能な TOPPERS InterOp クラスライブラリを構築する。Windows 上の同じ GCC クロスコンパイラにより TOPPERS と NETMF の両方をビルド可能とし、メンテナンスを容易にする。

TOPPERS 活用アイデア・アプリケーション開発コンテスト

応募作品の説明

2013 年 9 月 30 日

株式会社デバイスドライバーズ　日高亜友

hidaka@devdrv.co.jp

目次

1. はじめに

応募作品の説明の前に、今回のアイデアのテーマである.NET Micro Framework について簡単に紹介します。

1.1. .NET Micro Framework とは？

.NET Micro Framework （以下 NETMF）は米国 Microsoft Research が開発して、2001 年に北米で販売された SPOT Watch と呼ぶ多機能腕時計に採用された小規模組み込みシステム用のオペレーティングシステム(OS)とその開発環境です。

数百 KB 程度のメモリと MMU 無しの MCU 上で MSIL（Microsoft Intermediate Language＝.NET のアセンブリ、中間コード)をインタプリタ動作させるランタイムとそれをサポートする OS、ライブラリ、デバイスドライバ、ブートローダと Visual Studio の SDK で構成され、C#等の.NET 言語で記述したプログラムを Visual Studio 上でソースコードデバッグしたり、エミュレーターで動作させたりしながら、容易に組み込みアプリケーションを開発することが可能です。

現在はオープンソース化されて CodePlex（http://netmf.codeplex.com/）にてコミュニティにより開発・メンテナンスされています。

20 種以上の組み込みボードや様々な CPU に移植され、豊富なデバイスドライバと、ファイルシステムやプロトコル、サービスを含む豊富なミドルウェアのサポートにより業務用途や、教育用組み込みシステムで多数利用されています。

1.2. 特長（Mono との対比）

同じくオープンソースのマルチプラットフォームで動作する Mono プロジェクトと比較される場合がありますが、主に次の点が異なります。

① NETMF は無償の Express 版を含む Visual Studio のコンパイラを使用してビルドするため、コンパイラ・エンジンは Windows 用と同じです。すなわちエラー処理やオプション対応等のコンパイラの振る舞いや生成コードが共通で、Visual Studio に付属する多くのツールや機能が利用可能です。

② NETMF は小規模組み込み用途での利用を想定したモジュール化が推進され、ROM/RAM 数百 KB 程度のメモリで動作します。

③ NETMF には JIT（Just In Time）コンパイラ機能が無く、常に中間言語インタプリタで動作します。

1.3. OS 上への移植

以下に NETMF の構成図を示します。今回のアイデア実現と関係が深い特徴は、.NET アセンブリの実行環境である CLR(Common Language Runtime)をサポートするための仮想マシン環境は、PAL(Platform Abstraction Layer) で抽象化されていて、この PAL はハードウェアとデバイスドライバで構成される Native Platform の HAL 上だけでなく、OS Facilities をサポートする OS on OS 形態のソリューションにも対応可能な設計となっていることです。

今回の NETMF の移植に関するアイデアでは、TECS を使用してこの OS Facilities 層をどのように構築するかということが課題となります。

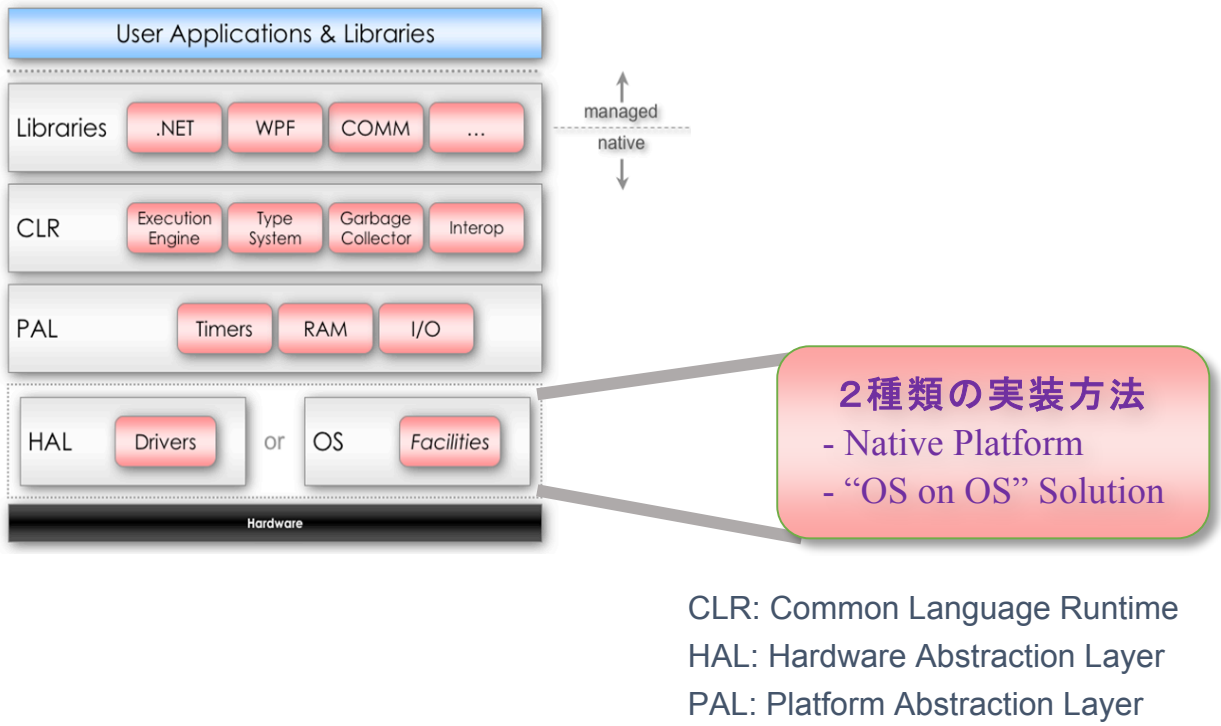

図 1． .NET Micro Framework の基本構造

1.4. 参考情報

今回の OS on OS 形態での TOPPERS への NETMF の移植については、日高が 2011 年に執筆した以下の Linux 上への移植の記事と、公開済ソースコードを参考にします。

① インターフェース誌 2011 年 4 月号
　技術解説 Linux 上への.NET Micro Framework 移植の勘所（前編）
② インターフェース誌 2011 年 6 月号
　技術解説 Linux 上への.NET Micro Framework 移植の勘所（後編）
③ ソースコード：CodePlex .NET Micro Framework for Linux
　http://netmflinux.codeplex.com/

1.5. ライセンス

.NET Micro Framework は CodePlex 上にて、Apache License 2.0 ライセンスに基づいて公開されています。(http://netmf.codeplex.com/license)

前項の Linux 上への移植例も同様ですが、今回の開発においてこの CodePlex 上の NETMF オリジナル・コードを改変した部分、あるいはオリジナル・コードを参考にして作成した部分に関しては、Apache License 2.0 ライセンスを引き続き適用するものとします。

OS Facilities 層を構築する TECS とそれに追加変更する部分に関しては、引き続き、TOPPERS ライセンスを適用するものとします。ともにオープンソースが前提のライセンスのため親和性は良く、同時にソースコードを配布することが可能と考えています。

2. 開発方法

以下に今回のアイデアの実現に際して想定している開発方法を簡単に示します。

2.1. コンパイラの選択

NETMF のビルドには ARM RVDS や ARM MDK, HEW 等も利用可能ですが、今回の開発の趣旨の一つとして、研究目的用にコードを開示することと、マルチプラットフォームサポートの観点から、コンパイラは GCC 系統の採用を優先して検討します。

NETMF のソースコードは gcc v4.2.1 以降をサポートしていて Windows 上で動作する以下のようなコンパイラが利用可能なため、TOPPERS と共通のクロスコンパイラが利用可能です。採用するクロスコンパイラの候補として、次のようなものがあります。後述する開発ターゲット環境によっても使い分けることが可能です。

① Cygwin ベースのクロスコンパイラ（自前で移植）
② MinGW ベースのクロスコンパイラ（自前で移植）
③ Windows 上で動作するクロスコンパイラ・パッケージ
　例：Launchpad GNU Tools, devkitPro, Yagarto, emIDE, , PizzaFactory,
　Mentor Graphics Sourcery Tools / Sourcery CodeBench (旧 CodeSourcery),

なお NETMF のビルド手順では Windows 上で動作する Visual Studio SDK 付属の MSBuild ツールやコマンド・プロンプト（バッチファイル）と VBScript 等の Windows 環境固有のスクリプトの動作が不可欠であるため、今回の開発プラットフォームとしては Windows（XP 以降）を対象とします。

Mac OS や Linux 等の他の環境でも、NETMF のビルドに必要な Windows スクリプト実行環境を構築することにより NETMF のビルドが可能になりますが、それらのプラットフォーム上での開発環境構築は本テーマの目的から逸脱するため、今回は配慮しません。

2.2. ターゲット環境

移植作業開始時点で TOPPERS / ASP+TECS と TINET が安定動作している環境を当面の移植のターゲットとします。今回はマルチプラットフォームのサポートが目的でもあるため、移植先のターゲットは一つに絞りません。

しかしながら調査したところ、現時点で明確に TINET をサポートしている TOPPERS / ASP+TECS 環境が見つかりませんでした。今後利用になるとの予測も含めて、現時点では以下のプラットフォーム等が実装先ターゲットの候補と考えられます。

① Armadillo-800 EVA
② アルファプロジェクト AP-SH4A-0A
③ LEGO Mindstorms EV3
④ アルファプロジェクト XG1808（③と同じ CPU 搭載）
⑤ GHI FEZ Hydra （NETMF 稼働済 Open Hardware ボード）
⑥ デバイスドライバーズ E!Kit-BF533（TOPPERS/JSP 移植済）

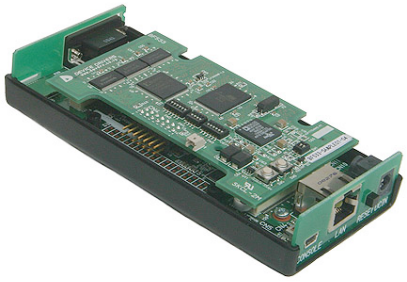

写真１． E!Kit-BF533

2.3. 構成概要

以下に NETMF を TOPPER 上に移植した場合のシステム全体の構成概要図を示します。

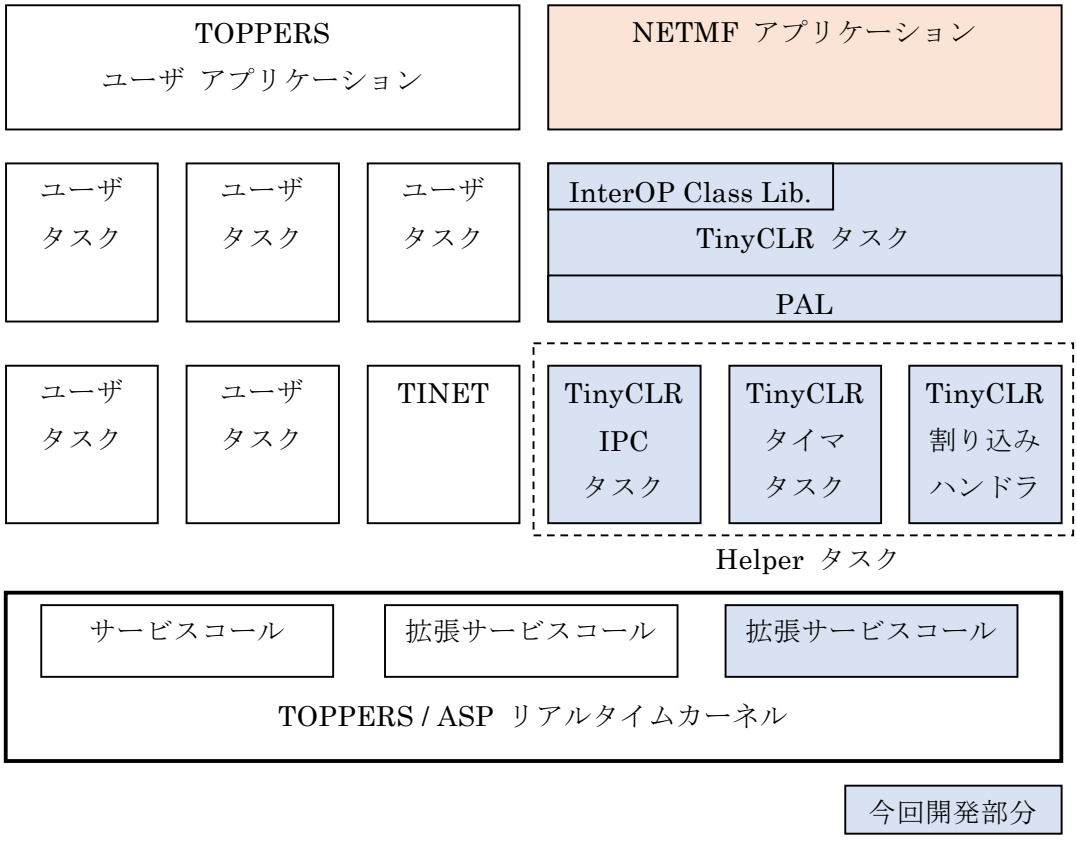

図２． NETMF の TOPPER 上への移植構成概要図

2.4. 作業概要

NETMF を TOPPERS 上に移植するための作業は次の通りです。

① ビルド環境の構築

TOPPERS のビルド環境を参考にして、同一のクロスコンパイラ NETMF 用のビルド環境を構築します。具体的には環境変数、ビルドスクリプト、リンカスクリプト、NETMF Solution ファイル、Project ファイルを作成または修正して、NETMF の NativeSample とライブラリ、単純な拡張サービスコール（ブリッジ機能）をサポートする various.cpp がエラー無く TOPPERS とリンクできることを確認します。

② OS Facilities 層の構築

TECS を使用して、NETMF 側に TOPPERS のリソースを提供する OS Facilities 層を設計、開発して組み込みます。移植における中心的作業部分です。NETMF のインターフェースは PAL (Platform Abstraction Layer) に合わせるため、この OS Facilities 層がデバイスドライバの機能を兼ね備えることになります。

TOPPERS が提供するリソースには標準的にサポートするタイマ等の各サービスコールのほかに、ネットワーク、シリアルポート、GPIO、I2C、SPI、グラフィックなどのドライバ層や拡張サービスコールと Flash ROM の入出力を含みます。

NETMF 側では、開発時にこれらのすべて必要な機能を呼び出す NativeSample を動作させて各サービスコールとデバイスドライバの動作をデバッグ、検証します。

③ Helper タスクの開発

NETMF は本来、ハードウェア上での動作を前提に開発されているコードのために、OS 上への移植を行う際には、前項の OS Facilities 層では吸収できない非同期動作をサポートする、独立した Helper タスクが必要です。TECS を使用してこれを開発します。各 Helper タスクが NETMF インタプリタの TinyCLR に提供する機能は次の通りです。

a. TinyCLR IPC タスク

NETMF アプリケーションと TOPPERS との双方向 IPC（Inter Process Communication）をサポートします。

b. TinyCLR タイマタスク

TinyCLR に対して定期的と非定期的なタイマをサポートします。

c. TinyCLR 割り込みハンドラ

TinyCLR に対して各種ドライバを含む割り込み動作をサポートします。

④ TOPPERS 拡張サービスコールの開発

スケジューリング、GC(Garbage Collection)、デバッガ制御等の NETMF の挙動を

制御する機能と、NETMF アプリケーションと双方向リアルタイム通信する機能を、TOPPERS 拡張サービスコールを開発して組み込みます。

⑤ TOPPERS InterOp クラスライブラリ構築

マネージドコードで記述された NETMF アプリケーション側から呼び出して TOPPERS の拡張サービスコールと連携するための汎用的なクラスライブラリを NETMF InterOp 機構を利用して実装します。

⑥ デバッグ機能の実装

Visual Studio が稼働する Windows ホスト PC に対して USB（移植ターゲットがデバイス側）、シリアルポート(RS232C)、LAN(TCP/IP)の 3 種類の方式で接続して、NETMF アプリケーションのデプロイとソースコードデバッグをサポートする、デバッグ機能を移植します。LAN 接続では物理層に依存せずにデバッグできるようにします。

⑦ CLR 動作検証

NativeSample を MSIL インタプリタとデバッガインタフェイスを備える TinyCLR に入れ替えて TOPPERS と一体化してビルドし、実際に NETMF アプリケーションをロードしてシステム全体の動作検証を行います。

3. アイデアのポイント

今回提案のアイデアの注目点は次の 3 点です。

① アプリケーション間連携
② マルチプラットフォーム
③ NETMF ミドルウェアの活用

以下に各注目点の詳細を示します。

3.1. アプリケーション間連係

OS 実装や動作環境にもよりますが、NETMF は実際 1ms～20ms 程度のコンテキスト・スイッチ能力と外部制御可能な GC 機能、1ns 以上で粒度調整可能な内部タイマを備えます。それでも中間言語インタプリタで動作することと、GC があることや各処理の実行時間が予測できないことから、一般的にはリアルタイム処理には向かないと思われています。

その様な指摘に対して開発元の Microsoft Research では常に次の様に答えています。「NETMF はリアルタイム向けのソリューションは提供していない。必要であればリアルタイム OS 上に NETMF を実装して、リアルタイムが必要な処理部分と分ければよい。」

実際に株式会社コアでは 2009 年に.NET Micro Framework V4.0 を T-Kernel 上に移植した事例を発表しています。しかしながら現在までリアルタイム OS 上への NETMF の移植のコードはオープンソースのコードとしては公開されていませんでした。従って NETMF のリアルタイム OS 上への移植では、リアルタイム OS と非リアルタイム OS である NETMF との柔軟な連携方法を提供する必要があります。次の 3 種類の連係方法を想定しています。

① NETMF 主体の連係

アプリケーションのメインロジックを NETMF 側に置き、NETMF アプリケーションが主体となって動作します。最も多く利用されると思われる連係方法です。

リアルタイム動作が要求される処理、サービス、ドライバを利用する場合には、InterOp クラスライブラリを使用して TOPPERS のサービスやタスクを呼び出します。

TOPPERS 側には主体となって動作するアプリケーションを配置する必要はありません。

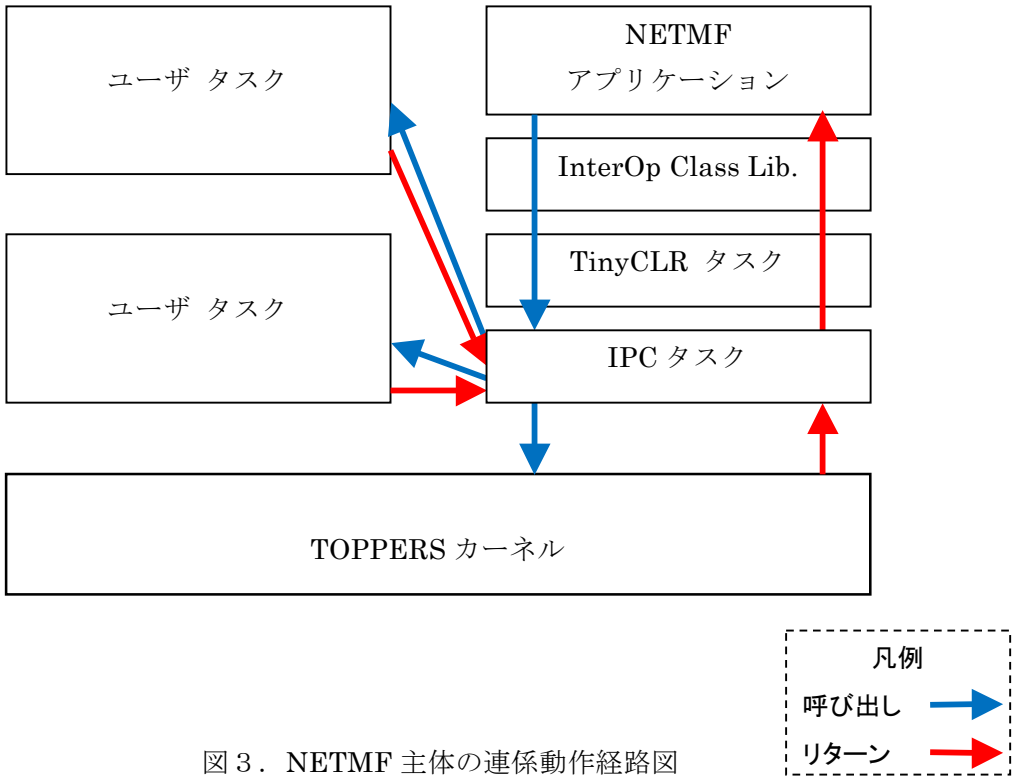

図 3．NETMF 主体の連係動作経路図

② TOPPERS 主体の連係

アプリケーションのメインロジックを TOPPERS 側に置き、TOPPERS 側が主体となって動作します。

NETMF アプリケーションは連係機能により下位層から IPC タスクを通して callback 呼び出しされる「アップコール」によって動作し、TOPPERS 側からは拡張サービスとして見えます。

この実装により NETMF アプリケーションが利用できる WCS 準拠の Web Service やセキュリティ機能、クラウド連係機能、WPF(Windows Presentation Foundation)グラフィック機能等の NETMF 固有の機能を TOPPERS アプリケーションから利用できます。

NETMF アプリケーションには特定機能をサポートするサービスを実装したサービス・アプリケーションを配置しておきます。

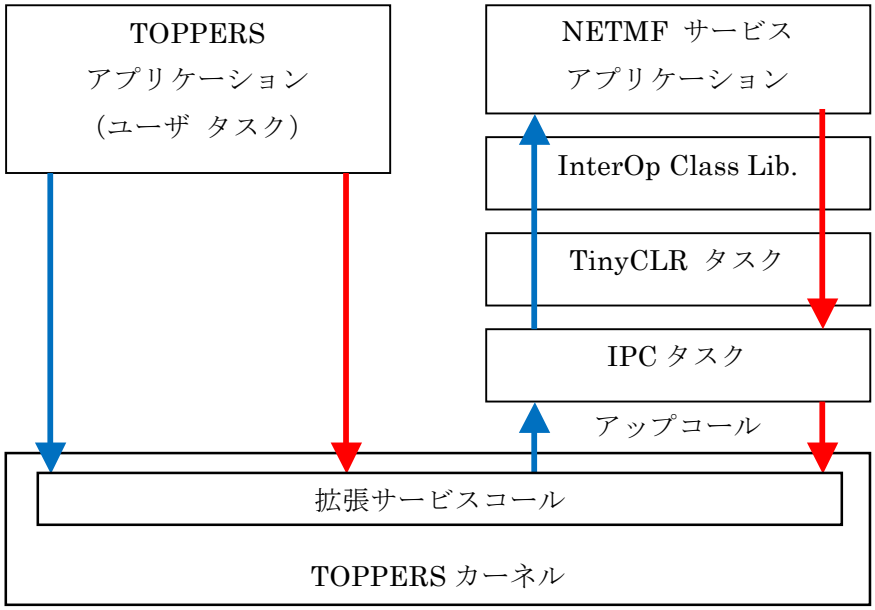

図４．TOPPERS 主体の連係動作経路図

③ 対等的な連係

前記①と②の混在利用です。様々な応用が考えられます。

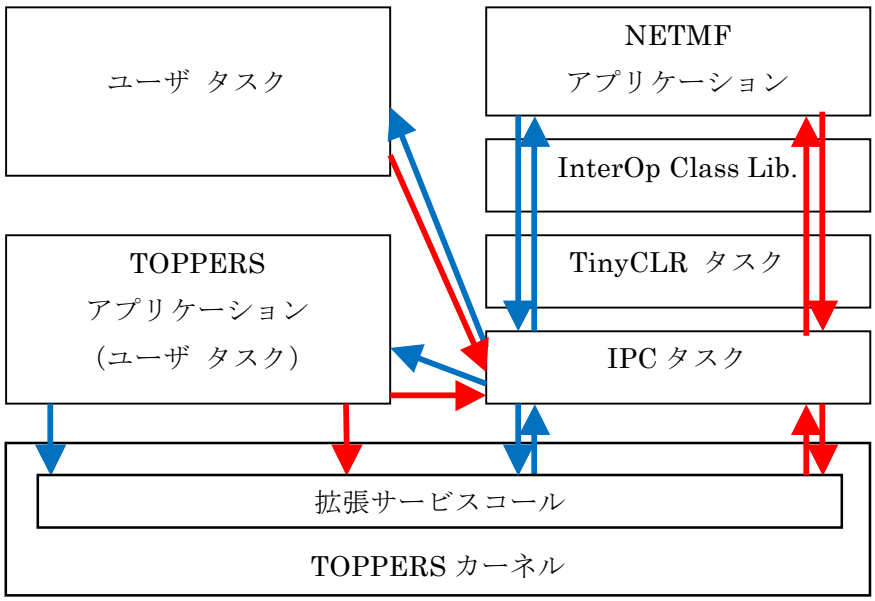

図５．対等な連係の動作経路図

3.2. マルチプラットフォーム

NETMF 稼働部分は TOPPERS で動作するユーザタスクのソースコードとして提供するため、TOPPERS が動作する環境であれば基本的には CPU やプラットフォームに関係なく動作させることが可能です。ハードウェア部分の差異は TECS, OS Facilities, PAL の 3 層で吸収するため、新しいプラットフォームへの移植も比較的容易と考えられます。

またマルチプラットフォーム展開を目的にしているため、NETMF がサポート可能な USB、シリアルポート(RS232C)、LAN(TCP/IP)の 3 種類すべてのデバッガ接続をサポートします。これにより動作環境に最適なデバッガ接続方法を選択することができます。

3.3. ミドルウェアの活用

NETMF はそれ自体のソースコードにオリジナルの 15 以上のプラットフォーム（組み込みボード）をサポートする各種デバイスドライバと 30 近くのサービスやハードウェアを抽象化した PAL と呼ぶミドルウェア層を備えます。これらのリソースを活用することで、TOPPERS の応用範囲を拡張することが可能です。

前項②で示した通り、アプリケーション間連携機能を介して NETMF アプリケーションを経由させて NETMF 用のミドルウェアを利用する方法もありますが、これらのデバイスドライバやミドルウェアは C または C++で記述されているモジュール化されたオープンソースのコードですので、TOPPERS のコードから直接呼び出したり、ライブラリ化やサービス化したりして利用することが可能です。一歩進んだ TOPPERS と NETMF の連係です。

具体的な事例としては、NETMF の「WearLeveling」という NAND Flash メモリ管理ドライバを TOPPERS から呼び出して利用することや、「BatteryModel」のドライバに適合するバッテリー管理機能を、TOPPERS のサービスとして利用することが可能になります。

4. 将来の展望

将来的には次の 2 点をサポートすることで、より一層 NETMF と TOPPERS の親和性が良くなり、一体となって動作するシステムの利点が増えると予想しています。

① JIT の実装

今回のアイデアでは依然として、NETMF のコードを実行している部分の実行時間が不明であり、またその動作時間を保障する手段がありません。このような部分があるということは、リアルタイムシステムとしては致命的な欠陥になる場合があります。NETMF に JIT を実装することである程度この問題を解決できます。

② グラフィック処理の標準化

NETMF と TOPPERS で Native コード動作の共通のグラフィック処理を利用できる様にすることで、ソフトウェア生産性の向上とシステムリソースの有効活用を図ります。

以上