OpenRTM-aist と TECS (TOPPERS Embedded Component System) の連携

Interoperation scheme between OpenRTM-aist and TECS (TOPPERS Embededded Component System)

○ 姜 榮煥（デジタルクラフト）　栗原 眞二（産総研）　韓 相勲（デジタルクラフト）
金 泰成（デジタルクラフト）　正 安藤 慶昭（産総研）

Yeonghwan KAN, Digital Craft Inc., yh-kan@aist.go.jp
Shinji KURIHARA, National Institute of Advanced Industrial Science and Technology
Sanghoon HAN, Digital Craft Inc.
Taesung KIM, Digital Craft Inc.
Noriaki ANDO, National Institute of Advanced Industrial Science and Technology

A component-based software development is widely applied to solve software complexity and maintainability problems. The TECS which is a ‥type of static component system for ITRON OS is suitable for low level software systems that require real-time capability and stability, whereas the RT-component framework is appropriate to various system integrations, since it has high flexibility and re-usability. Integrating TECS and RT-component provides flexibility and expandability to the TECS as well as stability to the RT-component, so that it makes possible to construct a flexible and stable system. In this paper methods to integrate TECS and RT-component are considered, and some examples are discussed.

Key Words: Distributed system、RT コンポーネント、RT ミドルウエア

1. はじめに

近年、先進国の少子化・高齢化により、サービスロボットの研究が進められている。サービスロボット技術は大規模化、複雑化し、同時に高信頼性、安全性が求められる。

これらに対し、大規模複雑化するロボットの機能要素を再利用可能なソフトウエアモジュールとして分割し、それらの組み合わせによりロボットシステムを構築することを目的としたソフトウエアプラットフォームとして、著者らは RT ミドルウエアの研究開発を行ってきた [1]。RT ミドルウェアはロボットの中位から上位のシステムに求められる柔軟性は高いものの、資源が制限（メモリが 1MB 以下の程度）されている組込みシステム上で直接動作させるのには適していない。

一方、組込み向けのリアルタイム OS として TOPPERS[2] プロジェクトが開発する μITRON4 準拠の TOPPERS は、様々なプロジェクトや製品に搭載されるなどの実績があるだけでなく、近年注目されている機能安全認証取得を志向するなど、ロボットシステムの下位の制御等に適した OS であるといえる。さらに、搭載するソフトウエアのモジュール化を向上させるための枠組みとして、TECS (TOPPERS Embedded Component System) といったコンポーネントフレームワークの研究開発も行われている。

TOPPERS と TECS はロボットの下位の制御システムには適しているが、GUI への拡張や柔軟なシステムへの対応は難しい。

本稿では、RT ミドルウエアと TECS （TOPPERS) の連携により、柔軟かつ安全なシステムを実現するために OpenRTM-aist と TECS の連携について実現方法と評価について述べる。

2. TECS

[3] においては、TECS と RT コンポーネント (RTC) を連携させる方法として 3 つの案が提案されたが、本稿ではこのうち、図 1 のように TECS の RPC を用いて RTC と通信する方法により連携を試みた。

2.1 TECS

組込みコンポーネントシステム TECS は、TOPPERS プロジェクトのコンポーネント仕様ワーキンググループにおいて仕様の策定が進められている、組込みシステムに適したコンポーネントシステムである [4]。

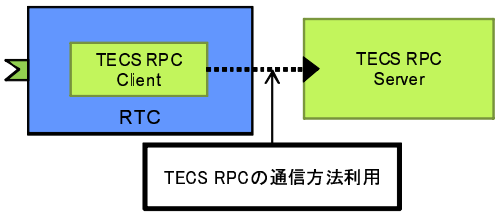

Fig.1 TECS RPC to RTC Communication

TECS の開発手順は、CDL（Component Definition Language）というコンポーネント記述言語でコンポーネントを定義し、tecsgen というテンプレートコードジェネレータにて C 言語コードを生成する。その後、コンパイルを行い実行するといった流れで開発を行う。

TECS は TOPPERS OS と Linux の両方で利用することが可能である。本稿では、実験の都合上 Linux 上で TOPPERS を動作させ RTC との連携を行ったが、原理的には同じ TECS のコンポーネントを TOPPERS OS 上で動作させることが可能である。

2.2 TECS オペイク RPC

プログラム（以下、タスク）から異なるアドレス空間上の処理を実行する手続きを RPC（Remote Procedure Call）と呼ぶ。TECS では、メモリが物理的に分離されていて共有できない場合、メモリを共有する代わりにネットワーク（TCP/IP）などを用いて関数を呼び出す事が可能である。TECS では、上記のようなリモート呼び出しの際に、メモリのアドレスが参照できないという不透明性が生じることから、不透明を意味する Opaque をつけてオペイク RPC と呼んで区別している。

TECS オペイク RPC はオペイク RPC の機能をコンポーネント化したことで複合コンポーネントとして実現されている。特徴としては、RPC に必要な最小限の通信だけをおこなうため軽量、ソケットチャンネル以外にも CAN などの組込み向けの通信チャンネルでも通信可能といった事が挙げられる。

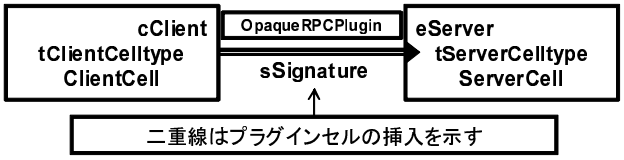

Fig.2 Insert TECS Opaque RPCPlugin

TECS のオペイク RPC は図 2 のように TECS コンポーネント間のインターフェースのコンポーネント記述（以下、シグニチャ）に TECS オペイク RPC チャンネルを入れることで実現できる。TECS オペイク RPC チャンネルは TECS コンポーネントであり、実際はプラグインとして提供される。

TECS オペイク RPC チャンネルは図 3 のようにデフォルトはソケットチャンネルのセルタイプが設定されており、必要に応じて定義したセルタイプを読み込んで置き換えることが可能である。

```
[out_through(),

to_through(rServer,OpaqueRPCPlugin,
    "taskCelltype=tTask,PPAllocatorSize=1024,\

    substituteAllocator=\"Alloc.eAlloc=>CAlloc.eAlloc\",\

    serverChannelCelltype = \"tSocketServer\",\

    clientChannelCelltype = \"tSocketClient\",\

    clientChannelCell= \"ClientChannel\"")]
```

RPC チャンネルの置き換えは、上記のように OpaqueRPC-Plugin にパラメータを設定して行う。サーバーチャンネルセルタイプの場合は、tSocketServer を、クライアントチャンネルセルタイプの場合は tSocketClient を指定することでソケットチャンネルとして通信が可能になる。

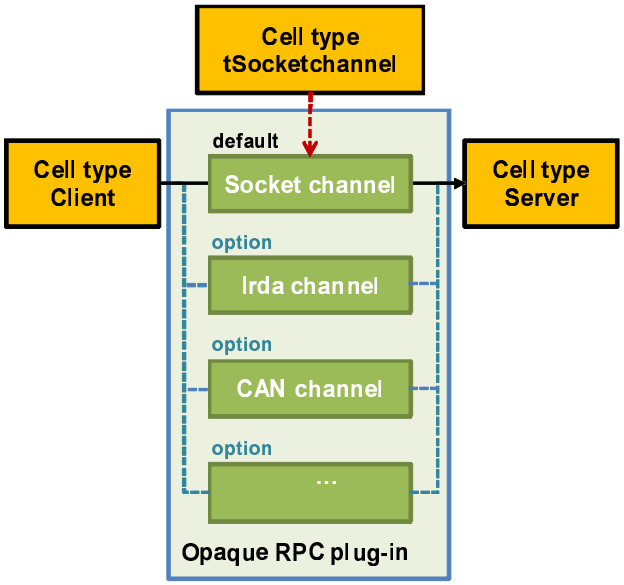

Fig.3 TECS RPC Change Channel

3. 設計方針

本節では、RTC と TECS RPC を連携するための TECS オペイク RPC のマーシャラへの対応と TECS 側に規定されているシグニチャへの対応方法について説明する。

3.1 TECS オペイク RPC のマーシャラへの対応

現在、RTC と TECS において通信可能なチャンネルは TCP/IP を利用した TECS オペイク RPC のソケットチャンネルである。TECS オペイク RPC はマーシャラ、アンマーシャラ、チャンネルなどのコンポーネントから構成される。OpaqueRPCPlugin から自動生成された、TECS オペイク RPC のクライアントのマーシャラを RTC に移植することで TECS オペイク RPC のサーバー側の関数呼び出しが可能となる。

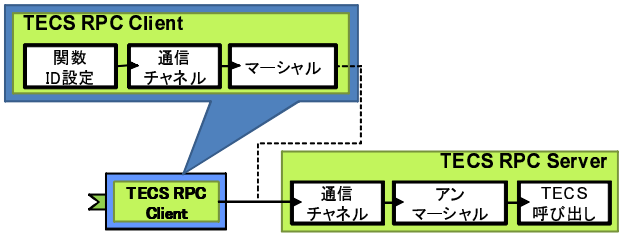

Fig.4 Insert TECS Opaque RPCPlugin

3.2 TECS オペイク RPC のシグニチャへの対応

シグニチャはコンポーネント間のセルを繋ぐ通路役割をする。一つのシグニチャで複数の関数を定義することができる。シグニチャ内の関数は ID によって区別される。OpenRTM-aist の Configuration 機能を用いることで呼び出す関数の切り替えを可能にした。

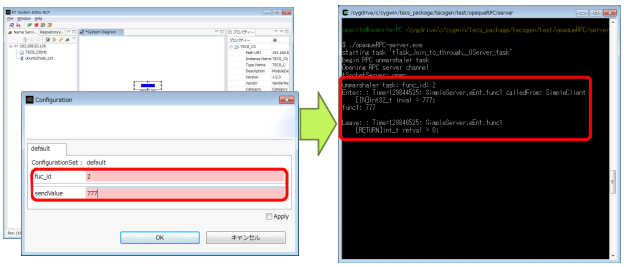

Fig.5 Using Configuration Set the function id

4. 機能・実装

4.1 TECS オペイク RPC のマーシャラの実現

RTC に TECS オペイク RPC クライアントの機能を実装する方法について説明する。

4.1.1 TECS オペイク RPC サーバー情報の設定

TECS オペイク RPC は静的なコンポーネントのため、実装の段階でサーバー側の情報をクライアント側に設定を行う必要がある。

```
#define SERVER_IP_ADDR "150.29.149.242"
#define SERVER_PORT_NO (8931)

inet_aton( SERVER_IP_ADDR, &addr.sin_addr );

addr.sin_port=htons(SERVER_PORT_NO);

if(connect(soc, (struct sockaddr*)&addr, sizeof(addr))<0)
{

close( soc );

printf("SocketClient: connect" );

}
```

RTC を実装する際に TECS オペイク RPC サーバー側に

接続するための IP とポートを設定してソケットをオープンする。

TECS オペイク RPC サーバー側のソケットがオープンされたら、マーシャリングを行う。TECS オペイク RPC は規定のメッセージでデータを挟んで送・受信することで TECS のマーシャラ・アンマーシャラ信号を判別する。

```
uint16_t sopMagic;
static int16_t len;
sopMagic=TDR_SOP_MAGIC1;

//送信をスタートするメッセージ送る
rtc_eCO_send((int8_t *)&sopMagic,
             (int16_t)sizeof(sopMagic));

//関数 ID を送信
rtc_eCO_send((int8_t*)nFunc_id,
             (int16_t)sizeof(*nFunc_id));

//データを送信する
rtc_eCO_send((int8_t*)inValue,
             (int16_t)sizeof(*inValue));
sopMagic=TDR_EOP_MAGIC1;

//データの送信済みメッセージ
rtc_eCO_send((int8_t *)&sopMagic,
             (int16_t)sizeof(sopMagic));
```

4.1.2 サーバーへのデータ送信

データの送信は上記のように TECS オペイク RPC サーバー側にマーシャラが処理を開始することを示すスタートメッセージを送る。次に、シグニチャに入っている関数の ID 番号を送信して ID 番号で呼び出せる用意ができる。関数 ID の区別ができたらデータの送信を行う。データの送信部はパラメータの定義によって変わるため OpaqueRPCPlugin からマーシャリングされたソースコードを再利用して実装を行う。データの送信が終了後、データの送信済みメッセージを送ってデータの送信を終える。

4.1.3 サーバーからのリターン値受信

RTC 側の送信が終わったら下記のようにサーバー側のリターンを待つ。サーバー側から受信スタートメッセージを受信してリターン値を受信した後、受信済みメッセージを受信する。

```
//サーバー側から受信スタートメッセージを受信
eTDR_receiveSOP(true);

//サーバー側からリターン値を受信
eTDR_getInt(returnValue);

//サーバー側から受信済みメッセージを受信
eTDR_receiveEOP(false);
```

4.2 TECS オペイク RPC のシグニチャの関数呼び出し

RTC の Configuration 機能を用いてシグニチャの関数を呼び出す機能を説明する。

CDL に記述されているシグニチャの定義関数は OpaqueRPCPlugin から自動生成された際に固有の関数 ID が発行される。ID に該当する関数を RTC に実装する。実装する際に、Configuration の値によって呼び出す関数を切り替えるような処理にすることで、RTC の実行中であっても動的に呼び出す関数を切り替えることが可能となる。

```
switch(nFunc_id)
{

  case 2:
    ファクション ID 2 ソースコード
  break;

  case 3:
    ファクション ID 3 ソースコード
  break;

}
```

5. 評価

本稿で実装した TECS と RTC の連携機能を評価するために、電動車椅子を TECS と RTC の連携により制御する。

5.1 電動車椅子

電動車椅子のジョイパッド用の RTC、TECS 対応 RTC、TECS RPC を利用してモータを制御するシステムについて説明する。

電動車椅子のノートパソコンではレーザセンサ、ジャイロセンサ、GPS、速度指令などの RTC を起動する。モータ制御 PC ではリアルタイム OS（Linux RT Preempt kernel）で 5ms 周期でモータの制御を行う。ジョイパッドからの速度指令を受けた RTC から TECS オペイク RPC にデータを変更して速度指令を行うシステムを構築した。今回のシステムでは、柔軟性が求められる上位のシステムを RTC にて構築し、リアルタイム性が求められる下位のシステムでは TECS を使用するといった RT ミドルウエアと TECS の長所を活かしたシステムとなっている。将来的にはモータ制御用 PC を TOPPERS OS が使用できるようなハードウエアにて再構築し、移動ロボットにとって利点となる小型化、省エネルギー化を図る予定である。

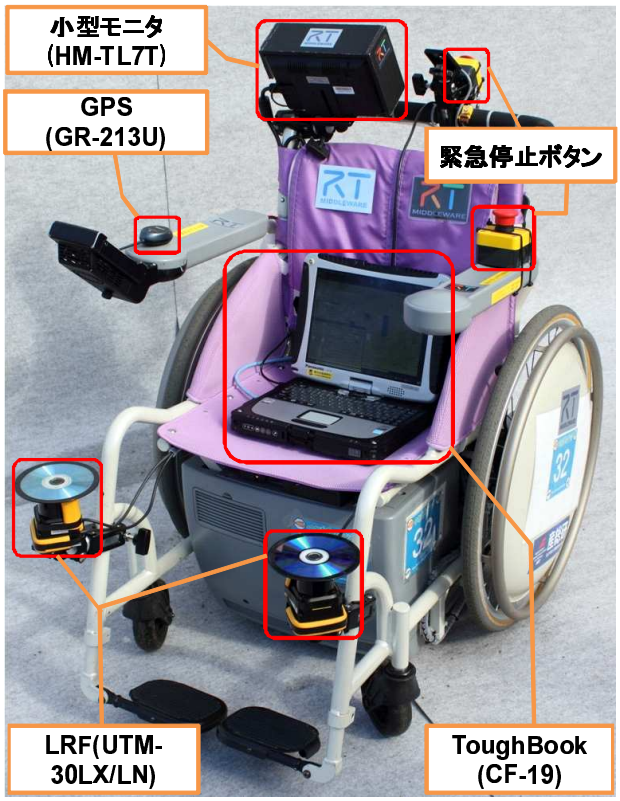

Fig.6 wheelchair robot

5.2 システム構成例

電動車椅子の RTC と TECS の連携は図 7 のようなシステム構成をしている。

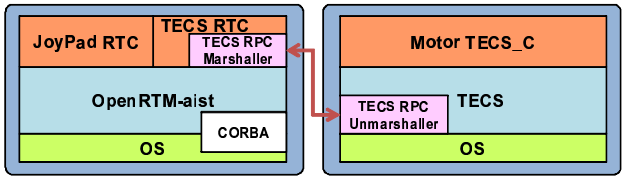

Fig.7 RTC and TECS Connect Construction

モータ制御を行う際のデータの流れを図 8 に示す。電動車椅子を制御するためには制御値を出力するジョイパッドコンポーネントとその制御値を TECS RPC Server に転送する TECS RTC を連携して制御値を転送する。その制御値は前項で説明した TECS RPC のマーシャラを用いて RTC 側から TECS RPC Server 側に転送する形である。TECS RPC 側から制御値を受けてアンマーシャリングし、関数や引数を渡してモーター制御を行う。

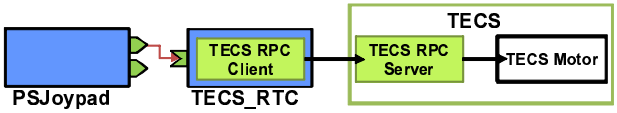

Fig.8 using RTC And TECS RPC Joypad Control to Motor

5.3 性能評価

5.3.1 CPU とメモリの使用量

性能評価のため、同じ機能を持つ RTC 版と TECS 版のモーター制御用コンポーネントを準備し、電動車椅子のモーター制御用ボード上で動作する RT Preempt Kernel 環境にて計測を行った。コンポーネントが動作する際の CPU とメモリの使用量を 10 回計測した結果を表 1 に示す。(電動車椅子のモーター制御用ボードの全体メモリは 996,436Kbyte である。)

Table 1 Used CPU and Memory

	CPU			Memory		
	Min	Max	平均	Min	Max	平均
TECS	0.70%	1.00%	0.85%	1.00%	1.00%	1.00%
RTC	9.02%	10.65%	9.84%	6.80%	6.80%	6.80%

TECS コンポーネントと RTC を比較した場合、TECS コンポーネントの CPU 使用量の方が RTC の CPU 使用量に対し約 1/10 程度となった。メモリにおいても TECS コンポーネントの方が RTC のメモリ使用量に対し約 1/7 程度となった。これらの結果により、TECS コンポーネントのほうがより軽量システムに適していると考えられる。

6. おわりに

本稿では TECS オペイク RPC のクライアント機能を RTC 側に実装し、TECS オペイク RPC のサーバーとの通信を行った。RTC と TECS を連携して実際に動かしたことは評価できるものの、改善点も見つかった。

TECS RPC の機能を十分に実現することが出来なかった。TECS RPC は TDR（TECS Data Representation）があり、データを伝送する役割とエンディアンと int 型のサイズの違いをチェックする機能を持っている。今回は TDR のデータを伝送する機能を実装したものの、チェック機能の実装までには至らなかった。また、TECS オペイク RPC のクライアント側にセマフォオプションがあるが RTC に実装するのは容易ではなく、今回は実装できなかったため今後の課題である。TECS 側の改善点としては TECS RPC 構造を把握するのにソースコードレベルの構造を把握し辛いことがある。その理由は TECS はエラー処理、パラメータの明確性などのために多層の関数のラッピングを行っているためである。また、RTC はコネクションがきれてもコンポーネントデータ通信が出来きなくなるだけで RTC をエラー状態にするか否かに関しては RTC 開発者が実装するようになっている。一方、TECS オペイク RPC の場合は通信が切れた場合のエラー処理への対応が必要となっている。

OpenRTM-aist と TECS の性能評価結果にも表れたように通常のリナックスシステムに TECS のシステムを導入したことでメモリと CPU の使用量が減ることが分かる。TOPPER と TECS で組込みシステムを構成した場合、省資源、省メモリと組込みシステムでのリアルタイム性、信頼性を向上させることができる。さらに TOPPERS と TECS の組み合わせではターゲットボードが変わっても再利用可能なシステム構築ができる。RTC と TECS を連携させる際に RTC のデータポートを通じて TECS RPC と通信を行い、下位のシステムも TOPPER OS 基盤の組込みシステムに実装を行う予定である。

文献

[1] Noriaki ANDO et al., “RT-Middleware: Distributed Component Middleware for RT (Robot Technology)“,2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2005), pp.3555-3560,2005

[2] TOPPERS プロジェクト Web page, http://www.toppers.jp/index.html

[3] 安藤他, “OpenRTM-aist と TECS(TOPPERS Embedded Component System) の連携に関する考察 “, 日本機械学会ロボティクス・メカトロニスク講演会 2010,2P1-B04,2010

[4] TOPPERS/TECS Web page, http://www.toppers.jp/tecs.html