社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－SLDM－134／2008－EMB－8（34）
2008 / 3 / 28

# 時間駆動とイベント駆動が混在する組み込み制御システムのための分散処理環境

石郷岡 祐† 伊丹悠一† 横山孝典†

† 武蔵工業大学
〒158-8557 東京都世田谷区玉堤 1-28-1
E-mail: †{g0663004,g0415007,tyoko}@sc.musashi-tech.ac.jp

**あらまし** 本論文では，全コンピュータが同期して周期的に処理を実行する時間駆動分散処理，イベントにより処理を実行するイベント駆動分散処理，ネットワークを介したデータのやりとりにしたがって処理を実行するデータ駆動分散処理をサポートする，組み込み制御システム向け分散処理環境を提案した．上記３種類の処理を混在実行させるため，実行周期を時間駆動セグメントと非時間駆動セグメントとに分け，前者で時間駆動処理を，後者でイベント駆動およびデータ駆動処理を実行する時分割実行方式を採用した．そして時分割動作を実現するリアルタイムＯＳと，上記３種類の分散処理を実現するための分散ミドルウェア，および開発環境を開発した．

**キーワード** 組み込みシステム，分散処理，時間駆動，イベント駆動，リアルタイム OS

# A Distributed Computing Environment for Embedded Control Systems with Time-Triggered and Event-Triggered Processing

Tasuku ISHIGOOKA†, Yuichi ITAMI†, and Takanori YOKOYAMA†

† Musashi Institute of Technology
1-28-1 Tamadutumi, Setagaya-ku, Tokyo, 158-8557 Japan
E-mail: †{g0663004,g0415007,tyoko}@sc.musashi-tech.ac.jp

**Abstract** The paper presents a distributed computing environment for embedded control systems that consists of time-triggered distributed processing, event-triggered distributed processing and data-driven distributed processing. We introduce time division scheduling in which the unit execution period is divided into the time-triggered processing segment and the non-time-triggered processing segment. The time-triggered distributed processing is executed in the former segment, and event-triggered distributed processing and data-driven distributed processing are executed in the latter segment. We have developed a real-time operating system for the time division scheduling, distributed computing middleware for the three kinds of distributed processing shown above, and a development environment.

**Key words** Embedded System, Distirbuted Processing, Time-Triggering,Event-Triggering, Real-Time Operating System

## 1. はじめに

分散型組み込み制御システムは自動車制御，FA(Factory Automation) 等の分野で用いられており，その多くはハードリアルタイムシステムである．ハードリアルタイムシステムは処理のデッドラインを必ず守る必要がある．ハードリアルタイムシステムにはイベントの発生をタイミングに処理を実行するイベント駆動アーキテクチャより，周期的に処理を実行する時間駆動アーキテクチャが適していると言われている [1]．しかし，多くの機能が求められる組み込み制御システムを時間駆動処理のみで構成できるとは限らない．例えば，自動車のエンジン制御 ECU(Electronic Control Unit, 電子制御ユニット) は，周期タスク以外にエンジン回転に同期して起動されるタスクを持つ．また，通信機能やユーザインタフェースを持つ組み込み制御システムには，イベント駆動が適した処理を含むことも多い．そこで，時間駆動とイベント駆動が混在した分散型の組み込み制御システムを容易に開発可能とする技術が求められている．

分散システムを容易に開発可能とする技術に分散オブジェ

クトが挙げられる．例えば，CORBA(Common Object Request Broker Architecture) [2] は情報システム分野で広く用いられている．また，リアルタイムシステム向けに Real-time CORBA [3]，組み込みシステム向けに Minimum CORBA [4] や組み込み CORBA [5] が提案されている．Real-time CORBA では，時間駆動とイベント駆動をサポートするイベントサービスが提供されている [6]．イベントサービスは，イベントの受け渡しを行うイベントチャネルによって実現される．イベントチャネルでは，イベントの経路を判断するルーティングや実行イベントのスケジューリングが動的に行われるため，処理の最悪時間の予測は容易ではない．従って，厳しいデッドラインがあるハードリアルタイムシステムに Real-Time CORBA を適用することは難しい．

我々は既に時間駆動処理を実行可能とする分散型組み込み制御システム向けのミドルウェアの開発を行っている [7]．本研究では，そのミドルウェアにイベントチャネルを追加することで，時間駆動とイベント駆動が混在する組み込み制御システムを実現可能とする．また，本ミドルウェアを利用した組み込み制御システムを容易に開発可能とするため，開発環境を提供する．自動車制御等の組み込み制御システムでは，MATLAB/Simulink [8] 等のツールを用いたモデルベースの制御設計が一般化されてきている．そこで本研究は，制御モデルを制御ブロック図で記述可能な制御システムを対象とする．また，ネットワークにシャーシ系，パワートレイン系の自動車制御で用いられる FlexRay [9] を用いる．

以下本論文では，まず第 2 節で分散処理モデルに関して述べる．次に第 3 節で提案する分散処理環境のソフトウェア構成に関して述べる．第 4 節ではミドルウェアを用いた分散処理モデルに関して述べ，第 5 節でミドルウェアの性能評価を述べる．第 6 節で競合研究に関して述べ，最後に第 7 節で本論文の結論を述べる．

## 2. 組み込み制御ソフトウェアの動作

### 2.1 分散処理モデル

本研究では，分散制御ソフトウェアの動作を，時間駆動分散処理，イベント駆動分散処理，データ駆動分散処理の 3 つのモデルで表現する．我々は，制御ブロック図で表される組み込み制御システムを，自身の属性値を周期的に更新するオブジェクトの集合で実現する手法 [10] と，それを分散環境で実現可能とする時間駆動分散オブジェクトモデルを提案した [7]．本論文では，このモデルに基づく処理を時間駆動分散処理と呼ぶ．また，本論文では，(広義の) イベント駆動分散処理を，(狭義の) イベント駆動分散処理と，データ駆動分散処理に分類する．イベントのみに同期して演算を行う処理を (狭義の) イベント駆動分散処理と呼ぶ．一方，ネットワークを介して送られてきたデータの受信イベントにより起動され，そのデータを用いて行う処理をデータ駆動分散処理と呼ぶ．また，(狭義の) イベント駆動処理とデータ駆動処理をあわせて，非時間駆動処理と呼ぶ．

以下，これらの分散処理モデルを例を用いて説明する．

(1) 時間駆動分散処理

時間駆動分散処理の例を図 1 に示す．ECU1 にエンジントルクオブジェクトが，ECU2 にスロットル開度オブジェクトが配置されている．本モデルのオブジェクトは周期的に自身の属性値を更新するメソッド update を持ち，オブジェクト間通信は属性アクセスのためのメソッド get の呼び出しに限る．

スロットル開度オブジェクトがエンジントルクオブジェクトの属性値を取得するには，図 1 に示すように，前者のオブジェクトがある ECU2 に後者のオブジェクトのレプリカを配置し，前者のオブジェクトはレプリカから属性値を取得する．エンジントルクオブジェクトからレプリカへ属性値を一定の時間周期でコピーをすることで，両者の一貫性を保持する．このコピー処理を複製処理と呼ぶ．

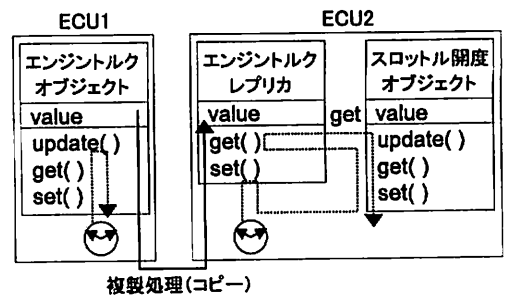


図 1 時間駆動分散処理の例

(2) イベント駆動分散処理

イベント駆動分散処理ではネットワークを介したオブジェクト間でイベントの受け渡しを行う．イベント駆動分散処理を用いた例を図 2 に示す．回転に同期した処理を行うため，角度センサオブジェクトは特定の角度でイベントを発生する．この例では，角度センサが回転に同期してイベントを出し，スロットル開度オブジェクトはそのイベントを受け取ったときに処理を開始する．

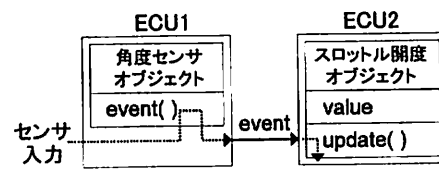


図 2 イベント駆動分散処理の例

(3) データ駆動分散処理

データ駆動分散処理は，ネットワークを介したオブジェクトの属性値の流れに沿って，各オブジェクトがデータを受信したときに処理を実行する．

図 3 に示すように，スロットル開度オブジェクトは図 1 の例と同様にエンジントルクレプリカから属性を取得する．エンジントルクオブジェクトが属性値を更新するとき，複製処理を用いてレプリカの属性を更新する．スロットル開度オブジェクトは参照するエンジントルクレプリカの属性が未更新であるとき，属性値が更新されるまで待つ．すなわち，スロットル開度オブ

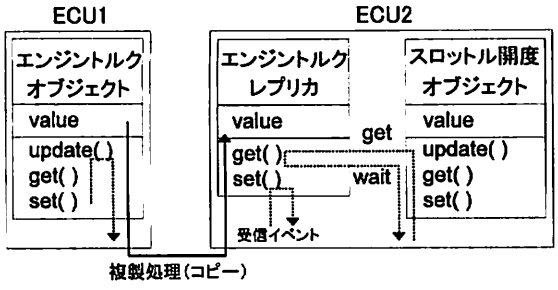


図 3　データ駆動分散処理の例

ジェクトは，レプリカの属性値が更新されたとき，または属性値を参照する前に属性値が更新されていたとき，レプリカの属性値を取得する．

### 2.2　時分割実行

組み込み制御システムでは，一定の制御周期（サンプリング周期）を持つ制御処理は小さいジッタを要求する．そこで本研究では，ジッタの発生は制御性能に影響することから，非時間駆動処理の応答時間よりも，時間駆動処理のジッタを小さくすることを重視する．そしてその実現のため，時間駆動処理を安定して実行させつつ，非時間駆動処理をも実行可能な時分割実行方式を提案する．

本方式は CPU の処理時間とネットワーク通信時間の両者について，全体時間を時間駆動処理を行なう区間（時間駆動セグメント）と非時間駆動処理を行なう区間（非時間駆動セグメント）に分けて各々の処理を実行する．CPU 処理時間については，後述する RTOS(Real-time Operating System) を開発することで，ネットワーク通信については FlexRay を使用することで実現する．

時分割実行方式の動作例を図 4 のタイムチャートを用いて説明する．時間駆動セグメントと非時間駆動セグメントは周期 T で繰り返し発生する．図 4 の例では，時間駆動セグメントで update メソッドを実行する時間駆動処理と複製処理を行う．時間駆動処理の前に複製処理の受信処理，時間駆動処理の後に複製処理の送信処理を行なう [7]．非時間駆動セグメントでは非時間駆動処理の実行を行う．イベントチャネルは非時間駆動処理で実行される．この例では FlexRay のコミュニケーションサイクルと本方式の周期 T を同一としている．また，動的セグメントと時間駆動セグメント，静的セグメントと非時間駆動セグメントを同一時間に分割している．

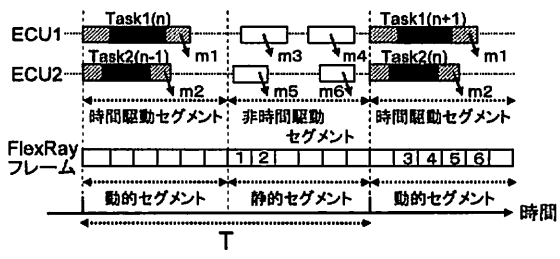


図 4　時分割実行方式のタイムチャートの例

一般に組み込み制御システムには複数の周期が存在するが，自動車制御等では最小の周期を T として $2^nT$ ($n = 0, 1, 2, \cdots$) の周期が用いられる．このようなシステムでは，時分割の周期を T とするのが基本である．オーバヘッドは多少大きくなるが，時分割の周期を $T/2^k$ ($k = 0, 1, 2, \cdots$) とし，非時間駆動処理の応答時間を改良することも可能である．

以上のように実行区間を分けることで時間駆動セグメントにおける時間駆動処理のジッタを低減できる．またネットワークや CPU 等の共有資源における競合が起こり難くなるため，最悪実行時間やジッタの予測が容易になる．本方式は FlexRay のような時分割ネットワークを用いるシステムに適しており，ネットワークの通信時間を含んだ厳密なタスク設計が可能になる．

## 3.　ソフトウェア構成

### 3.1　全体構成

提案する分散処理環境の全体構成を図 5 に示す．本環境は ECU 上で動作するアプリケーションプログラム，ミドルウェア，RTOS，ネットワークドライバ，CPU，ネットワークコントローラと，本環境の利用を支援する開発環境から成る．

アプリケーションプログラムは，アプリケーションを実現するオブジェクト群と分散処理を行なうためにオブジェクト毎に用意されるスタブから成る．ミドルウェアは複製処理とイベントチャネルの機能を提供する．RTOS はタスクの時分割実行を実現する．

開発環境は，CORBA 準拠の IDL(Interface Definition Language) からスタブ等を生成する IDL コンパイラと，開発するシステムの情報を入力し，静的にコンフィギュレーションデータを生成するコンフィギュレータから成る．

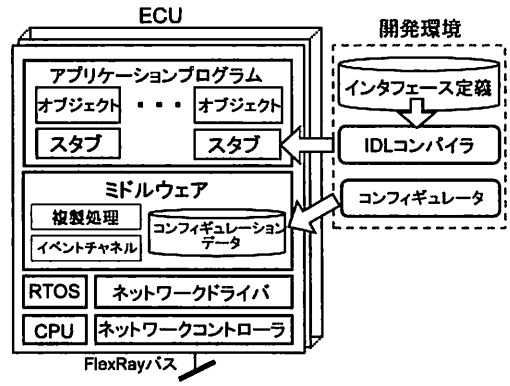


図 5　分散処理環境

### 3.2　RTOS

時分割実行が可能な RTOS は，OSEK-OS 準拠である TOPPERS/OSEK カーネルをベースにスケジューラに時間駆動処理と非時間駆動処理の切り替え時間を管理する機能を追加するとともに，実際にその切り替えを行えるようにディスパッチャを拡張することで実現する．

時間駆動セグメントで実行されるタスクを時間駆動タスク，

非時間駆動セグメントで実行されるタスクを非時間駆動タスクと呼ぶ．図6に示すように，拡張スケジューラが時間駆動セグメントと非時間駆動セグメントの切り替えを監視し，拡張ディスパッチャがそれを切り替えることで，タスクの時分割実行を実現する．また拡張スケジューラは，静的に用意されたタイムカレンダと自身が持つタイマを比較し，実行すべき時間駆動タスクの起動フラグを立てた後，ディスパッチャを呼ぶ．拡張スケジューラはOSEK-OSのカテゴリ1のISRで実行され，タイマはFlexRayのハードウェアタイマを用いてカウントする．ディスパッチャは，時間駆動タスク群から起動フラグが立っているタスクのみを順に起動する．

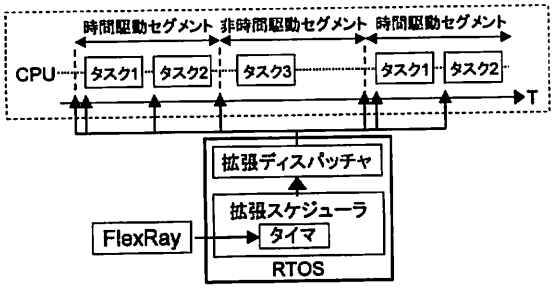


図6 RTOSを用いたタスクの時分割実行

なお時間駆動タスクより優先度が高い処理についてはタスクではなくカテゴリ1のISRで対応する．

### 3.3 ミドルウェア

時分割実行に対応したミドルウェアは，既に開発した時間駆動用ミドルウェア[7]に，イベントチャネルを追加することで実現する．本イベントチャネルはネットワークを介したイベント通信が可能であり，RTOSを利用した分散イベントサービスを提供する．本イベントチャネルはOSEK OSに対応させており，表1に示すように“イベント駆動によるタスク起動（mw_ActEvent()）”，“イベント設定（mw_SetEvent()）”，“イベント待ち（mw_WaitEvent()）”，“イベントクリア（mw_ClearEvent()）”のイベントサービスを提供する．RTOSの機能を利用することで，イベントチャネルの効率的な実装とメモリ消費量の削減が可能になる．

表1 分散イベントサービス

| サービス名 | API名 | 引数 |
|---|---|---|
| イベント駆動によるタスク起動 | mw_ActEvent | イベントID |
| イベント設定 | mw_SetEvent | イベントID |
| イベント待ち | mw_WaitEvent | イベントマスク |
| イベントクリア | mw_ClearEvent | イベントマスク |

イベントチャネルは，API(Application Program Interface)，イベントルータ，イベント通信機構で構成される．APIは，サービスを利用するアプリケーションによって呼ばれる．イベントルータは，コンフィギュレーションデータを参照し，イベントのルーティングを行う．イベントの届け先が同じECUである場合はRTOSへ，異なるECUである場合はイベント通信機構へイベントを渡す．イベント通信機構は，コンフィギュレーションデータを参照し，オブジェクトの属性取得，レプリカの属性更新，イベントとデータの送受信を行う．イベントチャネルの具体的な動作については4.2，4.3で述べる．

### 3.4 開発環境

組み込み制御システムを容易に開発可能とするための開発環境を提供する．図7に示すように，ミドルウェアが参照するコンフィギュレーションデータを生成するコンフィギュレータと，オブジェクトに合わせたスタブとレプリカを生成するIDLコンパイラを開発した．

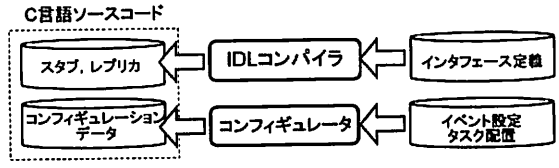


図7 開発環境を用いた生成の流れ

コンフィギュレータは，イベントと対象とするタスクの設定，タスクの配置，通信パラメータ等の入力情報に従って，コンフィギュレーションデータを生成する．また同様に，コンフィギュレータにオブジェクトの属性取得やレプリカの属性更新に関する情報を入力することで，それらに応じたコンフィギュレーションデータが生成される．

IDLコンパイラは，入力されたインタフェース定義に従ってスタブとレプリカを生成する．時間駆動向けに開発したIDLコンパイラ[7]に，データ駆動分散処理に対応したスタブとレプリカを生成する機能を追加する．データ駆動分散処理の対象となる属性の指定には，IDLの“attribute”宣言を用いる．IDLコンパイラの起動時にオプション指定することで，時間間駆動分散処理向けかデータ駆動向けかを区別する．

## 4. 分散処理動作

### 4.1 時間駆動分散処理の動作

図1の例に対応するミドルウェアを用いた複製処理の流れを図8に示す．ミドルウェアはコンフィギュレーションデータを参照し，複製処理を行なう．

まず，ミドルウェアの複製処理における送信の流れを説明する．図8で，ECU1の複製処理機構は，オブジェクト1の属性値を得るためにオリジナルスタブを呼ぶ（pack()）．オリジナルスタブは，アクセスメソッド（get_value()）を呼び出して属性値を取得し，ミドルウェア中の送信メッセージバッファに格納する．そして，複製処理機構が，送信メッセージバッファのデータをメッセージとして送信するように，ネットワークドライバに要求を出す（send()）．

次に，複製処理の受信の流れについて説明する．ECU2の複製処理機構は，他のECUから受信したメッセージを受信メッセージバッファに格納するように，ネットワークドライバに要求を出す（receive()）．次に，複製処理機構はオブジェクト1

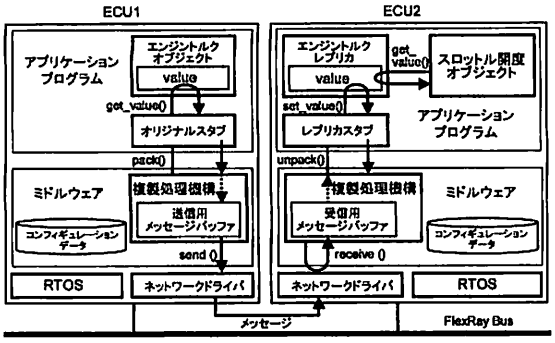


図 8 複製処理の流れ

のレプリカのレプリカスタブを呼ぶ（unpack()）．レプリカスタブは，受信メッセージバッファの内容を読み出し，レプリカのアクセスメソッド（set_value()）を呼び出してその属性に格納する．

以上により，時間駆動分散処理を実現できる．

### 4.2 イベント駆動分散処理の動作

図 2 の例に対応するイベント駆動分散処理の流れを図 9 に示す．まず，ECU1 の角度センサオブジェクトの処理が，回転角度に同期して，イベントチャネルの API（mw_ActEvent()）を呼び出す．API はイベントルータ（event_router()）を呼ぶ．イベントルータは，スロットル開度オブジェクトの位置をコンフィギュレーションデータで確認し，イベント通信機構（SendEvent()）を呼ぶ．イベント通信機構はネットワークドライバに送信要求を出す（send()）．

ECU2 では，受信割り込みがイベント通信機構（RecvEvent()）を起動する．イベント通信機構は，受信したメッセージバッファに従ってイベント ID を判断し，イベントルータ（event_router()）を呼ぶ．イベントルータはイベント ID に応じたタスクの起動要求を RTOS に出す（ActivateTask()）．

以上により，イベント駆動分散処理を実現できる．

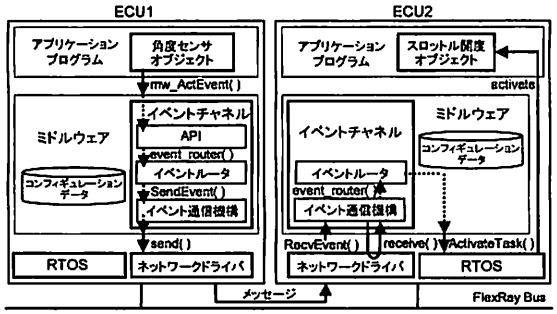


図 9 イベント駆動分散処理の流れ

### 4.3 データ駆動分散処理の動作

図 3 の例に対応するデータ駆動分散処理の流れを図 10 に示す．ECU1 のエンジントルクオブジェクトの処理が実行されると，自身の属性 "value" を更新するために，スタブ（set()）を呼ぶ．スタブはエンジントルクオブジェクトの属性を設定した後，イベントチャネルの API（mw_SetEvent()）を呼ぶ．API はイベントルータ（event_router()）を呼ぶ．イベントルータは，スロットル開度オブジェクトの位置をコンフィギュレーションデータで確認し，イベント通信機構（SendEvent()）を呼ぶ．イベント通信機構は，コンフィギュレーションデータにより属性取得の対象となるオブジェクトを特定し，エンジントルクオブジェクトの属性値を取得する．そしてイベント通信機構は取得した属性値をネットワークドライバへ渡し，送信要求を出す（send()）．

ECU2 では，受信割り込みがイベント通信機構（RecvEvent()）を起動する．イベント通信機構は，コンフィギュレーションデータを参照して受信したメッセージバッファの届け先であるレプリカを特定し，エンジントルクレプリカの属性値を更新する．同様に，イベント通信機構は受信したメッセージバッファからイベント ID を特定し，イベントルータ（event_router()）を呼ぶ．イベントルータは，コンフィギュレーションデータによってイベント ID に応じたタスクとイベントマスクを確認し，RTOS にイベント設定要求を出す（SetEvent()）．

一方 ECU2 のスロットル開度オブジェクトの処理を実行すると，エンジントルクオブジェクトの属性を取得するために，スタブ（get()）を呼ぶ．スタブは，レプリカの属性値が更新済みであることを確認するため，イベントチャネルの API（mw_WaitEvent()）を呼ぶ．API は RTOS にイベント待ち要求を出す（WaitEvent()）．

レプリカの更新を知らせるイベントが既に設定されているときは，処理は待ちに入らず，直ちに再開される．そしてスタブは，イベントのクリアを行うためイベントチャネルの API（ClearEvent()）を呼び，実行後にレプリカの属性値を取得する．一方，イベントがまだ設定されていないときは，スタブはイベントが設定されるまで待つ．

以上により，データ駆動分散処理を実現できる．

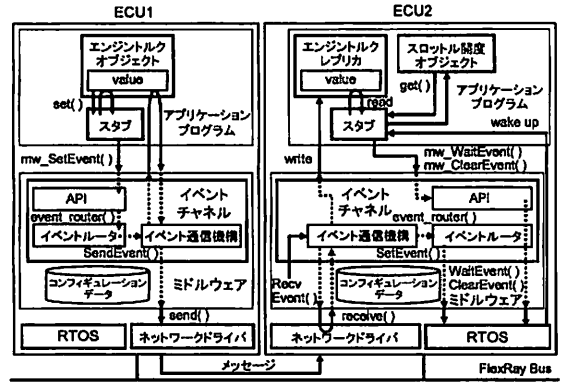


図 10 データ駆動分散処理の流れ

## 5. 実　　装

FPGA で実装した FlexRay コントローラ E-Ray を備えた GT200N00 上にミドルウェアの実装を行った．GT200N00 の

CPU は V850 で，その動作周波数は 50MHz である．

時間駆動分散処理と，RTOS のイベントサービスの時間を除いたイベント駆動分散処理とデータ駆動分散処理の測定を行った．測定結果を表 2 に示す．時間はハードウェアタイマを用いて測定し，単位は $\mu$sec である．表 2 より，イベントチャネルを含む各々の分散処理の最大時間と最小時間の差は 0.08$\mu$sec である．この値は GT200N00 の 4 クロックに相当するため，充分に小さく，実用上問題にならないと考える．

表 2　ミドルウェアの性能

| 種類 | 最大 | 最小 | 平均 | 最大-最小 |
|---|---|---|---|---|
| 時間駆動分散処理（送信側） | 43.28 | 43.20 | 43.20 | 0.08 |
| 時間駆動分散処理（受信側） | 47.04 | 47.04 | 47.04 | 0.00 |
| イベント駆動分散処理（送信側） | 33.20 | 33.12 | 33.12 | 0.08 |
| イベント駆動分散処理（受信側） | 6.08 | 6.00 | 6.00 | 0.08 |
| データ駆動分散処理（送信側） | 35.12 | 35.04 | 35.04 | 0.08 |
| データ駆動分散処理（受信側） | 3.20 | 3.20 | 3.20 | 0.00 |

## 6. 他の研究との比較

ハードリアルタイムシステム向けのハードリアルタイムイベントチャネルを有する分散処理環境として，COSMIC(Cooperating Smart Devices) ミドルウェアが提案されている [11]．このイベントチャネルは，静的に設定されたメッセージ送信のデッドラインを守ることが可能である．しかし，時間駆動分散処理との混在は考慮されていない．

タスクを時分割実行する RTOS として OSEKtime [12] が有名である．しかし時間駆動とイベント駆動を混在させるには，OSEKtime 上で OSEK-OS を動作させる必要があり，そのためのオーバヘッドが生じる．これに対し我々の RTOS は，両者を一つの OS で実装しており，オーバヘッドを低減できる．

## 7. おわりに

時間駆動とイベント駆動が混在する組み込み制御システムのための分散処理環境を開発した．本環境は時間駆動分散オブジェクトミドルウェアに，イベントによりタスクを起動するイベント駆動分散処理とデータを含むイベントによりタスクを起動するデータ駆動分散処理を実行可能とするイベントチャネルを追加することで実現した．また時間駆動処理と非時間駆動処理を各々の時間区間で実行する RTOS を開発した．本 RTOS は TOPPERS/OSEK カーネルに拡張したスケジューラとディスパッチャを追加することで実現した．

現在，ミドルウェアと RTOS のそれぞれの実装を終了し，それらの統合化を行っている．今後統合後の性能評価を行う予定である．

## 文　献


[1] Kopetz, H., "Event-Triggered Versus Time-Triggered Real-Time Systems," *Operating Systems of the 90s and Beyond 1991*, p.87-101, Dagstuhl Castle, Germany, July 1991.

[2] OMG Technical Document formal/02-06-01, "The Common Object Request Broker: Architecture and Specification," Version 3.0, 2002.

[3] OMG Technical Document formal/02-08-02, "Real-Time CORBA Specification," Version 1.1, 2002.

[4] OMG Technical Document formal/02-08-01, "Minimum CORBA Specification," Version 1.0, 2002.

[5] 伊賀徳寿，中本幸一，奥山嘉昭，佐藤直樹，檜原弘樹，"組込みシステム向け CORBA の開発と評価，" 情報処理学会論文誌コンピュータシステム，Vol.44，No.SIG10，pp.164-476，2003.

[6] Timothy H. Harrison, David L. Levine, and Douglas C. Schmidt, "The Design and Performance of a Real-time CORBA Event Service," *in Proceedings of OOPSLA '97*, Atlanta, Georgia, October 1997.

[7] 石郷岡　祐，横山孝典，"組み込み制御システム向け時間駆動分散オブジェクト環境，" 情報処理学会論文誌，Vol.48，No.9 pp.2936-2945，September 2007.

[8] Moscinski, J., "Advanced Control with MATLAB and Simulink," Ellis Horwood, Ltd., 1995.

[9] Makowitz, R. and Temple, C., "FlexRay - ACommunication Network for Automotive Control Systems," *Proceedings of 2006 IEEE International Workshop on Factory Communication Systems*, pp. 207-212, Torino, Italy, June 2006.

[10] 横山孝典，納谷英光，成沢文雄，倉垣　智，永浦渉，今井崇明，鈴木昭二，"組込み制御システムのための時間駆動オブジェクト指向ソフトウェア開発法，" 電子情報通信学会論文誌，Vol.J84-D-I，No.4 pp.338-349，April 2001.

[11] Kaiser, J., Brudna, C. and Mitidieri, C., "Implementing real-time event channels on CAN-bus," *5th IEEE International Workshop on Factory Communication Systems*, pp.274-256, Vienna, Austria, September 2004.

[12] OSEK/VDX, "Time-Triggered Operating System," Version 1.0, 2001.