Vol.2015-SLDM-170 No.36
Vol.2015-EMB-36 No.36
2015/3/7

マルチコア並列処理・分散処理統合機能を有するリアルタイムＯＳ

横山 幸太郎[1,a)]　齊藤 政典[1,†1]　兪 明連[1,b)]　横山 孝典[1,c)]

概要：本論文では，マルチコア並列処理と分散処理が混在する環境においてタスクを統一的に管理できる統合リアルタイム OS を提案する．本統合リアルタイム OS は，OSEK OS 仕様を拡張し，タスク管理およびイベント制御のためのシステムコールに位置透過性を持たせる．これにより、自 CPU 上のタスクに対してと同様に，他 CPU コア上や他ノード上のタスクに対しても同一 API を用いて，タスク管理およびイベント制御が可能になる．また，開発した統合リアルタイム OS のシステムコールの実行時間を測定し，実用上問題のないことを確認した．

キーワード：リアルタイム OS，組込みシステム，分散処理，並列処理，マルチコアプロセッサ

A Real-Time Operating System for Multicore Parallel Processing and Distributed Computing

Kotaro Yokoyama[1,a)]　Masanori Saito[1,†1]　Myungryun Yoo[1,b)]　Takanori Yokoyama[1,c)]

Abstract: The paper presents an integrated real-time operating system for multicore parallel processing and distributed computing. The integrated real-time operating system is an extension to OSEK OS and provides location-transparent system calls for task management and event control. By using the integrated real-time operating system, we can develop application programs with the same APIs not only for tasks on the same CPU but also for tasks on other nodes or other CPUs. We have evaluated the performance of the integrated real-time operating system and have confirmed that the performance is practical for embedded control systems.

Keywords: real-time operating system, embedded systems, distributed computing, parallel processing, multicore processor

1. はじめに

自動車制御等の組込み制御分野では，複数の組込みコンピュータをネットワークでつないだ分散型の組込み制御システムが用いられている．例えば自動車制御分野では，複数の ECU（Electoronic Control Unit）をネットワーク接続した分散制御システム構成が広く用いられている．また，組み込み制御システムのアプリケーションの多くはタスク単位で実装されておりリアルタイムオペレーティングシステム（Real-Time Operating System，RTOS）の機能を用いてタスクの管理を行うのが一般的である．

近年，マルチコアプロセッサが注目され，特に組込みステムでは複数の CPU コア上で独立に OS を動作させる非対称型あるいは機能分散型と呼ばれるマルチコア並列処理が有用視されている．また組込み制御分野で複数のコンピュータをネットワーク接続した分散制御システムが広く

[1] 東京都市大学
Tokyo City University
[†1] 現在，日立情報通信エンジニアリング株式会社
Presently with Hitachi Information & Communication Engineering, Ltd.
[a)] g1381527@tcu.ac.jp
[b)] yoo@cs.tcu.ac.jp
[c)] yokoyama@cs.tcu.ac.jp

用いられている．そのためマルチコア並列処理と分散処理の両者に対応可能な RTOS が求められている．

機能分散マルチプロセッサ向けの RTOS として，μITRON4.0 仕様 [1] を拡張した TOPPERS/FDMP カーネル [2] がある．この RTOS ではプロセッサ間でシステムコールを実行可能としており，異なるプロセッサ間でタスクの起動やタスク間の同期が可能である．また，自動車分野の業界標準仕様の策定を行っている AUTOSAR も，マルチコアプロセッサ向けの RTOS の仕様を定めている [3]．

一方，分散制御システム向けの RTOS として，我々が開発した分散 RTOS[4] がある．この分散 RTOS は OSEK OS 仕様 [5] に基づく TOPPERS/OSEK カーネル [6] に分散処理機能を追加し，異なるノード上にあるタスクを対象とした位置透過性のあるシステムコールを提供している．これにより，異なるノード間でタスク管理やイベント制御が可能になる．

ところが，マルチコア並列処理と分散処理が混在する環境において，異なるノード上及び CPU コア上のタスクを，同一 API を用いて統合的に管理できる RTOS は提案されていない．そこで本論文では，それらが混在する環境において統一的なタスク管理を行える RTOS を提案する．これを本論文では統合 RTOS と呼ぶことにする．本統合 RTOS は，同一ノード内の他の CPU コア上や他ノードの CPU コア上のタスクを対象に，同一 API を用いてタスク管理およびイベント制御を可能とする位置透過性のあるシステムコールを提供する．

本論文の構成は以下の通りである．まず 2 章で提案する統合 RTOS の仕様について述べ，3 章で統合 RTOS の機能と構成について述べる．そして，4 章で実装した統合 RTOS の性能評価を行い，5 章で本論文のまとめと今後の課題について述べる．

2. 統合 RTOS の仕様

2.1 拡張対象システムコール

本研究では OSEK OS 仕様のリアルタイム OS である TOPPERS/ATK1 カーネルを対象に，自 CPU 上のタスクのみでなく，マルチコアプロセッサにおける他 CPU コア上のタスクや，分散システムにおける他ノード上のタスクについて，同一のシステムコールを用いて管理できる統合 RTOS を実現する．開発にあたっては，既に開発した分散 RTOS[4] との互換性を保つようにする．本統合 RTOS により，対象タスクがどの CPU コア上あるいはノード上にあるかを意識せずにアプリケーションを記述できるようになる．

OSEK OS が備えるシステムコールには，タスク管理，イベント管理，アラーム管理，リソース管理，割り込み制御，OS の実行管理がある．このうち，タスクを直接対象とするシステムコールはタスク管理とイベント管理に関するもののみで，それらの一覧を表 1 に示す．タスク指定という欄は，システムコールの引数で対象タスクを指定する必要があるかどうかを示しており，その必要がある 5 つのシステムコール，ActivateTask()，ChainTask()，GetTaskState()，SetEvent()，GetEvent() を拡張し，他 CPU コア上及び他ノード上のタスクも指定可能とする．以下，他 CPU コア上のタスクに対するシステムコールをコア間遠隔システムコール，他ノード上のタスクに対するシステムコールをノード間遠隔システムコールと呼ぶ．

表 1　タスク管理とイベント管理のシステムコール

分類	システムコール名 (引数)	タスク指定
タスク管理	ActivateTask(Task)	○
	TerminateTask()	×
	ChainTask(Task)	○
	Schedule()	×
	GetTaskID(TaskIDRef)	×
	GetTaskState(Task, status*)	○
イベント管理	SetEvent(Task, Event)	○
	ClearEvent(Event)	×
	GetEvent(Event)	○
	WaitEvent(Task, EventMask*)	×

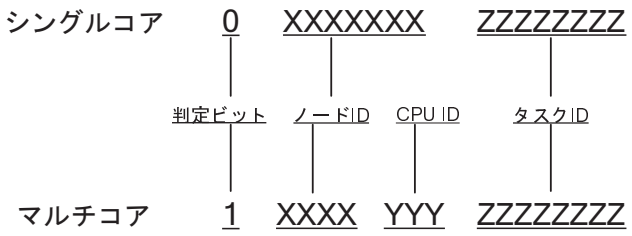

図 1　グローバルタスク ID の割り当て

2.2 グローバルタスク ID

本統合 RTOS では，システム全体でタスクを一意に指定可能とするため，全 CPU コア・全ノードで共通のグローバルタスク ID を定義する．グローバルタスク ID は分散 RTOS との互換性を持たせるため**図 1** に示すような 2Byte の値とする．上位 1Byte はノードあるいは CPU コアを示し，下位 1Byte で OSEK OS 仕様と同様のタスク ID を表す．

上位 1Byte のうち最上位ビットでシングルコアかマルチコアを表す判定ビットを，残り 7bit でノード ID と CPU ID を表す．現時点での実用的なマルチプロセッサの CPU コア数は 4 以下であり，今後も 8 程度以下と予測する．また自動車制御システムの場合，高性能なマルチコアプロセッサが要求されるパワートレイン系システムのノード数は 16 以下である．そこで，マルチコアの場合は上位 3bit でノード ID，下位 4bit で CPU ID を表すこととした．シングルコアの場合は 7bit 全てでノード ID を表す．

```
<file> ::=
    <OIL_version>
    <implementation_definition>
    <application_node_definition_list>
. . . . . .
<application_node_definition_list> ::=
    <application_node_definition>
    | <application_node_definition_list> <application_node_definition>
<application_node_definition> ::=
    <application_multi_definition> | <application_definition>
<application_multi_definition> ::=
    "MULTI" <name> "{" <application_definition_list> "}" <description> ";"
<applicarion_definition_list> ::=
    <applicarion_definition>
    | <application_definition_list> <application_definition>
<applicarion_definition> ::=
    "CPU" <name> "{" <object_definition_list> "}" <description> ";"
. . . . . .
```

図 2 拡張 OIL の構文

```
OIL_VERSION = "2.5";
IMPLEMENTATION Standard {
. . . . . .
}
MULTI multi1 {
  CPU cpu11 {
    . . . . . .
    TASK task111 { . . . . . };
    TASK task112 { . . . . . };
    . . . . . .
  };
  CPU cpu12 {
    . . . . . .
    TASK task121 { . . . . . };
    TASK task122 { . . . . . };
    . . . . . .
  };
  . . . . . .
};
```

```
MULTI multi2 {
  CPU cpu21 {
    . . . . . .
  };
  . . . . . .
};
CPU cpu3 {
    . . . . . .
    TASK task31 { . . . . . }
    TASK task32 { . . . . . }
    . . . . . .
};
CPU cpu4 {
    . . . . . .
    TASK task41 { . . . . . }
    TASK task42 { . . . . . }
    . . . . . .
};
. . . . . .
```

図 3 拡張 OIL の記述例

2.3 拡張 OIL

OSEK OS では，アプリケーションの設定（コンフィギュレーション）のために OIL（OSEK Implementation Language）[9] と呼ばれる専用言語を用いる．OIL によりタスクの宣言やイベントの設定などを記述し，SG（System Generator）に通すことでアプリケーション依存の構成データを記述したソースコードを出力する．本研究では，分散 RTOS 向けに 1 つのファイルに全ノードのアプリケーションの設定を記述できるように OIL を拡張する．

図 2 は，拡張部分の構文を BNF 記法で表記したものである．<application_definition>が拡張前から定義されている 1CPU 分のアプリケーションの設定情報を表す．我々は，<application_multi_definition>がマルチコアプロセッサ内の複数の CPU コアのアプリケーションの設定情報を表し，<application_node_definition_list>が分散システムの全ノードのアプリケーションの設定情報を表すように拡張した．

次に，拡張 OIL の記述例を**図 3** に示す．図 3 は，マルチコアプロセッサ multi1 と multi2，シングルコアプロセッサ cpu3 と cpu4 の 4 ノード分の設定情報を記述した拡張 OIL の例である．

統合 RTOS の構成データには，TOPPERS/OSEK カーネルがもともと必要としていた構成データに加え，統合 RTOS で追加したマルチコア並列処理および分散処理機能向けの構成データがある．これらの構成データを生成できる SG は現在開発中であるため，現時点では OIL 記述を参照して，手作業で構成データのソースファイルを記述する．

3. 統合 RTOS の機能と構成

3.1 方針

OSEK OS に基づく TOPPERS/ATK1[6] にマルチコア並列処理と分散処理のための機能を追加することで統合 RTOS を実現する．本研究で対象とするのは共有メモリを有するマルチコアプロセッサとし，コア間遠隔システムコールは共有メモリ経由の通信で実現する．

ネットワークとしては CAN[7] を使用する．我々が既に発表している分散 RTOS はネットワークとして FlexRay[8] を使用していたが，FlexRay コントローラを搭載したマルチコアプロセッサが入手できなかったため，CAN を用いることとした．このため本統合 RTOS では，ノード間の時刻同期機能は提供しない．CAN に対応しているプロセッサの多くは複数の CAN コントローラを搭載している．そこで我々は，CPU コア数と同一あるいはそれ以上の CAN コントローラを搭載したマルチコアプロセッサの場合には，ひとつの CPU コアにひとつの CAN コントローラを占有させ，全ての CPU コアが直接ネットワーク通信を行うことで，ノード間遠隔システムコールを実装する．CPU コア数より少ない CAN コントローラを搭載したプロセッサの場合は，ひとつの CPU コアのみでネットワーク通信を行うものとし，他の CPU コアは共有メモリ経由通信とネットワーク通信を組み合わせてノード間遠隔システムコールを実装する．

実装に用いるハードウェアはルネサスのデュアルコアマイクロプロセッサ SH7205 を搭載した評価ボード M3A-HS50G50 である．SH7205 は CAN コントローラを 2 つ搭載しているため，各 CPU コアが直接ネットワーク通信を行うことでノード間遠隔システムコールを実装する．

3.2 構成

統合 RTOS の構成を**図 4** に示す．統合 RTOS は，タスク位置判定機能，コア間遠隔システムコール機能，ノード間遠隔システムコール機能を備える．ノード間遠隔システムコール機能については分散 RTOS の機能を用いる．タスク位置判定機能は分散 RTOS が提供しているタスク位置判定機能を他 CPU コアを対象にする場合も判定が行えるように拡張する。コア間遠隔システムコールについては新規に開発を行う．またグローバルタスク ID などの情報は統合 RTOS 構成データとして保持される．

コア間遠隔システムコール機能は，メモリ経由要求送信処理，メモリ経由要求受信処理，メモリ経由返戻値送信処理，メモリ経由リターン処理から成る．ノード間遠隔システムコール機能はネットワーク経由要求送信処理，ネットワーク経由要求受信処理，ネットワーク経由返戻値送信処理，ネットワーク経由リターン処理から成る．

Vol.2015-SLDM-170 No.36
Vol.2015-EMB-36 No.36
2015/3/7

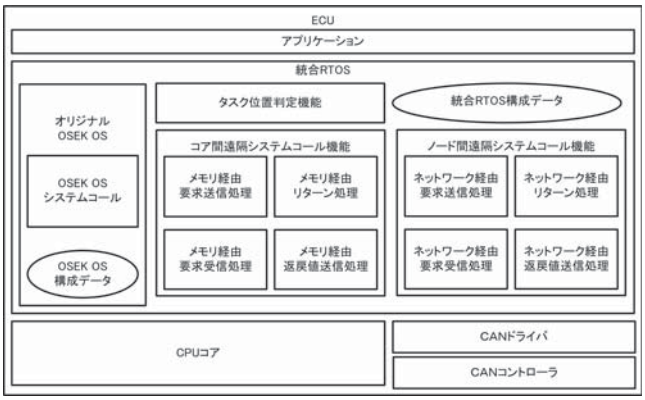

図 4　統合 RTOS の構成

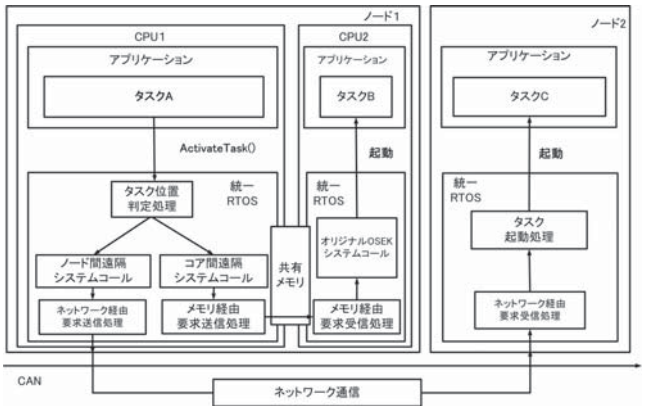

図 5　遠隔システムコールの動作

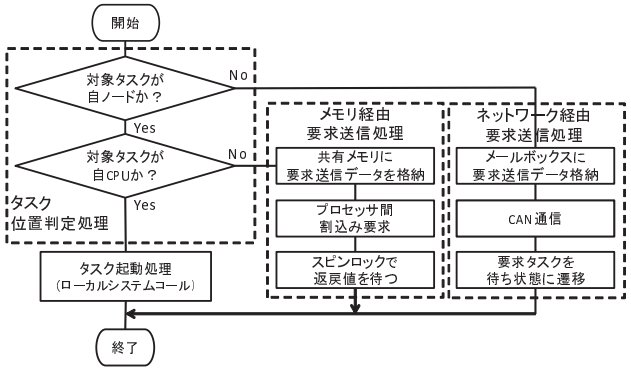

図 6　タスク位置判定・要求送信

3.3　遠隔システムコール

本統合 RTOS におけるコア間及びノード間遠隔システムコールの処理の流れを**図 5** に，遠隔システムコール内の各処理を**図 6**，**図 7** を示す．アプリケーションタスクがシステムコールを発行すると，タスク位置判定機能が，システムコールの対象タスクがどのノードのどの CPU コア上にあるか判定する．

対象タスクが同一ノード上の他 CPU コア上にある場合，図 6 に示すように，コア間遠隔システムコール機能のメモリ経由要求送信処理が，コア間要求メッセージを生成し，共有メモリ上のメッセージバッファに格納する．コア間要求メッセージは**図 8** に示すように，発行元 CPU ID，送信

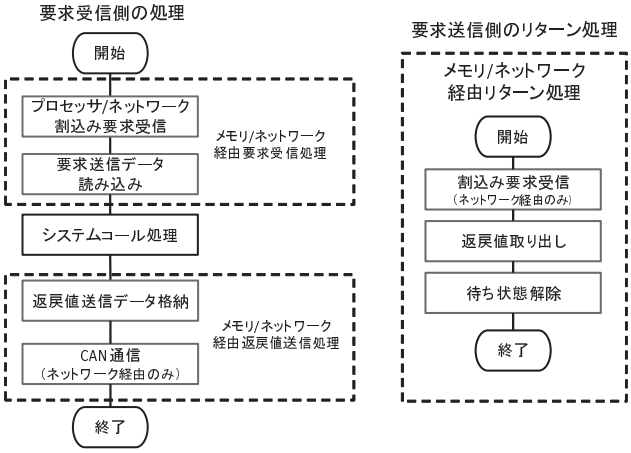

図 7　要求受信・返戻値送信・リターン処理

先 CPU ID，システムコール発行元タスク ID，対象タスク ID，システムコールの種類，システムコールの引数で構成する．そして，対象タスクのある CPU に対して割り込みを発行した後，システムコール発行元タスクはスピンロックにより返戻値を待つ．

割り込みを受けた CPU では，図 7 左に示すように，メモリ経由要求受信処理が共有メモリ上のコア間要求メッセージを読み出して解析し，指定されたシステムコールを実行する．システムコール実行後，メモリ経由返戻値送信処理が返戻値を含むコア間返答メッセージを共有メモリに格納する．

返戻値が共有メモリに格納されると，図 7 右に示すように，メモリ経由リターン処理が返答メッセージを解析し，返戻値をバッファに書き込む．そして，システムコール発行元タスクはスピンロックから抜け，バッファから返戻値を読み出し処理を再開する．

対象タスクが他ノード上にある場合，図 6 に示すように，ノード間遠隔システムコール機能のネットワーク経由要求送信処理がノード間要求メッセージを生成し，CAN ドライバにメッセージの送信要求を出し，システムコール発行元タスクを待ち状態に遷移させる．ノード間要求メッセージは**図 9** に示すように，CAN の ID 部分にメッセージ判定ビット，送信先 CPU ID およびノード ID を格納し，データフィールドに発行元ノード ID，発行元タスク ID，対象タスク ID，システムコールの種類，システムコールの引数で構成する．そして，CAN 通信によって要求送信後，システムコール発行元タスクを待ち状態に遷移させる．

対象タスクのある送信先ノードがメッセージを受信すると，図 7 左に示すように，ネットワーク経由要求受信処理によりノード間要求メッセージを解析し，指定されたシステムコールを実行する．その後，ネットワーク経由返戻値送信処理が発行元ノードにノード間返答メッセージを送信する．

メッセージを受信した発行元ノードでは，図 7 右に示す

図 8 コア間遠隔システムコール送信データ

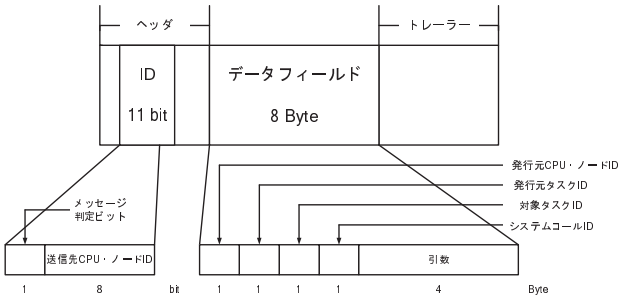

図 9 ノード間遠隔システムコール送信データ

ように，ネットワーク経由リターン処理が返答メッセージを解析し，返戻値をバッファに書き込み，システムコール発行元タスクの待ち状態を解除する．

4. 評価

本研究で実装した統合 RTOS とオリジナルの TOPPERS/ATK1 カーネルについて，拡張対象の 5 つのシステムコールの実行時間を測定した．デュアルコアプロセッサ SH7205 の CPU コアのクロック数は 200MHz である．

表 2 に統合 RTOS のシステムコールのコア間遠隔システムコールとローカルシステムコール，TOPPERS/ATK1 のシステムコールの実行時間を示す．拡張を行ったシステムコールはオリジナルのシステムコールと比較してローカルで 0.2〜0.4μsec のオーバーヘッド，コア間遠隔システムコールでは 3μsec のオーバーヘッドとなっている．

表 3 にノード間遠隔システムコール中の 4 つの処理の実行時間を示す．ノード間遠隔システムコールの応答時間には，これらの他，CAN の通信時間が加わる．転送速度 500kbps の場合，1 回の CAN 通信には 0.2msec 程度かかるが，それに比較してノード間遠隔システムコールの CPU 実行時間は十分に小さい値となっている．CAN 通信時間を含めたノード間遠隔システムコールの応答時間は 0.5msec 程度となるが，自動車制御アプリケーションの制御周期は 10msec から 100msec 程度であるため，実用上は問題ない値であると考えている．

表 2 システムコールの実行時間

システムコール	コア間遠隔	ローカル	TOPPERS/ATK1
ActivateTask()	5.79	2.43	1.59
ChainTask()	6.09	2.49	1.68
GetTaskState()	3.81	1.17	0.69
GetEvent()	4.14	1.17	0.78
SetEvent()	4.68	1.26	0.78

[μsec]

表 3 ノード間遠隔システムコール用処理の実行時間

システムコール	要求送信	要求受信	返戻値送信	リターン処理
ActivateTask()	1.98	1.14	0.99	2.25
ChainTask()	1.98	1.14	0.99	2.25
GetTaskState()	1.98	1.14	0.99	2.25
GetEvent()	1.98	1.14	0.99	2.25
SetEvent()	1.98	1.14	0.99	2.25

[μsec]

5. おわりに

分散処理環境とマルチコア並列処理環境が混在する環境において統一的なタスク管理が可能な，組み込み制御システム向けの統合 RTOS を提案した．そして，実装及び評価を行い，実用上問題ない値であることを確認した．

現在，統合 RTOS に対応した構成データの生成が可能な SG を開発中である．CAN コントローラ数が CPU コア数よりも少ないマルチコアプロセッサを対象とするため，メモリ経由通信とネットワーク経由通信を組み合わせてノード間遠隔システムコールを実現する機構も開発している．また，CPU コア間での排他制御機能を提供するため，OSEK OS のリソース管理のためのシステムコールを拡張している．我々はまた，分散共有メモリ機能を有する分散 RTOS も開発している [10]．本論文で提案した統合 RTOS にこれらの機能を統合化することで，単一 CPU による処理，マルチコア並列処理，および分散処理を意識せずに組込み制御システムを開発可能な RTOS を実現したいと考えている．

謝辞

本研究で使用した TOPPERS/ATK1 の開発者に感謝する．本研究の一部は JSPS 科研費 24500046 の助成を受けたものである．

参考文献

[1] 坂村健監修，高田広章編:μ ITRON4.0 仕様 Ver.4.02.00 トロン協会 (2004).
[2] 本田晋也，高田広章:ITRON 仕様 OS の機能分散マルチプロセッサ拡張，電子情報通信学会論文誌，Vol.J91-D, No.4，pp.934-944 (2008).
[3] AUTOSAR: *Specification of Multi-Core OS Architec-*

ture V1.1.0 R4.0 Rev 2 (2010).
[4] 知場貴洋，齊藤政典，伊丹悠一，兪明連，横山孝典：位置等価性のあるシステムコールを有する組み込み制御システム向け分散リアルタイムＯＳ，情報処理学会論文誌，Vol.53，No.12，pp.2702–2714 (2012).
[5] OSEK/VDX: *Operating System, Version 2.2.3* (2005).
[6] TOPPERS/OSEK カーネル，TOPPERS/ATK1: http://www.toppers.jp/atk1.html.
[7] Kiencke U.: Controller Area Network — from Cocept to Reality, *Proc. 1st International CAN Conference,* pp.0-11–0-20 (1994).
[8] FlexRay:http://www.exray.com/
[9] OSEK VDX, *System Generation OIL: OSEK Implementation Language Version 2.5* (2004).
[10] Chiba, T., Yoo, M., Yokoyama, T.: A Distributed Real-Time Operating System with Distributed Shared Memory for Embedded Control Systems, *Proc. IEEE 11th International Conference on Dependable, Autonomic and Secure Computing (DASC),* pp.248–255 (2013).