多様な組込み機器で動作する RTComponent-Lite の開発

○学 鈴木 喬 (首都大), 正 安藤 慶昭 (産総研), 学 稲垣 学 (芝浦工大),
学 大原 賢一 (筑波大), 正 大場 光太郎 (産総研), 正 谷江 和雄 (首都大)

A Development of RTComponent-Lite for Multiple Embedded Devices

*T.SUZUKI (Tokyo Met. Univ.),
N.ANDO (National Institute of Advanced Industrial Science and Technology(AIST)),
M.INAGAKI(Shibaura Inst. of Tech.), K.OHARA (Tsukuba Univ.),
K.OHBA (National Institute of Advanced Industrial Science and Technology(AIST)),
K.TANIE(Tokyo Met. Univ.)

Abstract— In our ubiquitous robot systems, which have various sensors, actuators and robots, various embedded devices are distributed and networked. These sensors and actuators are equipped with various types of embedded CPUs such as Microchip's PIC and Renesas's H8 and so on. In such system, more light-weight middleware is needed. In this paper, we propose a new framework of light-weight RT-Component, which is named "RT-Component-Lite", for various types of embedded devices. This framework is characterized by template-code generator, which supports same style programming for various types of CPUs. Finally, we will verify interoperability between RT-Component-Lite based components and CORBA based RT-Components.

Key Words: RT(Robot Technology), middleware, embedded device, ubiquitous robot, RT Middleware

1. はじめに

我々はセンサやコンピュータ、アクチュエータといったロボットを構成する機能要素が分散配置された空間をユビキタス・ロボットとして提案してきた [1]。このようなユビキタス・ロボットにおいては、多数のセンサやアクチュエータ、ロボット等が分散配置される。

我々の開発した空間機能モジュール（Ubiquitous Functions Activation Module(UFAM)） [2] やセンサネットワークにおける無線センサモジュール U-cube [3] 等は CPU に Microchip PIC を用いており、また Mica Mote などは MICA を用いている。また、ロボットの制御にはルネサス H8 マイコンも多く用いられ、多種多様な組込み CPU が空間に分散されることが考えられる。

RT ミドルウエアは、RT (Robot Technology) 要素のソフトウエアモジュール化を容易にし、システムインテグレーションを効率的に行うための基本機能を備えたソフトウエアプラットフォームである [4]。従来の CORBA 上に実装された RT ミドルウエアでは、空間に分散配置されたリソースの乏しいこれらの小型デバイスに適応することは不可能であった。

本稿では、多様な組込み機器を RT ミドルウエアネットワークに参加させる方法として RTComponent-Lite (RTC-L) [5] を紹介し、その実装として、ルネサス H8 及び Microchip PIC ベースの組込み機器の RTC-L コンポーネント化を行い、ソースコードレベルでの互換性及びその相互運用性について確認する。

2. RT コンポーネント

RT ミドルウエアは様々なロボット要素（RT コンポーネント）を通信ネットワークを介して組み合わせることが出来る。ここでは、RT コンポーネントのオブジェクトモデル (図 1) について整理する。

2·1 RTComponent

RT コンポーネントの本体であり、基本的なインターフェースおよび入出力を行う InPort/OutPort オブジェクトを 0 個以上持つ。処理を行うコアロジックを持ち、外部または内部からのイベントに応じて内部状態を遷移させる。これをアクティビティと呼び、他とは独立したスレッドに割り当てられ以下に挙げる入出力とは独立に処理が行われる。

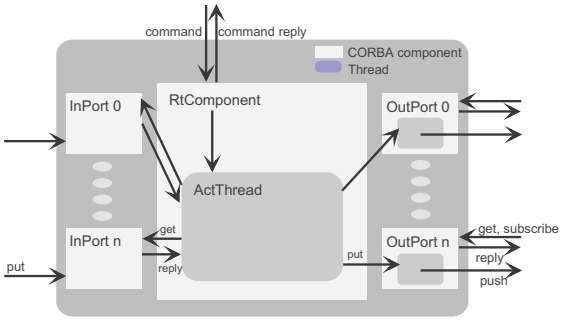


Fig.1 RT コンポーネント.

2·2 InPort オブジェクト

他の RT コンポーネントからの出力をを受け取りハンドリングする入力ポート。データのストリームを受け取る必要がある RT コンポーネントはこの InPort を持つ。

2·3 OutPort オブジェクト

他の RT コンポーネントへ処理結果のデータストリームを渡す出力ポート。受け取る側のコンポーネントから値を取得する pull 型のデータ出力と、サブスクライブすることにより受け取り側へ能動的にデータを送る push 型の動作がある。

2·4 アクティビティ

通常の分散オブジェクトでは、オペレーションに対して何らかの処理を行い、その結果を返すといった動作が一般的である。RT コンポーネントはコンポーネント自体が常に動作し続けるアクティビティを持ち、これがロボット等デバイス制御の主体となる。

さまざまなコンポーネントデベロッパによって作成された多数のコンポーネントを統一的に扱うには、コンポーネントのライフサイクルにおいて共通の状態遷移を持たせると管理が行いやすい。共通の状態遷移を持たせ、その意味を予め規定しておくことにより、多数のコンポーネントの挙動を統一的に制御することが可能となる。

RT コンポーネントのアクティビティは、`BORN`, `INITIALIZE`, `READY`, `STARTING`, `ACTIVE`, `STOPPING`, `ABORTING`, `ERROR`, `EXITING`, `FATALERROR`, `UNKNOWN` の 11 の状態を持つ。図 2 にアクティビティ部の状態遷移図 (UML ステートチャート) を示す。

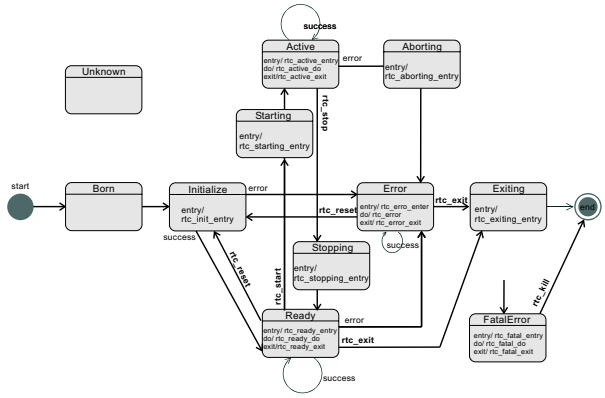


Fig.2 アクティビティの状態遷移図

3. 軽量版コンポーネント: RTC-Lite

多様な小型デバイスを RT ミドルウエアネットワークに参加させるために実装した、軽量版コンポーネント “RTComponent-Lite (RTC-L)” [5] について言及する。

3·1 組込み機器プラットフォーム

ユビキタス・ロボットに於いては、空間にコンピュータやセンサ、アクチュエータを分散させることで空間から人やロボットは情報的かつ物理的なサービスを受けることが出来る。分散配置されるセンサやアクチュエータには small RTUnit(図 3) や空間機能モジュール “Ubiquitous Functions Activation Module (UFAM) ” [2](図 4) のような小型のデバイスが用いられ、空間に分散配置される。こういった小型デバイスは制御マイコンに Microchip PIC やルネサス H8 、AVR などが用いられる場合が多い。そこで、本稿では PIC と H8 をベースとしたプラットフォームに対して、RT ミドルウエアネットワークに参加する方法を提案し、それぞれの互換性、相互運用性を確認する。

small RTUnit は比較的簡単な IO を備え、小型で、ネットワーク接続可能な実験用制御ユニットとして、我々が開発した。入出力制御用マイコンとして PIC 16F877A、ネットワークインターフェースとして Lantronix 社の XPort を搭載したプログラマブルなネットワーク IO デバイスである。

本稿では、PIC ベースプラットフォームとして small RTUnit を用いる。H8 ベースのプラットフォームとして、汎用的な H8/3052F マイコンボードを用いる。

こうしたプラットフォームが RT ミドルウエアにより統合することができ、互換性、相互運用性が確認できれば、様々なプラットフォームが混在するアプリケーションを容易に構築できる。

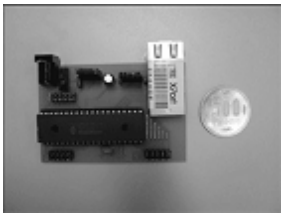


Fig.3 small RTUnit.

Fig.4 UFAM.

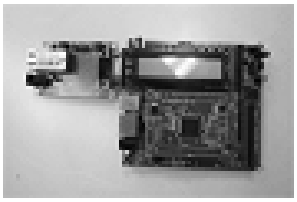


Fig.5 H8/3052.

3·2 軽量プロトコルとブリッジ

前述したようなリソースの乏しいデバイスでは、現在の CORBA に依存した RT ミドルウエアをそのまま実装することはできない。これらを、RT ミドルウエアネットワークに参加させ、管理するための何らかの方法が必要となる。そこで、RT コンポーネントのモデルを、マイコン等のデバイスに実装可能な形式に移植し、それらと、RT ミドルウエアとのブリッジを提供することにより、RT ミドルウエアネットワークに参加させる方法が考えられる。

図 6 に示すように、デバイス上のプログラムと、サーバ上のプロキシコンポーネント間のプロトコルを、RT コンポーネントのオブジェクトモデルに基づきプラットホーム（この場合組込みデバイス）独自のプロトコルで実装しなおすことで、組込みデバイスがあたかも 1 つの RT コンポーネントのように振る舞い、RT ミドルウエアにより統合可能にした。

このように、従来のコンポーネントをプラットフォーム独自のプロトコルで RTC-L との間にブリッジを提供する事で、H8/3052 も small RTUnit(PIC) も同じフレームワークでコンポーネント化することが可能である。

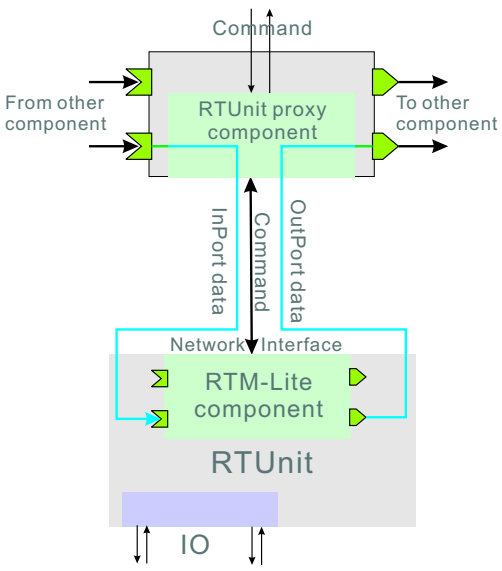
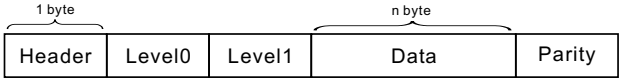


Fig.6 RTC-L コンポーネントと Proxy コンポーネント.



Header	ヘッダ
Level0	上記の各オブジェクトを選択するコマンド
Level1	上記の各オブジェクトのコマンド
Data	引数をマーシャリングしたデータ
Parity	パリティ

Fig.7 RTC-L メッセージ形式

プロキシコンポーネントは、外部からの RTC-L コンポーネントに対する要求を上記のプロトコルに変換し組込みデバイスに対して送る。あるいは、組込みデバイスからのデータを RT ミドルウエア (CORBA) のプロトコルに変換し、他のコンポーネントやクライアントプログラムに対して送る。これにより、CORBA により実装された従来の RT コンポーネントと、異なるプロトコルで実装された簡易コンポーネントが同一のミドルウエアで協調することができる。

RT コンポーネントは同一のオブジェクトモデルをしている為に、CORBA 以外の異なるプロトコルに於いてもこのようなブリッジを提供することが可能である。

3·3 テンプレートコードジェネレータ

RT ミドルウエア実装 “OpenRTM” [6] では、コンポーネントの仕様 (各種プロファイルや InPort/OutPort の型・数) を与えることで、雛形コードを生成するテンプレートジェネレータを提供している。本稿で提案した RTC-L も RT コンポーネントと同様のモデルを採用している為、同等のプロファイル情報で記述できる。

そこで、テンプレートジェネレータの付加機能として、small RTUnit 用のテンプレートジェネレータに加え H8/3052 用テンプレートジェネレータを作成した。図 8 に示すように、同一のプロファイル情報から、PC 上で動作する従来の RT コンポーネントと small RTUnit で動作する RTC-L のコードを生成することができる。その為、外部からコンポーネントとしてアクセスした場合、同様の RT コンポーネントとして扱うことが可能となる。

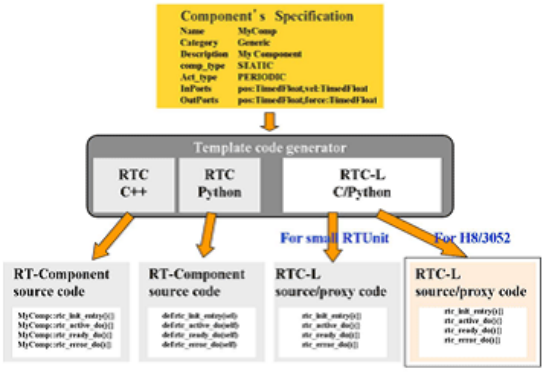


Fig.8 コードジェネレータ.

RT コンポーネントはインタフェース、属性などが次のようなプロファイル情報によって決定することが出来る。

- コンポーネントモジュール名
- コンポーネントカテゴリ名
- コンポーネントタイプ
- コンポーネントのアクティビティのタイプ
- InPort の数、名前、変数型
- OutPort の数、名前、変数型

その為、従来の rtc-template では、バックエンドコードを作成することで、生成するコードの種類を拡張することが可能であった。そこで、バックエンドコードを作成し、バックエンドオプションで PIC、もしくは H8 を与え、以下のような従来のコンポーネントの仕様をコマンドラインオプションで与えることで、雛形ファイルを生成することが可能となった (図 9)。

このような rtc-template のフレームワークを利用することで、さらに様々なデバイスへの移植も容易に行うことが可能である。

```
>rtc-template -bPIC -module-name=Samplemotor ¥
--module-desc=´My simple motor´ ¥
--module-version=0.1 --module-author=MyName ¥
--module-category=motor ¥
--module-comp-type=STATIC --module-act-type=PERIODIC ¥
--module-max-inst=1 ¥
--module-inport=0:hoge:TimedFloatSeq:4 ¥
--module-outport=0:hoge:TimedFloatSeq:4

Samplemotor.c was generated.
Samplemotor.h was generated.
......
```

Fig.9 rtc-template.

3·4 実装の流れ

従来の RT コンポーネントは前述したテンプレートコードジェネレータに、コンポーネントの仕様を与えることにより、雛形を生成する。生成された雛形内の各アクティビティにコードを実装する事で、コンポーネントを作成できる。RTC-L におけるコンポーネントの開発も同様にテンプレートコードジェネレータに仕様を与えることで、雛形を生成する為、同様の開発フローでコンポーネントの実装が可能である。

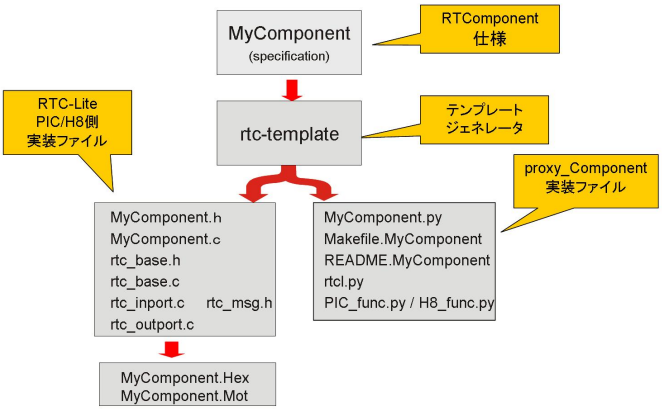


Fig.10 実装の流れ.

また、今回新たに追加した H8/3052 用コードジェネレータを利用する事で、PIC ベースの RTC-L と H8 ベースの RTC-L も同様の開発フローで実装が可能となった。このように、プラットフォーム独自のコードジェネレータを追加することで、ユーザ（この場合コ

ンポーネントデベロッパ）は多様なプラットフォームに対応した RTC-L で実装することが出来きる。

4. モータへの適応

モータはロボットを構成する最も基本的なコンポーネントである。本稿では、多様な組込み機器への RTC-L を用いたコンポーネント化の適応対象として Megarobotics 社のホビー用サーボモータである AI モータを使用する。AI モータはシリアルポートを持ち、コマンドにより簡単な位置制御、速度制御、位置計測、電流計測が行えるモータである。このモータを速度制御モータコンポーネントとして、ルネサス H8 ベースと Microchip PIC ベースの RTC-L で実装する。

4·1 システム構成

我々が提案した RTC-L の開発フローに従って、モータコンポーネントの開発とコンポーネントを組み合て RT コンポーネント接続実験を行う。

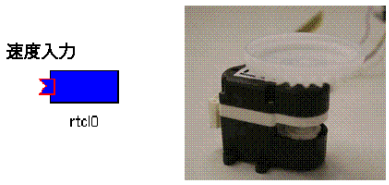


Fig.11 モータコンポーネント RTC-L（左）とモータ（右）.

モータコンポーネント（Motor Component）は、入力ポート（InPort）に対して速度指令値を入力すると、指定された速度で動く。

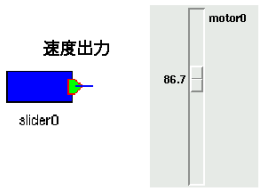


Fig.12 SliderComponen（左）とスライダー GUI（右）.

GUI スライダーコンポーネント（Slider Component）はスライダーの位置に応じた値が出力ポートより出力されるコンポーネントである [7]。

H8 と PIC 双方でモータコンポーネントを実装し、スライダーと接続し、モータを動作させる。

4·2 コンポーネントの実装

それぞれのモータコンポーネントの実装について示す。なお、それぞれのプログラムリストは RT コンポーネントとして共通の部分を除いた固有の実行部分を抜粋したものである。

このように、モータをコントロールする部分のプログラムは異なるプラットフォームに於いても同じような形で記述することが出来る。

プラットフォームに依存する部分だけを書き換えることで、AI モータの動作を記述したプログラムを再利用することが可能である。

```
RtmRes rtc_active_do()
{
// Please implement a task for RTC_ACTIVE do state
#ifdef DEBUG
  ai_motor_ctrl_PIC (g_ActIn[0]);  // AIMotor Control
#endif
  return RTM_OK;
}
```

Fig.13 Main part of the motor component (PIC base).

```
RtmRes rtc_active_do()
{
// Please implement a task for RTC_ACTIVE do state
#ifdef DEBUG
  ai_motor_ctrl_H8 (g_ActIn[0]);  // AIMotor Control
#endif
  return RTM_OK;
}
```

Fig.14 Main part of the motor component (H8 base).

5. おわりに

本稿では、多様な組込みデバイスにおける RTC-L のソースコードレベルでの互換性、再利用性に関して議論した。

簡易プロトコルとブリッジを提供することで、異なるプラットフォームに於いても RT ミドルウエアネットワークに参加することが可能である。このようなフレームワークを利用することで、CORBA 以外のプロトコルであっても、軽量コンポーネントを作成することが可能である。

また、ルネサス H8 ベース及び Microchip PIC ベースの組込みデバイスに対して RTC-L を適用し、ソースコードレベルでの互換性、再利用性に関して示した。

今後は、RTC-L の機能拡張を行うと共に更にリソースの乏しい、UFAM [2] などへの適応も検討を行う。

参考文献

[1] 大原 賢一, 大場 光太郎, 金 奉根, 谷川 民生, 平井 成興, ”ユビキタス・ロボットにおける機能分散に関する検討”, 第 23 回 日本ロボット学会学術講演会予稿集, p.2B21, 2005.09

[2] 大原賢一, 大場光太郎, 金奉根, 谷川民生, 平井成興, 谷江和雄, “空間機能化のための空間機能モジュールの提案”, 日本機械学会 ロボティクス・メカトロニクス講演会 2005, p.2A1N055, 2005.06

[3] 鹿島拓也, 猿渡俊介, 川原圭博, 南正輝, 森川博之, 青山友紀, “センサネットワーク開発用モジュール U3 におけるソフトウエアデザイン及びプロトタイプアプリケーションの実装”, DICOM2003, p.4A, 2003.6.4

[4] 末廣 尚士, 北垣 高成, 神徳 徹雄, 尹 祐根, 安藤 慶昭, ”RT コンポーネントの実装例.RT ミドルウエアの基本機能に関する研究開発 (その 1)”, 第 21 回 日本ロボット学会学術講演会予稿集, p.1F27, 2003.09

[5] 安藤 慶昭, 鈴木 喬, 稲垣 学, 大原 賢一, 大場 光太郎, 谷江 和雄, “組込み機器のための軽量 RT コンポーネント:RTComponent-Lite”, 計測自動制御学会 システムインテグレーション部門 講演会 2005 (SI2005), pp.969-970, 2005.12

[6] OpenRTM：http://www.is.aist.go.jp/rt/OpenRTM-aist-Tutorial/

[7] 安藤 慶昭, 末廣 尚士, 北垣 高成, 神徳 徹雄, 尹 祐根, “RT 要素のモジュール化および RT コンポーネントの実装”, 第 9 回 ロボティクスシンポジア, pp.288-293, 2004.03