The 25th Annual Conference of
the Robotics Society of Japan

RSJ

第25回
日本ロボット学会
学術講演会

講演概要集

会期 ▶ 2007年9月13～15日
会場 ▶ 千葉工業大学津田沼キャンパス

主催 ▶ (社)日本ロボット学会

RT コンポーネントのリアルタイム化を実現する実行コンテキストの拡張

○安藤慶昭 (産総研)　清水昌幸 (産総研)　尹祐根 (産総研)　神徳徹雄 (産総研)

Extended ExecutionContext for Real-time Execution of RT-Components

*Noriaki ANDO (AIST), Masayuki SHIMIZU (AIST),
Woo-Keun YOON (AIST), Tetsuo KOTOKU (AIST)

Abstract—This paper shows how to realize real-time execution of RT-Components's main logic by extension of ExecutionContexts. The ExecutionContexts that is logical thread object executes the RT-Component's business logic of so-called "core logic". To realize real-time executable RT-Component, an ExecutionContext is extended as a real-time thread object. Since the binding between ExecutionContexts and RT-Components are made dynamically, we can change logic execution scheme dynamically. One of another example will be shown and finally conclusion will be mentioned.

Key Words: RT-Component, real-time, execution context

1. はじめに

著者らは、ロボットの機能要素 (RT 機能要素) を RT コンポーネントと呼ぶ構成要素としてモジュール化し、その組み合わせによりロボットシステムを容易に構築するためのプラットフォームとして RT ミドルウエアを提案してきた [1]。

RT コンポーネント開発者は、既存のロボット用ソフトウエアをミドルウエアが提供するフレームワークに埋め込むことで、これまでのソフトウエア資産を容易に RT コンポーネント化し統合することができる。

フレームワークに埋め込まれる RT コンポーネントごとに特有な部分を、コアロジックと呼ぶ。通常のプログラムとは異なり、RT コンポーネントのコアロジックの実行は、スレッドの論理表現である実行コンテキスト (ExecutionContext) と呼ばれるオブジェクトと関連付けられることにより行われる。

本稿では、RT コンポーネントと実行コンテキストの動的関連付け、および実行コンテキストの拡張により、RT コンポーネントのリアルタイム実行や外部トリガによるステップ実行等、多様な実行形態を実現する方法を示す。

2. 実行コンテキスト

通常のプログラムは、起動されるとプロセスのメインスレッドや、プロセス内で生成されたスレッドにより処理が実行される。

これに対して RT コンポーネントでは、スレッドを抽象的に表現した実行コンテキスト (ExecutionContext) と呼ばれるオブジェクトがスレッドに相当する実行主体となる。実行コンテキストは実行周期や状態を持ち、状態に応じて RT コンポーネントのコアロジックに実装されたコールバック関数を呼ぶことで、主たる処理 (例えばロボットの制御等) を行う。

2·1 実行コンテキストの動的関連付け

RT コンポーネントが生成されると、通常一つの実行コンテキストが RT コンポーネントに関連付けられ、コアロジックの実行が開始される。RT ミドルウエアでは、図 1 に示すように、動的リンク可能なモジュールとして作成された複数種類の実行コンテキストを、コンポーネント生成時に選択的に関連付ける (バインディングする) ことができる。

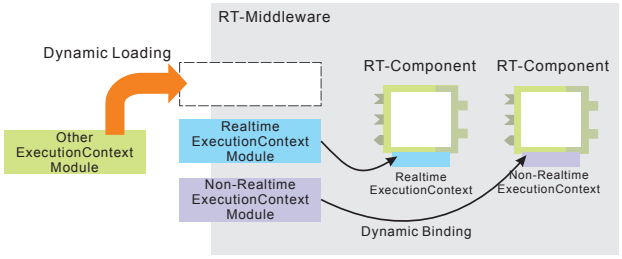

Fig.1 Dynamic Loading and Binding of Execution-Context.

2·2 多様な実行形態の実現

例えば、リアルタイム実行がサポートされない通常の Linux 上でコンパイルされた RT コンポーネントを、ARTLINUX のようなリアルタイム実行をサポートする環境で実行する際に、実行コンテキストモジュールの切り替えを行うだけで、リアルタイム実行を行うことが可能である。動的リンク機構を用いているため、RT ミドルウエアおよび RT コンポーネント共に、再コンパイルすることなくこの機能を利用することができる。

3. リアルタイム実行の実現

ARTLINUX は `art_enter()`, `art_wait()` 等の簡単なシステムコール呼び出しにより、通常のプロセスをリアルタイム化可能な Linux のリアルタイム化カーネルである。

3·1 実行コンテキストの拡張

ExecutionContext のロジック実行を行うスレッド部分を拡張し、ARTLINUX によりリアルタイム実行するための ExecutionContext 実装した。ソースコードの一部を Figure 2 に示す。

Fig.2 の `invoke_worker()` により ExecutionContext に関連付けられた RT コンポーネントのコアロジック

```
int ArtExecutionContext::svc(void)
{//リアルタイムスレッドの開始
 art_enter(ART_PRIO_MAX-1,
           ART_TASK_PERIODIC, m_usec);
 do
  { // コンポーネントの Action の実行
    std::for_each(m_comps.begin(),
                  m_comps.end(),
                  invoke_worker());
    art_wait();
  } while (m_running);

  art_exit();
  return 0;
}
```

Fig.2 Real-Time ExecutonContext for ARTLINUX.

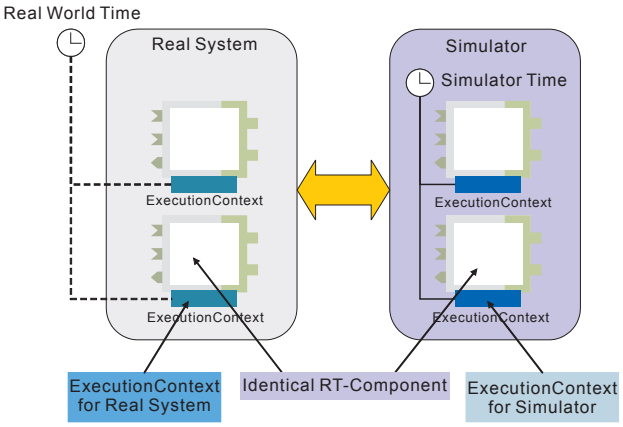

Fig.3 Extended ExecutionContext for Simulator.

が実行される。

リアルタイム化した ExecutionContext を実装し、文献 [2] において用いたハプティックインターフェースおよび HRP2 それぞれ、1ms および 5ms のリアルタイム周期制御に用いた。設定した周期に対する実行周期の平均、最大・最小周期および標準偏差を表 1 に示す。スレッドを ExecutionContext として外部化することによるオーバーヘッドはほとんど観測されなかった。

Table 1 リアルタイム ExecutionContext による実行周期

設定周期 [ms]	平均 [ms]	最大/最小 [ms]	標準偏差 [ms]
1.00	1.00	1.03/0.98	6.74×10^{-3}
5.00	5.00	5.02/4.98	6.14×10^{-3}

上述したように、RT コンポーネントの実行は ExecutionContext に集約されており、ExecutionContext を ARTLINUX の機能を利用しリアルタイムスレッド化することで、RT コンポーネントの実装に手を加えることなくリアルタイム実行を実現することが可能である。

4. 外部トリガ実行の実現

こうした実行コンテキストの拡張による実行制御手法は、シミュレータ等への応用が考えられる。

現実のシステムでは、各コンポーネントにおける時間の進み方は、現実の時間の進み方と同じであるが、シミュレータにおいては、シミュレータを構成する全てのコンポーネントはシミュレータが管理する時間 (=シミュレータ時間) で動作する必要がある。そこで、Figure 3 に示すように、同一のコンポーネントで現実時間での動作とシミュレータ時間での動作をシームレスに実現するための実行コンテキストの拡張を行った。

4·1 実行コンテキストの拡張

Figure 4 に示す CORBA IDL (Interface Definition Language) 定義のように、現実時間で動作させる際に使用する実行コンテキストの標準インターフェースを、シミュレータ用に拡張し、シミュレータが設定する刻み幅で時刻を進める為のオペレーション (`tick()`) を追加した実行コンテキストを作成した。

現実の時間刻みでロジックを駆動する代わりに、シミュレータが供給する外部トリガ (=`tick()` オペレーション呼び出し) により、関係する RT コンポーネントのロジックが駆動される。これにより、同一の RT コンポーネントを内部の変更や再コンパイルを行うこと

```
interface ExtTrigExecutionContextService
 : ExecutionContextService
{
  void tick();
};
```

Fig.4 Extended Interface of ExecutionContext

なく、実システムとシミュレータシステムの両方に利用することができる。

5. まとめ

本稿では、RT コンポーネントの実行主体である、実行コンテキストに対して拡張を行うことで、RT コンポーネントのリアルタイム実行や、外部トリガによる実行等、多様な実行形態を実現する方法を示した。RT コンポーネントと実行コンテキストの関連付けは動的に行うことが可能であり、実行コンテキストの動的ライブラリを作成することで、実行時に実行コンテキストと RT コンポーネントの動的関連付けによる実行形態の変更を実現することができる。

実行コンテキストと RT コンポーネントは多対多の関連を持つことが可能である。今後は、こういった機能を利用した、コンポーネントの複合化や、一つの RT コンポーネントに対して、実行周期の異なる複数の実行コンテキストを関連付けたマルチレート実行等の多様な実行形態を実現していく。

謝辞

本研究は、平成 19 年度 NEDO 次世代ロボット共通基盤開発の一環として実施したものである。ここに感謝の意を表する。

参考文献

[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] 清水他, “タスクスキルトランスファー手法におけるスキルの再利用性の検証実験”, 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 2006 (SI2006), pp.1287-1288, 2006.12