

組み込み制御システム向け分散共有メモリ機構を持つリアルタイム OS

近藤 翼[†] 兪 明連[†] 横山 孝典[†]
[†]東京都市大学[‡]

1. はじめに

近年、分散型の組み込み制御システムが増えている。例えば、自動車制御分野では複数の ECU (Electronic Control Unit) を CAN や FlexRay と呼ばれるネットワークで繋いだ分散システム構成が用いられている。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。

また、組み込み制御システムのアプリケーションは複数のタスクで構成され、タスク間でデータを共有することが多い。分散システム構成の場合は、ノード内のタスクだけでなくノード間でもデータを共有できることが望ましい。異なるノード間でデータを共有可能な分散共有メモリを利用することで、アプリケーションプログラムを書き換えることなく分散化が可能となる。このため、分散共有メモリ機能を有するリアルタイムオペレーティングシステム (Real-Time Operating System, RTOS) が求められている。

これまでに、仮想記憶機構を利用したページベースの分散共有メモリが多く提案されてきた [1]。しかし、これらは汎用 OS 向けのものであり、リアルタイム性が必要であるとともに、メモリ管理ユニット (MMU) を持たないプロセッサを使用することが多い組み込み制御システムには適していない。

リアルタイム性のある分散共有メモリを持つものとして、我々が開発した RTOS がある [2][3]。ネットワークに、FlexRay を用いることで最悪応答時間を予測可能としている。しかし、これまでは、共有変数の読み書きをノード単位で管理していたため、必ずしも効率の良いものではなかった。そこで、共有変数の読み書きをタスク単位で管理することによる改善を図った。本論文ではこの RTOS について述べる。

2. RTOS の機能

自動車制御分野の標準的な RTOS である OSEK OS [4] を拡張して、分散共有メモリ機構を有する RTOS を実現する。本 RTOS では、OSEK OS の排他制御用の API である GetResource(), ReleaseResource() を呼び出して分散共有メモリにアクセスすることとする。

本分散共有メモリ機構は、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メモリ領域を提供する。本研究では、図 1 のように、複数タスクが

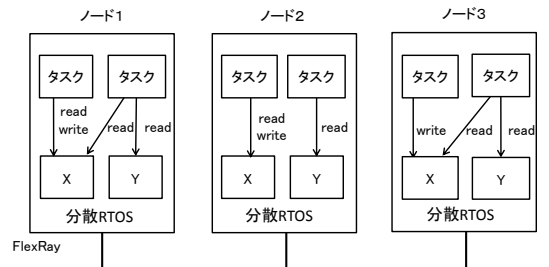


図1 ノード構成例

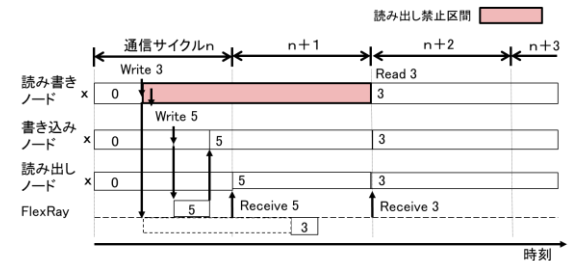


図2 読み出し禁止区間を用いた一貫性保証

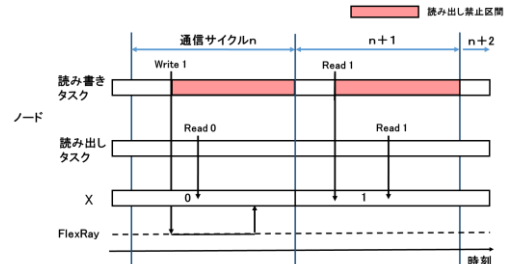


図3 タスク単位での読み出し禁止区間管理

1つの共有変数に対してアクセスする場合、読み書き両者を行うタスクと、書き込みまたは読み出しのみを行うタスクで処理を変えることで効率のよいアクセスを可能とする。

3. 分散共有メモリの一貫性保証

FlexRay は TDMA プロトコルに基づくネットワークで、全ノード上の FlexRay コントローラは同期したネットワーク時刻を持ち、通信サイクルに従って、周期的な通信を行う。FlexRay コントローラはデータの送信要求を受けると、フレーム ID と一致する ID を持つスロットの時刻まで待機してからデータを送信する。このため、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を実行する順序が異なる場合がある。そこで我々はデータの一貫性を保証するため、データの読み出しを禁止する区間を設ける。

読み出し禁止区間を用いた場合の一貫性保証を図

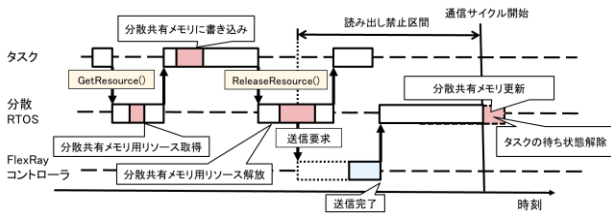


図4 読み書きタスクのタイムチャート

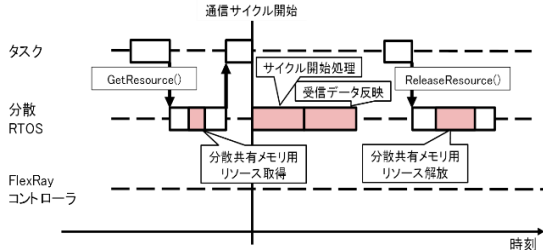


図5 読み出しタスクのタイムチャート

2を用いて説明する。図2では、読み書きを行うノードでは、書き込みを行ってから、実際に転送が行われた次の通信サイクルの開始時まで読み出しを禁止する。これにより、順序一貫性と同等の一貫性を保証している。

さらに本研究では、図3に示すようにタスクが読み書きタスクであるか、読み出しタスクであるか、書き込みタスクであるかの判定を行い、読み書きタスクだけに読み出し禁止区間を設けることで、効率的なアクセスを実現した。

図4は読み書きタスクが分散共有メモリに書き込みを行った際のタイムチャートである。GetResource()でリソースの取得後に書き込みを行い、ReleaseResource()で送信要求とリソースの解放を行う。そして、データの読み出し禁止区間の終了後に分散共有メモリ更新処理でデータの反映を行っている。

また、図5は読み出しタスクのタイムチャートを表したものである。通信サイクル開始時にサイクル処理で受信データの有無を確認し、受信データがある場合は、受信データ反映処理でデータの反映を行う。

4. 実装

本 RTOS はオープンソースの TOPPERS/ATK1[5] を拡張して実装した。実装に用いるハードウェアは FlexRay コントローラを内蔵した V850E/PH03 (32bit RISC) プロセッサを搭載した評価ボードである。CPU コアのクロック数は 128MHz、FlexRay の通信速度は 10Mbps である。

5. 評価

本 RTOS が、自動車制御アプリケーションに適用可能かを確認するため、分散共有メモリの性能評価を行った。具体的には 2 ノード 2 タスク構成の評価用アプリケーションを作成し、分散共有メモリの処理毎にその実行時間を計測した。

表 1 本研究の処理実行時間

		リソース取得処理		リソース解放処理		分散共有メモリ更新	サイクル開始処理	受信データ反映処理
		送信あり	送信なし	送信あり	送信なし			
1Byte	平均	1.3	10.8	1.3	1.6	32.0	1.5	
	最悪	1.3	10.8	1.3	1.6	32.0	1.5	
2Byte	平均	1.4	10.8	1.4	1.7	32.0	1.5	
	最悪	1.4	10.8	1.4	1.7	32.0	1.5	
4Byte	平均	1.6	11.1	1.6	1.8	32.2	1.7	
	最悪	1.6	11.1	1.6	1.8	32.2	1.7	

単位[μsec]

図4, 図5に示した、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, 通信サイクル開始処理, 受信データ反映処理の実行時間を、共有データが 1Byte, 2Byte, 4Byte のそれぞれの場合において計測した結果を表1に示す。表1に示した値は、FlexRay の通信サイクルの値やアプリケーションの実行時間の影響は受けない。

書き込みノードでの書き込みから読み出しノードで読み出し可能となるまでの最悪応答時間は、2 通信サイクルの時間に読み出しノード側の処理時間を加えたものになる。FlexRay の典型的な通信サイクルは 1msec であり、分散共有メモリの処理実行時間は、それより十分小さい。従って、最悪応答時間はほぼ通信サイクルで決まると言って良い。また、読み出し禁止区間も FlexRay の通信サイクルが 1msec の場合、2msec 以下である。一般に自動車制御アプリケーションにおいて、ネットワーク通信を行うタスクの周期は 10~100msec 程度であるため、最悪応答時間及び読み出し禁止区間はそれより十分小さく、実用上問題ないと考えている。

5. おわりに

効率の良いアクセスの可能な分散共有メモリ機構を持つ RTOS を提案するとともに、実装及び評価を行い、実用上問題ない性能が得られたことを確認した。今後、API を用いない分散共有メモリを用いた RTOS を開発予定である。

謝辞

本研究で使用した TOPPERS/ATK1 の開発者に感謝する。本研究費は JSPS 科研費 JP2450046 および JP15K00084 の助成を受けたものである。

参考文献

- [1] Protic, J., Tomasevic, M. and Multinovic, V.: Distributed Shared Memory: Concepts and Systems, IEEE Parallel and Distributed Technology, Vol. 4, No2, pp. 63-71 (1996).
- [2] 知場貴洋, 兪明連, 横山孝典, 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, 情報処理学会研究報告, Vol. 2013-EMB-28 No. 3.
- [3] 田村有士, 知場貴洋, 兪明連, 横山孝典, 組み込み制御システム向け分散共有メモリ機構を持つリアルタイム OS, FIT2014, 予稿集第 1 分冊 pp. 235-236 (2014).
- [4] OSEK/VDX Operating System Version 2.2.3 (2005).
- [5] TOPPERS/ATK1: <http://www.toppers.jp/atk1.html>.