Vol.2013-SLDM-160 No.3
Vol.2013-EMB-28 No.3
2013/3/13

分散共有メモリ機構を持つ組み込み制御システム向け分散リアルタイムOS

知場 貴洋[1,a)]　兪 明連[1,b)]　横山 孝典[1,c)]

概要：本論文では，自動車などの組み込み制御システムを対象にした，分散共有メモリ機構を持つ分散リアルタイム OS について述べる．まず，アプリケーションのタスクが共有メモリに読み書きをするためのメモリアクセス機能と，ネットワーク通信を用いた更新処理によって一貫性を保つ手法を提案する．そして，自動車制御向けのリアルタイム OS である OSEK OS をベースに分散共有メモリ機構を備える分散リアルタイム OS を実装する．ネットワークには，TDMA プロトコルに基づく FlexRay を用いる．

キーワード：リアルタイム OS，分散システム，分散共有メモリ，組み込み制御システム

1. はじめに

自動車制御やファクトリーオートメーションなどの分野では，複数の組み込みコンピュータをネットワークで繋いだ分散型の組み込み制御システムが用いられている．組み込み制御システムの多くはハードリアルタイムシステムであるため，分散型の組み込み制御システムではネットワーク通信も含めたシステム全体で制御処理のデッドラインを守る必要がある．

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，組み合わされる．複数のタスクで構成されるアプリケーションでは，あるタスクが他タスクを起動したりタスク間で同期をとったりする必要がある．また，異なるタスク間で排他制御を行ってデータを共有する場合もある．単一プロセッサ上で動作するアプリケーションの場合は，タスクの起動や同期，タスク間の排他制御はリアルタイムオペレーティングシステム（Real-Time Operating System，RTOS）の機能を用いて行うのが一般的である．分散型の組み込み制御システムでは，アプリケーションのタスクは自ノード内のタスクだけでなく他ノード上のタスクと同期をとったりデータを共有できることが望ましい．そのため，複数のノード上に配置されたタスクを統一的に管理できる分散 RTOS が求められている．

これまでに多くの分散 OS に関する研究がなされており，リアルタイムシステム向けの分散 OS の研究 [1] や，複数のコンピュータ資源を位置透過に扱う分散 OS の提案 [2] がなされている．しかし，分散 OS に関するこれまでの研究の多くは主に汎用コンピュータを用いた分散システムを対象としており，リソース制約の厳しい組み込み制御システム向けの研究はほとんどなされていない．

また，自動車等の組み込み制御アプリケーションは，マルチタスク構成で実装されることが多いが，タスク間のデータのやり取りは共有変数を用いて実装されることが多い．このため，分散型の組み込み制御システムでも共有メモリが使用できればアプリケーションを書き換えずに分散化することが可能になる．分散システム上で仮想的な共有メモリを提供する分散共有メモリが提案がなされてきた [3]．その多くは OS の仮想メモリ機能を利用した，ページベースの分散共有メモリである．しかし，ページベースの分散共有メモリは時間保証が困難でハードリアルタイムシステムに適していない．また，組み込み制御システム向けの RTOS は仮想メモリ機能を有しておらず，仮想メモリ機能を利用した分散共有メモリ機構を実装することは困難である．

我々は，分散処理環境においてノードとタスクの位置関係を意識せずにアプリケーションを記述可能な，組み込み制御システム向けの分散 RTOS に関する研究を行ってきた [4]．我々が提案した分散 RTOS は，自動車制御分野の標準的な RTOS である OSEK OS[5] をベースとして，異なるノード上にあるタスクを対象にできる位置透過性のあるシステムコールを提供している．また，分散 RTOS を搭載する全ノード間で時刻同期を行う機能も備えている．ネッ

[1] 東京都市大学
Tokyo City University, Setagaya, Tokyo 158–8557, Japan
[a)] g1181525@tcu.ac.jp
[b)] yoo@cs.tcu.ac.jp
[c)] yokoyama@cs.tcu.ac.jp

トワークには，自動車制御分野で普及しつつある TDMA（Time Division Multiple Access）プロトコルに基づくリアルタイムネットワーク FlexRay[6] を用いている．同期した時刻に基づいて処理を行うことで，分散システム全体で同一のグローバル時刻に基づくスケジューリングアルゴリズムを採用することも可能になる．また，通信量を考慮して FlexRay 通信のパラメータを設計した場合，ネットワーク通信時間を含んだシステムコールの最悪応答時間は予測可能である．

本分散 RTOS の新たな機能として，異なるノードのアプリケーション間で値を共有可能な組み込み制御アプリケーション向け分散共有メモリ機構を提案する．アプリケーションのタスクが共有データ領域に値を書き込むと，分散 RTOS はネットワーク通信を介して分散共有メモリの更新処理を行う．複数のノードで並行して値が書き込まれた場合も，分散 RTOS は更新処理によってメモリの一貫性を保つ．

本論文では提案する分散共有メモリ機構の機能と構成について述べる．そして，分散共有メモリ機構を持つ分散 RTOS を実装し，組み込み制御システムへの適用可能性について論じる．

2. 分散 RTOS の機能

提案する分散 RTOS は，分散処理向けの機能としてノード間の時刻同期機能，位置透過性のあるシステムコール機能，分散共有メモリ機能を備える．

ノード間の時刻同期機能は，分散 RTOS を搭載した全ノード間で同期されたシステム時刻（グローバル時刻）を提供する．各ノードの FlexRay コントローラは全 FlexRay コントローラ間で同期したネットワーク時刻を持ち，ネットワーク時刻を元に TDMA 方式に基づいた通信を行う．ネットワーク時刻を分散 RTOS に供給することで，全ノードの分散 RTOS がグローバル時刻を持つことが可能となる．

位置透過性のあるシステムコール機能は，対象のタスクがどのノード上にあっても発行可能な位置透過性のあるシステムコールを提供する．本分散 RTOS は，OSEK OS のシステムコールのうち，他ノードのタスクを対象とする可能性があるシステムコールに位置透過性を持たせる。他ノードに対して発行するシステムコールを遠隔システムコールと呼ぶ。開発者は，対象のタスクがどのノード上にある場合も同じシステムコールを用いてアプリケーションを記述できる．遠隔システムコールが発行されたと分散 RTOS が判断した場合，他ノード上のタスクに対する処理のために必要な通信処理を行う．

分散共有メモリ機能は，ネットワーク通信を用いた更新処理によってノード間で一貫性を保つメモリ領域を提供する．これにより，全ノード上のアプリケーション間で値の共有が可能となる．

3. 分散共有メモリの概要

本分散 RTOS は，全ノード上のアプリケーション間でデータを共有可能な分散共有メモリを提供する．複数のタスク間でデータを共有するアプリケーションを記述する場合，開発者はデータを共有するタスクの位置に関わらず，分散共有メモリの読み書きによって処理を記述できる．

全ノードの分散 RTOS は，分散共有メモリのデータのコピーを持つ．各ノードのアプリケーションは，自ノード上のコピーを通じて分散共有メモリのデータを読み書きする．あるノード上のアプリケーションが分散共有メモリのコピーに値を書き込んだ場合，分散 RTOS は通信処理を行って他ノードに分散共有メモリのデータを送信する．分散共有メモリのデータを受け取ったノード上の分散 RTOS は，コピーの値を更新する．

分散共有メモリにアクセスするタスクが複数存在するアプリケーションでは，分散共有メモリを読み書きするために同じアプリケーションのタスク間で排他制御を行う必要がある．そこで，分散 RTOS はアプリケーションのタスク間で排他制御を伴う分散共有メモリアクセス機能を提供する．具体的には，OSEK OS のリソース機能を拡張して分散共有メモリアクセスに対応する．OSEK OS のリソースは排他制御に用いるオブジェクトであり，システムコールの GetResource() を発行する事でリソースを取得し，ReleaseResource() を発行する事で取得したリソースを解放するまで，リソースを取得したタスクや割り込み処理の優先度はそのリソースを共有するタスクや割り込み処理のうち最高優先度がまで一時的に引き上がる．どちらのシステムコールも，引数には取得するリソースの ID を指定する．本研究では GetResource() と ReleaseResource() に分散共有メモリ用の処理を追加し，分散共有メモリ用リソースを取得中に行った読み書き処理のみ一貫性を保つ仕様とした．これにより，同一ノード上のタスク間でリソース機能により排他制御を行い，共有変数にアクセスする場合と同様の形式で共有メモリへのアクセスを記述可能となる．

4. 一貫性モデル

本研究で提案する分散共有メモリのような共有データのコピーを分散配置したシステムでは，複数のノード上のアプリケーションが同時にコピーを読み書きする可能性がある．複数のコピーに対して同時に書き込みが行われた場合，全ての書き込み処理完了後に各ノードで読み出すコピーの値は，特定の一貫性を持った値とならなければならない．

分散データストアの一貫性モデルは．データ中心一貫性モデルとクライアント中心一貫性モデルに大別される [7]．本研究で提案する分散共有メモリには，データ中心一貫性モデルのうちの順序一貫性を採用する．

本研究でネットワークに用いる FlexRay はコミュニケー

ションサイクルと呼ばれる時間単位があり，FlexRay コントローラはこのサイクルごとに静的に設定されたスケジューリングに基づいた通信を行う．FlexRay には周期的なデータ転送を行うための静的セグメントと，要求によりデータ転送を行う動的セグメントがある．

FlexRay コントローラはデータの送信要求を受けると，送信可能なネットワーク時刻まで待機してからデータを送信する．1 コミュニケーションサイクル中に複数のノードで FlexRay コントローラに送信要求が行われた場合，FlexRay コントローラに送信要求を出した順序と実際にデータ送信が完了する順序が異なる場合がある．本研究で提案する分散共有メモリは順序一貫性を維持するため，書き込み処理を行ったノードでは書き込まれた値を即座に分散共有メモリに反映せず，他ノードへのデータ送信が完了した際に反映する．また，受信した分散共有メモリのデータを反映させる処理は，コミュニケーションサイクル開始時に同期した割り込みによって行う．

複数のノードで分散共有メモリに値が書き込まれてから全ノードに値が反映されるまでの流れを**図 1** に示す．コミュニケーションサイクル n の時点では，全ノードで分散共有メモリの値は A である．サイクル n からサイクル n+1 の間にノード 2 とノード 3 で書き込み処理が行われるが，ノード 3 の書き込みデータのみが同じサイクル中に送信を完了し，送信完了割り込みによってノード 3 上の分散共有メモリに反映され，値は C となる．サイクル n+1 開始時にノード 1 とノード 2 で分散共有メモリの受信データが反映され，全ノードで値は C となる．ノード 2 の書き込みデータはサイクル n+1 からサイクル n+2 の間に送信を完了し，送信完了割り込みによってノード 2 上の分散共有メモリに反映され，値は B となる．その後，サイクル n+2 開始時にノード 1 とノード 3 で分散共有メモリに受信データが反映され，全ノードの値は B となる．

また，n+2 サイクルから n+3 サイクルの間にノード 1 とノード 3 で書き込み処理が行われ，どちらの書き込みデータも同じサイクル中に送信を完了する．この場合，n+3 の開始時には，後に送信完了したノード 3 の書き込みデータのみを反映する．そのため，n+3 サイクル開始後に全ノードの値は E となる．

このように，FlexRay によるデータ送信が完了した順に分散共有メモリを更新することで，全ノードが同じ順序で値の変化を観測する事が可能となり，順序一貫性を維持できる．

5. 分散共有メモリの実装

本研究で実装する分散共有メモリにアクセスする処理の記述例を**図 2** に示す．distmem_res は分散共有メモリ用リソースの ID であり，distmem_buf はアプリケーションが分散共有メモリのデータを読み書きする際に用いる変数で

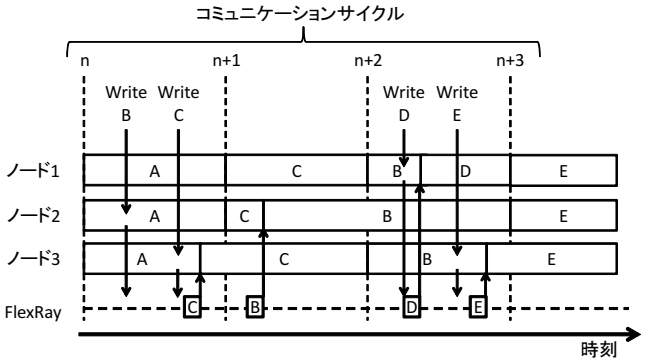

図 1 分散共有メモリの値の遷移
Fig. 1 Time Chart of Distributed Shared Memory.

```
/* 分散共有メモリ用リソース取得 */
GetResource(distmem_res);

/* 分散共有メモリの値を取得 */
value = distmem_buf;

/* 分散共有メモリに値を書き込み */
distmem_buf = 10;

/* 分散共有メモリ用リソース解放 */
ReleaseResource(distmem_res);
```

図 2 分散共有メモリにアクセスする処理の記述例
Fig. 2 Example Description of Distributed Shared Memory Access.

ある．この例では，分散共有メモリのデータは 32bit の値 1 つで，distmem_buf は符号無し 32bit 整数の変数である．OSEK OS では OIL(OSEK Implementation Language)[8] と呼ばれる言語を用いてアプリケーションのタスクに関する情報やリソースに関する情報などを記述し，SG(System Generator) に入力することで OSEK OS の設定情報を記述したソースコードを出力する．共有メモリのためのリソースと変数について OIL で宣言し，SG で生成する予定であるが，現時点では未開発のため，distmem_res と distmem_buf の宣言は分散 RTOS のソースファイルに直接記述している．

アプリケーションは GetResource() を発行して distmem_res を取得することで 4 章で述べた一貫性に基づいた distmem_buf の読み書きが可能になる．アプリケーションが ReleaseResource() を発行して distmem_res を解放すると，分散 RTOS は distmem_buf の値が GetResource() 発行前と異なっているか確認する．GetResource() 発行前と異なっていた場合は書き込みが行われたと判断し，通信処理を行って書き込まれた値を全ノードに送信する．

分散共有メモリの書き込みノード上の処理のタイムチャートを**図 3** に示す．この図は，あるノード上のタスクが分散共有メモリへの書き込み処理を開始してから分散 RTOS が自ノードの分散共有メモリの値を更新するまでの処理の流れを表している．書き込みノード上で行う処理は，分散共

有メモリ用リソース取得処理，分散共有メモリ用リソース解放処理，分散共有メモリ更新処理の 3 つに分けられる．分散共有メモリの値を記憶するには，タスク用変数と分散 RTOS 用バッファの 2 つの記憶領域を用いる．図 2 の distmem_buf はタスク用変数に対応する．

タスクが GetResource() を発行すると，分散 RTOS はシステムコールの対象リソースが分散共有メモリ用リソースであるか判定する．対象リソースが分散共有メモリ用リソースの場合，分散 RTOS は分散共有メモリ取得処理を呼び出し，分散 RTOS 用バッファの値をタスク用変数にコピーする．そして，OSEK OS のリソース取得処理を行う．

タスクが ReleaseResource() を発行すると，分散 RTOS はシステムコールの対象リソースが分散共有メモリ用リソースであるか判定する，対象リソースが分散共有メモリ用リソースの場合，分散 RTOS は分散共有メモリ用リソース解放処理を呼び出す．分散共有メモリ用リソース解放処理では，タスク用変数と分散 RTOS 用バッファの値を比較し，値が異なっていれば FlexRay コントローラに送信要求を出す．分散共有メモリ用リソース解放処理が完了すると，分散 RTOS は OSEK OS のリソース解放処理を行う．

分散共有メモリ更新処理はタスク用変数の値を送信完了した際の割り込みによって呼び出され，送信を完了したタスク用変数の値を分散 RTOS 用バッファにコピーする．

次に，分散共有メモリの読み出しノード上の処理のタイムチャートを**図 4** に示す．この図は，他ノードから分散共有メモリの書き込みデータを受信してから，分散 RTOS が受信したデータを分散共有メモリに反映するまでの処理の流れを表している．また，この図は，分散共有メモリの受信データが存在するコミュニケーションサイクルの開始時に，タスクが分散共有メモリ用リソース取得中の場合の例ある．読み出しノード上で行う処理は，分散共有メモリ用リソース取得処理，サイクル開始処理，受信データ反映処理，分散共有メモリ用リソース解放処理の 4 つに分けられる．また，タスク用変数，分散 RTOS 用バッファの他，FlexRay 用バッファを使用する．

分散 RTOS は，コミュニケーションサイクル開始に同期したサイクル開始処理で受信したデータのチェックを行う．分散共有メモリの受信データが存在した場合，分散 RTOS は受信データ反映処理を呼び出す．受信データ反映処理は，受信データの内容を解読した後，分散共有メモリ用リソースを取得中のタスクが存在するか確認を行う．

分散共有メモリ用リソース取得中のタスクが存在する場合，受信した分散共有メモリ更新用の値を FlexRay 用バッファ内に保持する．タスクが分散共有メモリ用リソースを解放すると，分散共有メモリ用リソース解放処理が呼び出される．分散共有メモリ用リソース解放処理では，分散 RTOS が FlexRay 用バッファ内に保持している更新用の値を分散 RTOS 用バッファにコピーする．

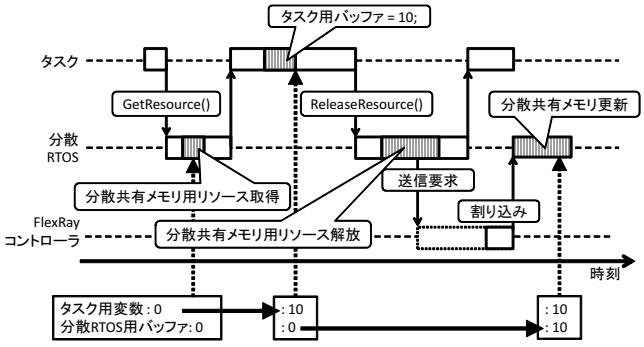

図 3 書き込みノード上の処理のタイムチャート
Fig. 3 Time Chart of Processing on Write-Node.

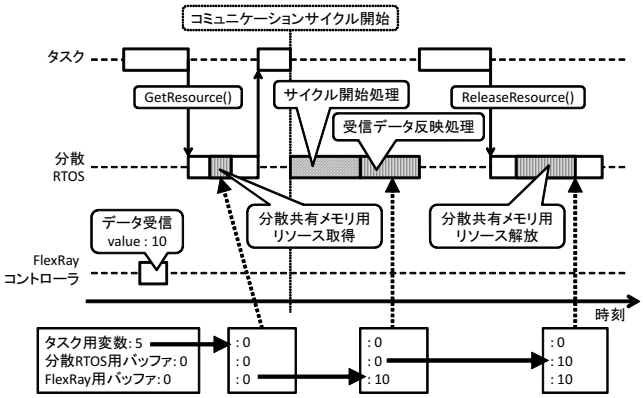

図 4 読み出しノード上の処理のタイムチャート
Fig. 4 Time Chart of Processing on Read-Node.

サイクルの開始時に分散共有メモリ用リソース取得中のタスクが存在しない場合，分散 RTOS は受信データ反映処理で分散共有メモリ更新用の値を分散 RTOS 用バッファにコピーする．

6. 分散共有メモリ処理の最悪応答時間

分散共有メモリ処理の最悪応答時間について論じるため，まず FlexRay 通信の最悪遅延時間について述べる．FlexRay のコミュニケーションサイクルは複数のタイムスロットで構成されており，コミュニケーションサイクルを構成するタイムスロットの数や各スロットの ID は全サイクルで共通である．FlexRay ドライバが生成する送信フレームのヘッダ情報の 1 つにフレーム ID があり，FlexRay コントローラはフレーム ID とスロット ID が一致するとそのフレームを送信する．メッセージ RAM に書き込まれた送信フレームは，ID が一致するスロットの時刻まで保持される．また FlexRay では，通信速度やコミュニケーションサイクルの設定によって 1 サイクル中に送信可能な総データサイズが決まる．同じサイクル中に送信を要求したフレームの合計サイズが送信可能な総データサイズを超えた場合，送信できなかったフレームは次サイクルに持ち越される．

FlexRay 通信のパラメータは通信量を考慮して設計され

る．すなわち，同一サイクル中に 1 サイクルで送信可能な総データサイズを超える通信を要求しないように設計される．この場合，あるサイクルで対応するタイムスロットが終了する前に送信フレームの書き込みが完了すれば，同じサイクルでフレームを送信できる．対応するタイムスロットが終了した後に送信フレームの書き込みが完了した場合は，次サイクルのタイムスロットでフレームを送信できる．よって，FlexRay 通信の最悪遅延時間は 2 サイクルである．

次に，分散共有メモリ処理の最悪応答時間を**図 5** を用いて説明する．図 5 は，あるノードで分散共有メモリへの書き込みが完了してから他ノードの分散 RTOS が受信した書き込みデータを確認するまでの時間（書き込みデータ確認時間）と，分散共有メモリへの書き込みが完了してから他ノードで読み出す分散共有メモリの値に反映されるまでの時間（書き込みデータ反映時間）を表している．この図は，ノード 1 の分散 RTOS が分散共有メモリ用リソース解放処理で生成した分散共有メモリの更新メッセージを，生成したサイクル中に送信できず，次のサイクルに持ち越される場合である．また，ノード 2 の分散 RTOS が受信データ反映を行う際に，分散共有メモリ用リソースをノード 2 上のタスクが取得中の場合を示している．

この場合は，アプリケーションが分散共有メモリへの書き込みを完了してから更新メッセージの送信完了まで最大で 2 サイクルかかる．よって，最悪の書き込みデータ確認時間は，2 サイクルにサイクル開始処理と受信データ反映処理の時間を加えた値となる．サイクル開始処理と受信データ反映処理は最高優先度の割り込み処理で行われるため，分散共有メモリを読み書きするタスクの実行時間やタスク状態によってそれらの処理時間が変動する事は無い．

一方，最悪の書き込みデータ反映時間は，最悪の書き込みデータ確認時間にタスクが分散共有メモリ用リソースを取得して行う処理時間を加えた値である．したがって，分散共有メモリ用リソースを取得してから解放するまでのタスクの処理時間が一定サイクル以内であれば，コミュニケーションサイクルを用いて最悪の書き込みデータ反映時間を表すことができる．例えば，図 5 のように，ノード 2 のタスクが分散共有メモリ用リソースを取得してから解放するまでの時間が 1 サイクル未満であれば，最悪の書き込みデータ反映時間は 3 サイクルとなる．

FlexRay のコミュニケーションサイクル長の標準的な値は，1msec から 10msec である．自動車制御用アプリケーションにおいて，一般にアプリケーションのタスクが共有データアクセスのためにリソースを取得している時間はそのタスクの処理時間としてはごくわずかであり，1 サイクルを超えることは考えられない．しかし，プリエンティブなマルチタスク環境で低優先度タスクがリソースを取得した後，高優先度タスクにプリエンプトされリソース取得から解放までの時間大きく延びる可能性がある．そのため，

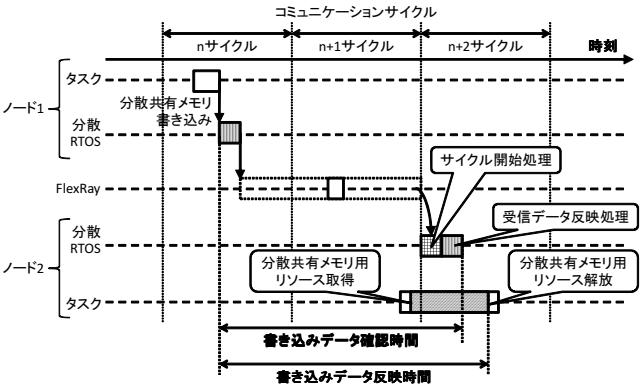

図 5 分散共有メモリ処理の最悪応答時間

Fig. 5 The Worst Case Responce Time of Distributed Shared Memory.

最悪の書き込みデータ反映時間を算出するには，各ノード上で共有メモリにアクセスするタスクがリソース取得中にプリエンプトされる最悪時間を考慮して，アプリケーション毎に算出する必要がある．

7. 分散共有メモリの評価

OSEK OS 仕様の RTOS である TOPPERS/OSEK カーネル [9] を拡張して本論文で提案した分散共有メモリ機構を備える分散 RTOS を実装し，分散共有メモリ処理の実行時間と，システムコールのオーバーヘッドを計測した．分散共有メモリ処理は，図 3 における分散共有メモリ用リソース取得処理，分散共有メモリ用リソース解放処理，分散共有メモリ更新処理と，図 4 におけるサイクル開始処理，受信データ反映処理，分散共有メモリ用リソース取得処理の実行時間を計測した．計測には V850E/PHO3 プロセッサと FlexRay コントローラを搭載した評価ボード GT201N10 を用いた．V850E/PHO3 は 32bit RISC プロセッサで，クロックは 128MHz，ROM 容量 992kByte，RAM 容量 92kByte である．FlexRay コントローラのクロックは 80MHz である．そして，2 ノード構成上で評価用アプリケーションを動作させて，性能評価を行った．分散共有メモリ上のデータは 32bit 長 1 つの場合である．

実行時間の計測にはハードウェアカウンタを用いた．ハードウェアカウンタのクロックは 32MHz で，1 カウントは 31.25nsec である．いずれの計測もハードウェアカウンタを用いて実行時間を 50 回計測し，50 回中の最悪値と平均値を最終的な計測結果として扱う．計測結果は 0.01μsec（10nsec）単位で示す．

分散共有メモリ処理の計測結果を**表 1** に示す．かっこ外の値は平均値，かっこ内の値は最悪値である．書き込みノード側の分散共有メモリ取得処理，分散共有メモリ用リソース解放処理，分散共有メモリ更新処理の実行時間の合計は約 10μsec，読み出しノード側のサイクル開始処理及び受信データ反映処理の実行時間の合計は約 33μsec

表 1　分散共有メモリ処理の実行時間
Table 1　Execution Time of Distributes Shared Memory Mechanism.

分散共有メモリ用リソース取得		0.79(0.81)
分散共有メモリ用リソース解放	データ送信有り	7.73(7.75)
	データ送信無し	0.90(0.90)
サイクル開始処理		32.07(32.09)
受信データ反映		0.84(0.84)
分散共有メモリ更新処理		1.13(1.15)

かっこ内は最悪実行時間　[μsec]

である．OSEK OS が対象とする自動車制御アプリケーションの場合，通信を伴う処理の制御周期は 10msec から 100msec 程度である．FlexRay のコミュニケーションサイクル長が 1msec の場合，最悪の書き込みデータ確認時間は約 2.033msec になる．また，分散共有メモリのデータを受信するノード側において，アプリケーションが分散共有メモリ用リソースを取得してから解放するまでの最悪時間が 0.967msec より短いと仮定した場合，最悪の書き込みデータ反映時間は 3msec になる．本分散共有メモリのいずれの最悪時間も，制御周期の 3 分の 1 程度の値である．FlexRay では各ノードからの通信が衝突するということはないので、今回の測定項目については、ノード数を増大させても値に影響することはない。

一般に制御アプリケーションは周期タスクで実装されることが多く，タスク間で共有されるデータは制御周期に一度書き換えが発生する．また，近年の自動車制御アプリケーションは MATLAB/Simulink[10] を用いたモデルベース開発により開発されることが多いが，Simulink モデルに基づく場合，アプリケーションモジュール間は一方向のデータの流れになることが多い．本分散共有メモリを用いた場合，あるモジュールで書き換えた値は最悪でも制御周期の 3 分の 1 程度の時間で他のモジュールに反映できるため，そのようなアプリケーションには十分である．

次に，システムコールのオーバーヘッドの計測結果を**表 2** に示す．GetResource()，ReleaseResource() のオーバーヘッドには，システムコールの対象リソースが分散共有メモリ用リソースであるか判定する処理時間が含まれる．どのシステムコールの場合も分散 RTOS の処理のオーバーヘッドは TOPPERS/OSEK カーネルの処理時間の 10%未満であり，十分に小さい値である．また，計測結果の平均値と最悪値の差はいずれも，計測に用いたタイマの精である 0.03125μsec（31.25nsec）以下である．従来の TOPPERS/OSEK カーネルにおける平均値と最悪値の差と比較しても大きな差はなく，実行時間の予測性の点でも問題ないと考える．

8. おわりに

組み込み制御システムを対象に，異なるノード上のアプリケーション間で値を共有可能な分散共有メモリ機構を実装するため，アプリケーションのタスクが共有メモリに読み書きをするためのメモリアクセス機能と，FlexRay 通信を用いた更新処理によって一貫性を保つ手法を提案した．そして，提案した分散共有メモリ機構を備える分散 RTOS を実装し，分散共有メモリ機構向けに追加した処理の実行時間と，システムコールのオーバーヘッドを計測し，実用性があると考えられる性能を得た．

表 2　システムコールのオーバーヘッド
Table 2　Overhead of System Calls.

システムコール	分散 RTOS	TOPPERS/OSEK カーネル
GetResource()	2.51(2.53)	2.38(2.40)
ReleaseResource()	2.79(2.81)	2.66(2.68)

かっこ内は最悪実行時間　[μsec]

現在の実装では，分散共有メモリのサポートするデータ長は 32bit のみである．今後，他のデータ長もサポートするよう拡張を予定している．また，分散共有メモリのコンフィギュレーション環境の整備として，分散共有メモリの設定情報を記述できるよう OIL を拡張し．拡張した OIL に対応する SG を開発することを予定している．

謝辞　本研究のベースとした TOPPERS/OSEK カーネル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] Stankovic, J.A. and Ramamritham, K.: The Spring Kernel: A New Paradigm for Real-Time Systems, *IEEE Software*, Vol.8, No.3, pp.62–72 (1991)
[2] 芝 公仁，大久保 英嗣：分散オペレーティングシステム Solelc の設計と実装，電子情報通信学会論文誌，Vol.J84-D-1, No.6, pp.617–626 (2001).
[3] Protic, J., Tomasevic, M. and Milutinovic, V.: Distributed Shared Memory: Concepts and Systems, *IEEE Parallel and Distributed Technology,* Vol.4, No.2, pp.63–71 (1996).
[4] 知場貴洋，齊藤政典，伊丹悠一，兪明連，横山孝典：位置透過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: *OSEK/VDX Operating System Version 2.2.3* (2005).
[6] Makowitz, R. and Temple, C.: FlexRay - A Communication Network for Automotive Control Systems, *Proceedings of 2006 IEEE International Workshop on Factory Communication Systems,* pp.207–212 (2006).
[7] Tanenbaum, A.S. and Van Steen, M.: Distributed Systems: Principles and Paradigms, Pearson Education (2002).
[8] OSEK/VDX: *OSEK/VDX System Generation OIL: OSEK Implementation Language Version 2.5* (2004).
[9] TOPPERS プロジェクト：TOPPERS/OSEK カーネル, available from ⟨http://www.toppers.jp/osek-os.html⟩.
[10] The MathWorks, Inc.: http://www.mathworks.com/