Vol.2010-SE-168 No.3
Vol.2010-EMB-17 No.3
2010/6/1

車載 ECU 統合向け 異種 OS 間通信ミドルウェア

石谷健[†]　山崎二三雄[†]　長尾卓哉[†]　山田真大[†]
松原豊[†]　本田晋也[†]　高田広章[†]

近年，車載システムにおける ECU（電子制御システム）搭載数は著しく増加しており，コスト削減などの目的で複数の ECU を 1 つの ECU に統合したいという要求がある．ECU 統合の手法の 1 つに，複数 ECU の異なる OS 上で動作していたアプリケーションを 1 つの ECU 上で動作させる手法があり，我々は OSEK OS と ITRON の両 API をサポートする OS（DUOS）を開発している．本論文では，ECU 統合前に ECU 間で行っていた通信を DUOS 上で実現可能にする，OS 間通信方式について述べる．本 OS 間通信方式は，既存の車載システムの通信仕様をベースとしており，メッセージ通信，状態変数通信，状態変数グループ通信の 3 種類の通信方式から構成されている．OS 間通信方式は，性質の異なる OSEK OS と ITRON から使用されるため，それぞれの OS に適した API を定めた．また，定めた仕様を基に，DUOS 上のミドルウェアとして実装した．

Communication middleware between different kind OSes for in-vehicle ECU integration

Takeshi Ishitani[†] and Fumio Yamazaki[†] and Takuya Nagao[†] and Masahiro Yamada[†] and Yutaka Matsubara[†] and Shinya Honda[†] and Hiroaki Takada[†]

Recently, the number of Electronic Control Units (ECUs) used in the in-vehicle systems remarkably increases. Therefore, integrating ECUs becomes an important issue to reduce the cost and space. One of method to integrate ECUs is to run applications those run on each different OS in different ECUs on single OS. For the purpose of it, we developed a new OS called DUOS, supporting both API of OSEK OS and ITRON. This paper presents a method of inter-OSes communication for DUOS. It's consisted of message communication, status value communication and group of status value communication, designed similar as inter-ECU communications to realize inter-ECU communications performed before ECU integration. It's used by OSEK OS and ITRON, so its API are adapted to each OS. Based of specification of inter-OSes communication, it's implemented as middleware on DUOS.

1. はじめに

近年，車載システムの高機能化や安全支援機能の拡張が行われるようになり，搭載される ECU の数は増加している．これに伴い，ソフトウェア開発費用やワイヤーハーネス重量の増大，搭載スペースの不足などが問題となっている．これらの問題を解決するために，複数の ECU を 1 つの ECU に統合する ECU 統合手法が検討されている．その 1 つとして，各 ECU で動作していたアプリケーションを一般的な OS を用いて，1 つの ECU で動作させる手法がある．この手法を用いた場合，アプリケーションの変更やアプリケーション間の調整が避けられない．具体的には，ECU 毎に異なる OS を使用していた場合には，統合後の OS と異なる OS 上で動作していたアプリケーションを書き換える必要がある．また，それぞれの ECU で独立に動作していたアプリケーションが 1 つの ECU で動作することになるため，スケジューリングやリソース資源に対する検討が必要になる．

我々は，ECU 統合の際に既存アプリケーションへの影響を小さくできる OS として DUOS を開発している．DUOS の開発においては，ECU 統合前に動作していた OS を OSEK OS[1]，ITRON[2]と仮定している．DUOS ではその両 API をサポートすることで，OSEK OS または ITRON 上の既存アプリケーション（以下，DUOS においては OSEK アプリケーション，ITRON アプリケーションと呼ぶ）への影響を抑えることが可能である．

また，DUOS ではアプリケーション間で OS API を発行することはできない仕様となっている．これは，DUOS におけるアプリケーション間の通信が，ECU 統合前においては ECU 間通信に相当するため，統合後も OS API ではなく，ECU 間通信と同等の方式で実現した方が，アプリケーションの変更が少ないというメリットがある．また，アプリケーション間の独立性を維持できるというメリットもある．したがって，アプリケーション間での OS API 発行の必要性は低く，同一 ECU 内であっても，統合前の ECU 間通信と同等の方式でアプリケーション間の通信を実現した方が良い．

そこで，本研究では，アプリケーション間の通信を実現する OS 間通信機能を設計し，DUOS 上のミドルウェアとして実装した．OS 間通信方式は，OSEK COM 仕様や AUTOSAR COM 仕様といった既存の車載システムの通信仕様をベースとしており，メッセージ通信，状態変数通信，状態変数グループ通信の 3 種類の通信方式から構成されている．メッセージ通信は，受信したデータをキューイングすることが可能であり，データの取りこぼしを防ぐことができる．状態変数通信は，データのキューイングを行わず，データはバッファに上書きされる．書込まれたデータは何度でも読出すことが可能であるため，読出し側が最新のデータのみを必要としている場合に有効である．

[†] 名古屋大学大学院情報科学研究科附属組込みシステム研究センター
Center for Embedded Computing Systems Graduate School of Information Science, Nagoya University

Vol.2010-SE-168 No.3
Vol.2010-EMB-17 No.3
2010/6/1

状態変数グループ通信は，状態変数通信の拡張であり，複数データの送受信タイミングを一致させることによって，複数データの一貫性を保証することができる．

また，OS 間通信機能は，OSEK 及び ITRON 仕様における通信機能に合わせて，OSEK，ITRON アプリケーション毎にそれぞれ異なる API を用意している．OSEK 仕様における通信機能の特徴としては，タスクの起動や起床，コールバック関数の呼び出しなどによって，データの送受信を相手に通知することが挙げられる．ITRON 仕様における通知機能の特徴としては，データの送受信ができない場合は，タスクが送受信待ち状態に遷移し，相手からのデータの送受信によって起床することが挙げられる．以上の特徴より，OSEK アプリケーション用 API は，OSEK アプリケーションに対する通知機能を持つが，API 発行元タスクを送受信待ち状態に遷移させる機能は持たない．ITRON アプリケーション用 API は，OSEK アプリケーションに対する通知機能と，API 発行元タスクを送受信待ち状態に遷移させる機能を持つ．

本論文では，まず 2 章で車載 ECU 統合向け OS について述べ，3 章で既存車載システムにおける通信機能について述べる．4 章では OS 間通信の要件について述べ，5 章では OS 間通信の仕様について述べる．6 章では仕様に基づいて実装した OS 間通信の性能評価について述べる．7 章では関連研究について述べる．8 章では本論文のまとめを行う．

2. 車載 ECU 統合向け OS

DUOS は，ECU 統合の際に既存アプリケーションへの影響を可能な限り小さくすることを目指して開発した OS である．本章では，DUOS の概要について説明する．

図 1 は ECU 統合前と統合後のシステムを示している．ECU 統合に関しては，統合前の各 ECU の CPU パワーの合計よりも，統合後の CPU パワーの方が大きいことが前提である．また，統合前の ECU の OS は，一般的な車載システムで使用されている OSEK OS と ITRON であることを前提としている．

図 1 の ECU 統合前のシステムに対する ECU 統合を検討する．ECU 統合後のシステムにおけるオーバーヘッドを抑えるために，ECU 統合後は 1 つの OS 上で 4 つのアプリケーションを動作させることが望ましい．また，これらのアプリケーションへの影響を小さくするために，ECU 統合前のアプリケーションが使用していた OS の機能をサポートすることや，各アプリケーションが時間制約を満たせるようにすることが必要である．DUOS はこれらの要件を満たすために，OSEK OS API レイヤや ITRON API レイヤ，階層型スケジューラ[3][4][5]から構成されている．

OSEK OS API レイヤと ITRON API レイヤは，それぞれ OSEK OS，ITRON の機能を備えている．OSEK アプリケーションは，OSEK OS API レイヤの機能が使用でき，ITRON アプリケーションは，ITRON API レイヤの機能が使用できる．

階層型スケジューラは，システムで 1 つであるグローバルスケジューラと，各アプリケーションに対応するローカルスケジューラの 2 階層で構成されている．グローバルスケジューラはアプリケーションのスケジューリングを行っており，ローカルスケジューラはアプリケーション内におけるタスクのスケジューリングを行っている．

グローバルスケジューラは，周期実行されるアプリケーションを EDF 方式に基づいてスケジューリングする．アプリケーションのデッドラインは起動周期に一致する．周期内におけるアプリケーションの動作可能な時間は，アプリケーションの周期と CPU 利用率より決定しており，グローバルスケジューラがアプリケーションの動作可能な残り時間（バジェット）を管理している．実行中のアプリケーションのバジェットがなくなると，グローバルスケジューラはバジェットが存在するアプリケーションの中で最もデッドラインが早いアプリケーションに切替える．また，アプリケーションが次の周期を迎えると，バジェットを初期状態に戻す．

ローカルスケジューラのスケジューリング方式は，静的優先度割り当ての固定優先度方式であり，OSEK OS，ITRON でも採用されている．ローカルスケジューラは，アプリケーションに属する実行可能なタスクの中で，最も優先順位の高いタスクをスケジューリングする．

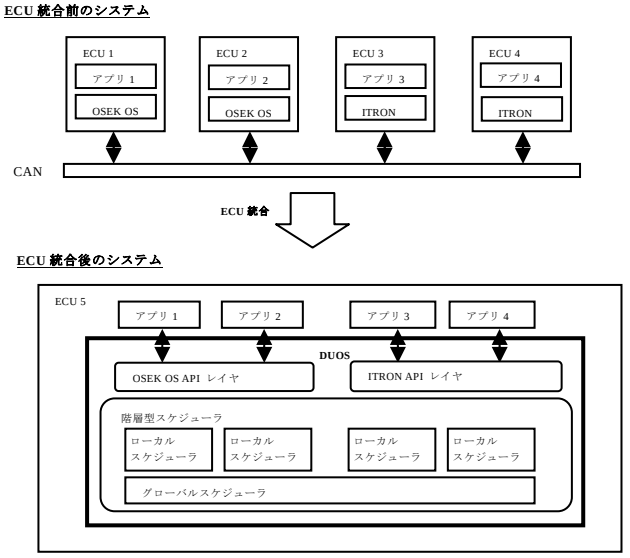

図 1 ECU 統合前後のシステム

Vol.2010-SE-168 No.3
Vol.2010-EMB-17 No.3
2010/6/1

3. 車載システム用通信機能

DUOS における OS 間通信について説明する前に，まず既存の車載システムにおける通信機能について説明する．本章では，OSEK COM ver3.0.3[6]と AUTOSAR COM ver3.0.2[7]の仕様を取り上げる．

3.1 OSEK COM 仕様

OSEK COM 仕様には，送信されたデータを受信側でキューイングするキューイングメッセージと，キューイングされない非キューイングメッセージが存在する．キューイングメッセージでは，キューに空きがある状態で新しいデータが到着すると，データは FIFO 順にキューに格納される．しかし，キューがフルの状態で新しいデータが到着すると，そのデータは格納されない．キューにデータが存在する状態でデータの受信を行うと，受信されたデータはキューから削除される．非キューイングメッセージはキューイングを行わず，通信用のバッファは 1 つである．非キューイングメッセージではデータが削除されることはなく，何度でも読出すことが可能である．非キューイングメッセージへの書込みはデータの上書きとなり，読出す度に最新の値を読出すことができる．

3.2 AUTOSAR COM 仕様

AUTOSAR COM 仕様における通信方式の基本的な部分は OSEK COM 仕様を継承しているが，AUTOSAR COM 仕様にはシグナルグループという新たな概念が存在する．その概要を ECU 間で行われる通信の概略図である図 2 を用いて説明する．

AUTOSAR では，送受信データをグループ化することが可能であり，グループ化していない送受信データをシグナル，グループ化した送受信データをグループシグナル，そのグループのことをシグナルグループと表現している．シグナルグループは，複数のグループシグナルの送受信タイミングを一致させることによって，データ一貫性を保証したい場合に用いられる．データ一貫性を保証するために使用されるのが，各グループシグナルに対応するシャドウバッファである．AUTOSAR COM を用いて ECU 間通信を行う場合，送信側 ECU における AUTOSAR COM の上位レイヤはシャドウバッファ内に送信したいグループシグナルを書込む．シャドウバッファ内のグループシグナルは，上位レイヤのトリガによってシグナルグループ単位で一度に I-PDU に書込まれる．I-PDU に書込まれたシグナルグループやシグナルは，受信側 ECU に送信される．受信側における処理は，送信側における処理の逆のプロセスで行われる．上位レイヤは I-PDU からシャドウバッファにシグナルグループ単位で読出し，シャドウバッファからグループシグナルを読出す．

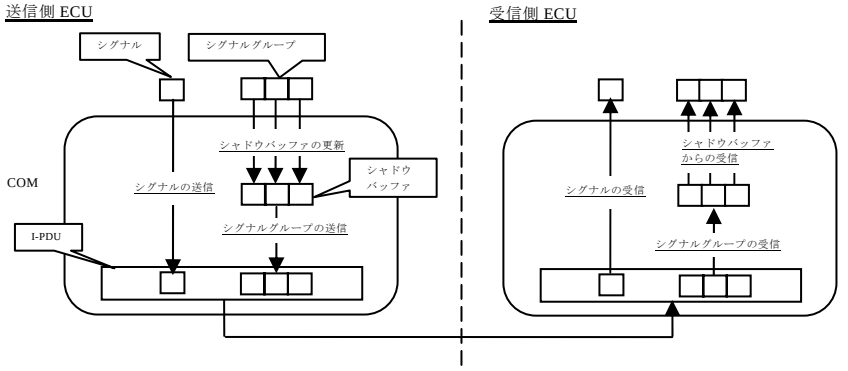

図 2　AUTOSAR COM 仕様

4. OS 間通信の要件

OS 間通信の仕様検討にあたって，次の要件を定めた．

(a) 既存の車載システムにおける通信方式をベースとする．
(b) 任意のアプリケーション間における通信を可能にする．
(c) 各 OS 仕様（OSEK OS，ITRON）は拡張しない．
(d) ミドルウェアとして実現する．

(a)に関しては，ECU 統合前に ECU 間で行っていた通信を OS 間通信で代替する際に，既存アプリケーションへの影響を小さくするためである．OS 間通信のベースとする通信方式は，3 章で述べた OSEK COM 仕様のキューイングメッセージと非キューイングメッセージ，AUTOSAR COM 仕様のシグナルグループとする．

(b)を満たすためには，OSEK アプリケーション間や ITRON アプリケーション間の通信のみならず，OSEK アプリケーションと ITRON アプリケーション間の通信も可能とする必要がある．そのため，通信を行うアプリケーションが OSEK，ITRON アプリケーションであることに関わらず，OS 間通信の通信方式は共通とする．

(c)に関しては，DUOS では OSEK OS 仕様と ITRON 仕様に準拠して拡張は行っていないため，OS 間通信においても同様に OSEK 仕様と ITRON 仕様に拡張を行わずに適合させる．(b)より，OS 間通信の通信方式は，OSEK，ITRON アプリケーションに関わらず共通であるが，両アプリケーションに対して同じ仕様の API を提供しようとす

Vol.2010-SE-168 No.3
Vol.2010-EMB-17 No.3
2010/6/1

ると，OSEK 仕様と ITRON 仕様の通信機能の性質は異なるため，どちらかの仕様の拡張は避けられない．

OSEK 仕様における通信機能は，基本的にノンブロック通信（ないしは非同期通信）であり，データがない場合も待ち状態とならない．また，データの送受信が発生した時点で，タスクの起動や起床，コールバック関数の呼び出しなどによって通知がなされる．一方，ITRON 仕様における通信機能は，ブロック通信が可能であり，バッファの関係でデータの送受信ができない場合，API 発行元タスクを送受信待ち状態に遷移させる．データの送受信が発生すると，ブロックされていたタスクは起床させられる．そのため，ITRON 仕様をベースとすると，OS 間通信用 API においても待ち状態への遷移機能が必要となり，OSEK 仕様に待ち状態を導入しなければならなくなる．逆に，OSEK 仕様をベースとすると，ITRON 仕様に対してコールバック関数の仕様を追加しなければならなくなる．また，ITRON アプリケーションにとっては，送受信待ち状態に遷移することが出来ないため，使い勝手が悪くなる．

そこで，OSEK，ITRON 仕様における通信機能に合わせて，同一の通信方式であっても，OSEK，ITRON アプリケーション毎にそれぞれの OS の通信機能の性質に合わせた OS 間通信用 API を用意する．OSEK アプリケーション用 API は，OSEK アプリケーションに対する通知機能を持つが，API 発行元タスクを送受信待ち状態に遷移させる機能は持たない．ITRON アプリケーション用 API は，OSEK アプリケーションに対する通知機能と，API 発行元タスクを送受信待ち状態に遷移させる機能を持つ．

最後に，(d)に関しては，OS 間通信のベースとした OSEK COM 仕様や AUTOSAR COM 仕様では，通信機能が移植性を考慮して OS とは独立したミドルウェアとして定義されているためである．OS 間通信も DUOS とは独立したミドルウェアとして実装することによって，DUOS における通信のみではなく，マルチ OS 環境の OS 間における通信としても再利用可能となる．なお，DUOS の基本的な考え方として，アプリケーション間では OS API を発行することはできないが，OS 間通信の実現のためには，OS API の発行は必須であるため，アプリケーション間で発行して問題がない幾つかの OS API に関しては，OS 間通信の内部において使用することとする．

5. OS 間通信の仕様

OS 間通信は，OSEK COM 仕様，AUTOSAR COM 仕様より，キューイングメッセージをベースとしたメッセージ通信，非キューイングメッセージをベースとした状態変数通信，非キューイングメッセージの拡張機能としてシグナルグループの特徴を盛り込んだ状態変数グループ通信の 3 種類とした．本章では，メッセージ通信，状態変数通信，状態変数グループ通信の仕様について説明する．

5.1 メッセージ通信

メッセージ通信は，メッセージキューを使用して任意のサイズのデータを送受信することによって通信する．キューの構造は，OSEK COM 仕様のキューイングメッセージと同様である．また，メッセージ通信は，送信アプリケーション，受信アプリケーションが 1 つのみ設定可能である 1:1 通信である．したがって，メッセージ通信では，OSEK アプリケーション間，ITRON アプリケーション間，OSEK-ITRON アプリケーション間で通信を行うパターンが存在する．

通知機能としては，メッセージキューにデータを送信したことを受信アプリケーションに通知する送信通知，メッセージキューに空きができたことを送信アプリケーションに通知する送信空き通知の 2 種類がある．但し，通知機能は，通知先のアプリケーションが OSEK アプリケーションである場合にのみ有効である．通知方式は，タスク起動，タスク起床，コールバック関数コールから選択して設定可能である．

図 3 に OSEK アプリケーション間におけるメッセージの送受信と通知機能に対する処理概要を示す．送信アプリケーションに属するタスク 1 が，メッセージの送信を行うと，送信通知としてタスク 2 に対するタスク起動が設定されている場合，受信アプリケーションに属するタスク 2 が起動される．また，メッセージキューがフルのため，送信エラーとなった後，メッセージキューに空きができると，送信空き通知としてタスク 1 に対するタスク起動が設定されている場合，送信アプリケーションに属するタスク 1 が起動される．

次に，待ち状態への遷移に関しては，ITRON アプリケーションのみ可能であり，メッセージキューが空やフルであるために送受信できない時に，送受信待ちや送受信タイムアウト待ちに遷移する機能がある．

図 4 に ITRON アプリケーション間におけるメッセージの送受信と待ち機能に対する処理概要を示す．メッセージキュー1 がフルの場合，送信アプリケーションに属するタスク 2 がメッセージを送信しようとすると送信待ち状態に遷移する．その後，受信アプリケーションに属するタスク 3 が，メッセージの受信を行うことによって，メッセージキュー1 に空きができると，メッセージの送信，送信待ち解除が行われる．また，メッセージキュー2 が空の場合，受信アプリケーションに属するタスク 4 がメッセージを受信しようとすると受信待ち状態に遷移する．その後，送信アプリケーションに属するタスク 1 が，メッセージの送信を行うと，送信メッセージはタスク 4 に渡され，受信待ち解除が行われる．

Vol.2010-SE-168 No.3
Vol.2010-EMB-17 No.3
2010/6/1

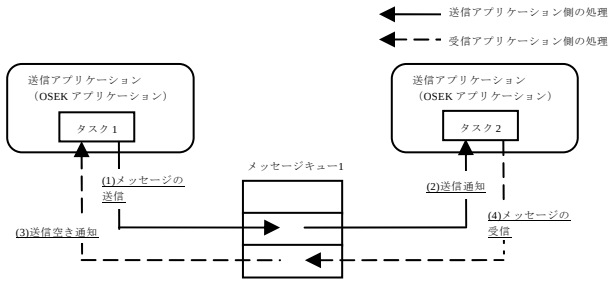

図 3　メッセージの送受信と通知機能の関係
（OSEK アプリケーション間）

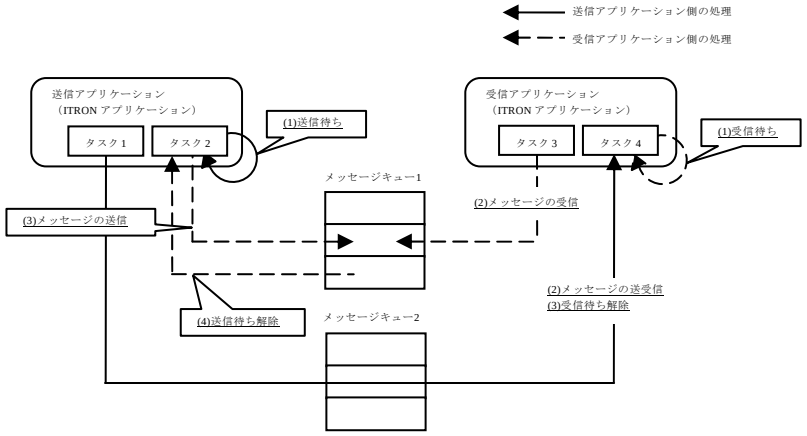

図 4　メッセージの送受信と待ち機能の関係
（ITRON アプリケーション間）

5.2　状態変数通信

状態変数通信は，状態変数に任意のサイズのデータを書込んだり，読込んだりすることによって通信する．状態変数通信の構造は，OSEK COM 仕様の非キューイングメッセージと同様であり，メッセージ通信のようにデータをキューイングすることはなく，以前に書込まれたデータを上書きする．また，書込まれたデータは読込みによって消費されることはない．1 つの状態変数に対して書込みアプリケーションは 1 つのみ設定可能であるが，読込みアプリケーションは複数設定可能である．つまり，状態変数は 1:n 通信である．

通知機能としては，状態変数にデータを書込んだことを読込みアプリケーションに通知する書込み通知があり，通知先のアプリケーションが OSEK アプリケーションである場合にのみ有効である．通知方式は，タスク起動，タスク起床，コールバック関数コールから選択して設定可能である．読込みアプリケーションが複数設定されている場合は，読込みアプリケーション毎に通知方式を設定可能である．また，待ち状態への遷移に関しては，ITRON アプリケーションのみが可能であり，最新のデータを取得するために，状態変数の値が更新されるまで待ち状態に遷移する更新待ちの機能がある．

図 5 に状態変数の書込み/読込みと，通知機能や待ち機能に対する処理概要を示す．読込みアプリケーションに属するタスク 3 が，更新待ちを有効にして読込みを行うと，更新待ち状態に遷移する．その後，書込みアプリケーションに属するタスク 1 が状態変数への書込みを行うと，タスク 3 へ状態変数のデータが渡され，更新待ち解除が行われる．また，別の読込みアプリケーションへの書込み通知として，タスク 2 に対するタスク起動が設定されている場合，タスク 2 が起動される．

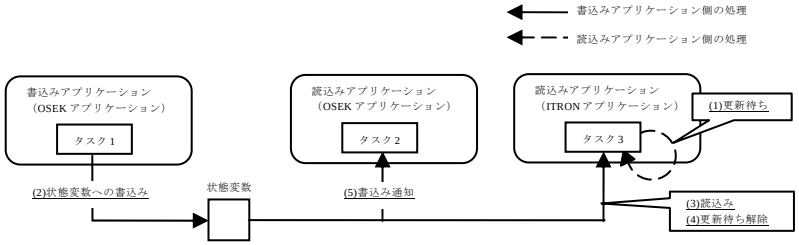

図 5　状態変数の書込み/読込みと，通知機能や待ち機能の関係

5.3　状態変数グループ通信

状態変数グループ通信は，AUTOSAR COM 仕様のシグナルグループの構造をベースに，状態変数通信の機能拡張を行っている．状態変数グループ通信では，複数の状態変数をグループ化して，グループ化した状態変数を一度に読込みアプリケーション側に渡すことによって，データ一貫性を保証することができる．これを実現するために，書込みバッファ，メインバッファ，読込みバッファを用いる．これらのバッファは状態変数毎に用意されている．また，読込みアプリケーションが複数設定されている場

Vol.2010-SE-168 No.3
Vol.2010-EMB-17 No.3
2010/6/1

合は，読込みアプリケーション毎に読込みバッファが用意される．図 6 に状態変数グループ通信の処理概要について示す．書込みアプリケーションに属するタスク 1 は，状態変数の書込みバッファに対して書込みを行い，任意のタイミングでグループ化された書込みバッファを一度にメインバッファに書込む．読込みアプリケーションに属するタスク 2 は，メインバッファのデータを任意のタイミングで一度に読込みバッファに読込み，その後読込みバッファに対して読込みを行う．

通知機能としては，書込みアプリケーションがメインバッファの値を更新したことを，読込みアプリケーションに通知する更新通知があり，通知先のアプリケーションが OSEK アプリケーションである場合にのみ有効である．通知方式は，タスク起動，タスク起床，コールバック関数コールから選択して設定可能である．読込みアプリケーションが複数設定されている場合は，読込みアプリケーション毎に通知方式を設定可能である．また，待ち状態への遷移に関しては，ITRON アプリケーション，かつ読込みアプリケーションのみが可能であり，メインバッファの値が更新されるまで待ち状態に遷移するグループ更新待ちの機能がある．

図 7 に状態変数グループの書込み/読込みと，通知機能や待ち機能に対する処理概要を示す．読込みアプリケーションに属するタスク 3 が，グループ更新待ちを有効にしてメインバッファから読込みバッファへの読込みを行うと，グループ更新待ち状態に遷移する．その後，書込みアプリケーションに属するタスク 1 が状態変数への書込み，状態変数グループへの書込みを行うと，状態変数グループへの書込みに続いてタスク 3 が属するアプリケーションに対応する読込みバッファに対する状態変数グループからの読込みが行われ，タスク 3 のグループ更新待ち解除が行われる．また，別の受信アプリケーションへのグループ更新通知として，タスク 2 に対するタスク起動が設定されている場合，タスク 2 が起動される．

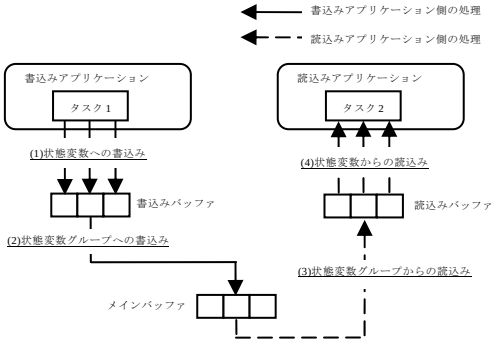

図 6　状態変数グループ通信

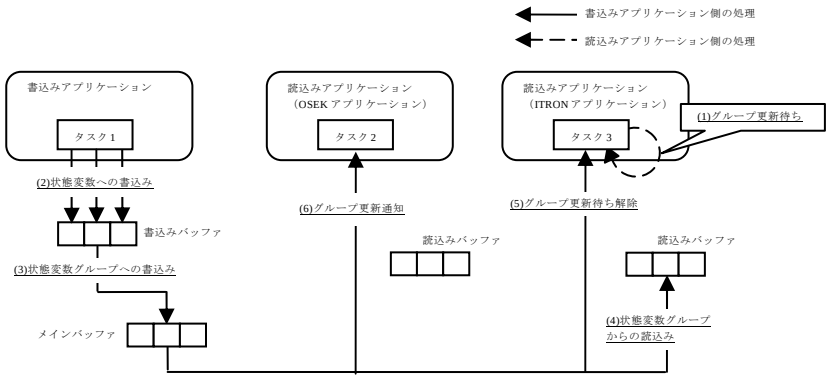

図 7　状態変数グループの書込み/読込みと，通知機能や待ち機能の関係

5.4 OS 間通信用 API

本研究で実装した 3 種類の OS 間通信用の API 数は 76 個である．ITRON アプリケーションのタスクコンテキストから呼び出される API 名の先頭には itron を付け，ITRON アプリケーションの非タスクコンテキストから呼び出される API 名の先頭には iitron を付けている．また，OSEK アプリケーションから呼び出される API 名の先頭には osek を付けている．これは，OSEK OS の API は，タスクコンテキスト用と非タスクコンテキスト用で同一であり，ITRON の API は，タスクコンテキスト用と非タスクコンテキスト用に区別されていることに倣ったためである．

Vol.2010-SE-168 No.3
Vol.2010-EMB-17 No.3
2010/6/1

(1) **メッセージ通信用 API**

メッセージ通信用の API 数は 28 個である．その一部を表 1 に示す.

表 1　メッセージ通信用 API

API の名称	API の説明
itron_snd_msg osek_snd_msg	メッセージの送信を行う．itron_snd_msg が送信できない場合は待ち状態に遷移し，osek_snd_msg が送信できない場合はエラーでリターンする．必要であるならば，送信通知や受信待ち解除を行う．
itron_psnd_msg iitron_psnd_msg	メッセージの送信をポーリングで行う．必要であるならば，送信通知や受信待ち解除を行う．
itron_tsnd_msg	メッセージの送信を行う．メッセージが送信できない場合は，タイムアウト付き待ち状態へ遷移する．必要であるならば，送信通知や受信待ち解除を行う．
itron_rcv_msg osek_rcv_msg	メッセージの受信を行う．itron_rcv_msg が受信できない場合は待ち状態に遷移し，osek_rcv_msg が受信できない場合はエラーでリターンする．必要であるならば，送信空き通知や送信待ち解除を行う．
itron_prcv_msg iitron_prcv_msg	メッセージの受信をポーリングで行う．必要であるならば，送信空き通知や送信待ち解除を行う．
itron_trcv_msg	メッセージの受信を行う．メッセージが受信できない場合は，タイムアウト付き待ち状態へ遷移する．必要であるならば，送信空き通知や送信待ち解除を行う．

(2) **状態変数通信用 API**

状態変数通信用の API 数は 18 個である．また，状態変数グループ通信と共通に使用できる API が 6 個である．その一部を表 2 に示す.

表 2　状態変数通信用 API

API の名称	API の説明
itron_write_stv iitron_write_stv osek_write_stv	状態変数への書込みを行う．必要であるならば，書込み通知や更新待ち解除を行う．
itron_read_stv iitron_read_stv osek_read_stv	状態変数からの読込みを行う．itron_read_stv の引数で更新待ちが有効になっている場合は，更新待ち状態に遷移する．

(3) **状態変数グループ通信用 API**

状態変数グループ通信用の API 数は 24 個である．また，状態変数通信と共通に使用できる API が 6 個である．その一部を表 3 に示す.

表 3　状態変数グループ通信用 API

API の名称	API の説明
itron_shadow_write_stv iitron_shadow_write_stv osek_shadow_write_stv	状態変数グループに属する状態変数の書込みバッファに書込みを行う．
itron_update_stvgp iitron_update_stvgp osek_update_stvgp	状態変数グループに属する全ての状態変数の書込みバッファからメインバッファへのコピーを行う．必要であるならば，グループ更新通知やグループ更新待ち解除を行う．
itron_fetch_read_stv iitron_fetch_read_stv osek_fetch_read_stv	状態変数グループに属する状態変数の読込みバッファから読込みを行う．
itron_fetch_stvgp iitron_fetch_stvgp osek_fetch_stvgp	状態変数グループに属する全ての状態変数のメインバッファから読込みバッファへのコピーを行う．itron_fetch_stvgp の引数でグループ更新待ちが有効になっている場合は，グループ更新待ち状態に遷移する．

6. OS 間通信の評価

本章では，DUOS におけるミドルウェアとして実装した OS 間通信の性能に対する評価を行い，組込みミドルウェアとして許容できる範囲内にあることを確かめる．

6.1 評価環境

測定環境としては，Cortex-A9 MP Core プロセッサを搭載した KZM-CA9-01 ボードを用いた．使用したコアは 1 コアのみである．コアクロックは 400MHz，タイマークロックは 400MHz，キャッシュは命令・データ共に 32KB である．送信/書込み，受信/読込み時間の測定に関しては，100 回繰り返し測定を行い，その平均値を測定結果とした．キャッシュは測定毎にパージした．

6.2 送信/書込み，受信/読込み時間測定

送信/書込み，受信/読込み時間測定では，OS 間通信における送信/書込み時間と受信/読込み時間の測定を行った．測定条件を下記に示す.

- OS 間通信を行うコンテキストはタスクとする.
- 送受信データサイズ，書込み/読込みデータサイズは 4B とする.
- 待ち解除を行う場合は，待ち解除されるタスクのみが待ち状態に遷移していることとする．また，待ち解除されたタスクが所属するアプリケーションは実行可能状態であり，待ち解除されたタスクはそのアプリケーション内にて実行可能状態

Vol.2010-SE-168 No.3
Vol.2010-EMB-17 No.3
2010/6/1

に遷移することとする．

- 送信/書込み，受信/読込みによる通知方式はタスク起動とする．また，起動されたタスクが所属するアプリケーションは実行可能状態であり，起動されたタスクはそのアプリケーション内にて実行可能状態に遷移することとする．
- 状態変数グループ通信における測定では，状態変数グループに属する状態変数は 1 つとする．また，測定結果は，書込みバッファとメインバッファへの書込み処理の合計時間，メインバッファと読込みバッファからの読込み処理の合計時間とする．

(1) 1:1 通信

1:1 通信における送信/書込み時間と受信/読込み時間を測定した．1:1 通信では，1 つの送信/書込みアプリケーションと 1 つの受信/読込みアプリケーションが通信を行う．1:1 通信における送信/書込み時間の測定結果を表 4 に，受信/読込み時間の測定結果を表 5 に示す．

表 4　1:1 通信における送信/書込み時間

OS 間通信の種類	API 属性 (API 名)	送信/書込み通知の有無	受信/更新待ち解除処理の有無	送信/書込み時間
メッセージ通信	OSEK アプリケーション用 API (osek_snd_msg)	無し	無し	44 μs
		無し	有り	62 μs
		有り	無し	57 μs
	ITRON アプリケーション用 API (itron_snd_msg)	無し	無し	47 μs
		無し	有り	63 μs
		有り	無し	63 μs
状態変数通信	OSEK アプリケーション用 API (osek_write_stv)	無し	無し	34 μs
		無し	有り	55 μs
		有り	無し	52 μs
	ITRON アプリケーション用 API (itron_write_stv)	無し	無し	35 μs
		無し	有り	55 μs
		有り	無し	53 μs
状態変数グループ通信	OSEK アプリケーション用 API (osek_shadow_write_stv, osek_update_stvgp)	無し	無し	67 μs
		無し	有り	96 μs
		有り	無し	86 μs
	ITRON アプリケーション用 API (itron_shadow_write_stv, itron_update_stvgp)	無し	無し	68 μs
		無し	有り	97 μs
		有り	無し	85 μs

表 5　1:1 通信における受信/読込み時間

OS 間通信の種類	API 属性 (API 名)	送信空き通知の有無	送信待ち解除処理の有無	受信/読込み時間
メッセージ通信	OSEK アプリケーション用 API (osek_rcv_msg)	無し	無し	43 μs
		無し	有り	67 μs
		有り	無し	56 μs
	ITRON アプリケーション用 API (itron_rcv_msg)	無し	無し	47 μs
		無し	有り	71 μs
		有り	無し	62 μs
状態変数通信	OSEK アプリケーション用 API (osek_read_stv)	無し	無し	33 μs
	ITRON アプリケーション用 API (itron_read_stv)	無し	無し	33 μs
状態変数グループ通信	OSEK アプリケーション用 API (osek_fetch_read_stv, osek_fetch_stvgp)	無し	無し	80 μs
	ITRON アプリケーション用 API (itron_fetch_read_stv, itron_fetch_stvgp)	無し	無し	80 μs

表 4，表 5 の測定結果より，OS 間通信の送信/書込み時間，受信/読込み時間は，OS 間通信の種類，通知や待ち解除の有無によって数十 μs のばらつきはあるが，全ての測定結果は 100 μs 以下である．OS 間通信は ECU 間通信を代替する機能であることより，この測定結果は十分に小さい．

(2) 1:n 通信

状態変数通信と状態変数グループ通信の 1:n 通信における書込み時間を測定した．1:n 通信では，1 つの書込みアプリケーションと複数の読込みアプリケーションが通信を行う．本測定では 1:2 通信と 1:4 通信に対して測定を行った．状態変数通信と状態変数グループ通信の仕様上，1:n 通信における読込み時間は 1:1 通信と同等であるため，書込み時間のみを測定した．状態変数通信の測定結果を表 6 に，状態変数グループ通信の測定結果を表 7 に示す．

Vol.2010-SE-168 No.3
Vol.2010-EMB-17 No.3
2010/6/1

表 6　状態変数通信における 1:n 通信の書込み時間

API 属性 (API 名)	書込み 通知の有無	更新待ち 解除処理の有無	1:1 通信の 書込み時間	1:2 通信の 書込み時間	1:4 通信の 書込み時間
OSEK アプリケーション用 API (osek_write_stv)	無し	無し	34μs	35μs	37μs
	無し	有り	55μs	78μs	123μs
	有り	無し	52μs	73μs	112μs
ITRON アプリケーション用 API (itron_write_stv)	無し	無し	35μs	36μs	37μs
	無し	有り	55μs	78μs	120μs
	有り	無し	53μs	72μs	108μs

表 7　状態変数グループ通信における 1:n 通信の書込み時間

API 属性 (API 名)	グループ更新 通知の有無	グループ更新待ち 解除処理 の有無	1:1 通信の 書込み時間	1:2 通信の 書込み時間	1:4 通信の 書込み時間
OSEK アプリケーション用 API (osek_shadow_write_stv, osek_update_stvgp)	無し	無し	67μs	68μs	70μs
	無し	有り	96μs	127μs	181μs
	有り	無し	86μs	105μs	141μs
ITRON アプリケーション用 API (itron_shadow_write_stv, itron_update_stvgp)	無し	無し	68μs	71μs	73μs
	無し	有り	97μs	126μs	178μs
	有り	無し	85μs	105μs	145μs

表 6，表 7 の測定結果より，1:1 通信，1:2 通信，1:4 通信を比較すると，1:1 通信における通知や待ち解除の有無による数十μs の差が，読込みアプリケーション数に比例して増加している．これは，読込みアプリケーション毎に通知によって起動されるタスクや，待ち状態に遷移しているタスクが存在しており，書込み処理によって全てのタスク起動や待ち解除処理を行っているためである．1:n 通信の場合は，起動されるタスク数や待ち解除されるタスク数によって書込み時間が異なってくるが，1:4 通信においても 200μs 以下となっている．OS 間通信は ECU 間通信を代替する機能であることより，この測定結果は十分に小さい．以上の内容より，OS 間通信は組込みミドルウェアで許容できる範囲内にある．

6.3 プログラムサイズ測定

実装した OS 間通信機能のサイズが，組込みミドルウェアとして許容される範囲内にあることを確かめるために，DUOS と OS 間通信機能のサイズを測定した．測定条件を下記に，測定結果を図 8 に示す．

- OS 間通信機能のサイズを測定する場合は，OS 間通信で使用するオブジェクト数は 1 つとし，通知機能は未使用とする．
- 状態変数グループ通信における測定では，状態変数グループに属する状態変数を 1 つとする．

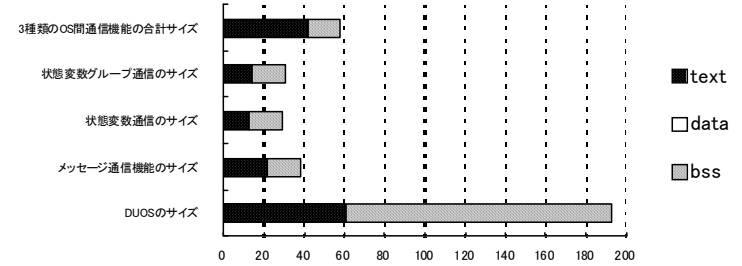

図 8　DUOS と OS 間通信機能のサイズ（単位は KB）

図 8 より，DUOS と OS 間通信機能の data 領域はほぼ 0B である．text 領域に関しては，DUOS のサイズよりも 3 種類の OS 間通信の合計サイズの方が約 20KB 小さく，bss 領域に関しては，DUOS のサイズよりも OS 間通信機能のサイズの方が十分に小さい．したがって，OS 間通信機能は DUOS よりも小さなサイズであり，組込みミドルウェアで許容できる範囲内にある．OS 間通信機能の bss 領域が約 20KB となっている理由は，データのやりとりを行うための共有メモリ領域や，OS 間通信用のアプリケーション固有のデータブロック領域を，静的に固定サイズで確保しているためである．デフォルトでは共有メモリ領域を 4KB，アプリケーション固有のデータブロックサイズを 8KB 確保しており，ユーザが修正することによって bss 領域のサイズを小さくすることが可能である．

7. 関連研究

DUOS と本研究で実装した OS 間通信では，ECU 統合を実現する方法として，1 つの OS 上で複数のアプリケーションを動作させ，そのアプリケーション間で通信を行う方法を採用している．しかし，ECU 統合を実現するその他の方法として，複数の OS 上でアプリケーションを動作させ，その OS 間で通信を行う方法（マルチ OS 技術）も考えられる．そこで，本章では，マルチ OS における OS 間通信の特徴をまとめ，本研究で実装した OS 間通信と比較する．既存の OS 間通信機能としては，ユーザモード OS における通信や OS 切替方式における通信が挙げられる．

ユーザモード OS は，マルチ OS 技術の 1 つであり，OS 上で別の OS を動作させる方式[8][9]である．ユーザモード OS を採用している Emblix 仕様では，ITRON 上で Linux が動作する Linux on ITRON 構成をとっている．Linux on ITRON における OS 間通信

では，ITRON 上のアプリケーションと Linux 上のアプリケーションで独自の OS 間通信用 API を定義している．また，通信機能としては，FIFO 方式と共有メモリブロック方式が定義されている．FIFO 方式は，内部にバッファを持ち，データをコピーして相手 OS に送信する．FIFO 方式には，ITRON，Linux 上のアプリケーションが書込み，読出し待ち状態に遷移する機能がある．また，共有メモリ方式は，OS 間で共通にアクセスできるメモリ領域を介してデータの受け渡しを行う．共有メモリ方式では，排他制御の仕組みとしてロック/アンロック用 API が提案されており，ロック待ち状態に遷移する機能がある．本研究で実装した OS 間通信は，OSEK，ITRON アプリケーション用の独自 API を有しており，Emblix 仕様と同様である．メッセージ通信機能は，Emblix 仕様の FIFO 方式と類似の機能であり，ITRON アプリケーションの送受信待ち状態への遷移機能も Emblix 仕様と類似している．しかし，OSEK アプリケーションへの通知機能と類似の機能は Emblix 仕様には存在しない．また，状態変数通信，状態変数グループ通信と類似の機能も Emblix 仕様には存在せず，逆に Emblix 仕様の共有メモリ方式は本研究で実装した OS 間通信には存在しない．共有メモリ方式と比べて状態変数通信や状態変数グループ通信のデメリットは，データのコピーを行うため，直接メモリ領域にアクセスする共有メモリ方式よりもオーバーヘッドが大きくなってしまう点である．逆に，メリットは，ロック/アンロック用 API が不要な点であり，デッドロックや優先度逆転の問題が発生することはない．

OS 切替方式において研究されている OS 間通信[10]では，一般的な OS が備えているファイル API（open, read, write）が用意されている．また，OS 切替方式では，socket に準拠した通信手順を採用しており，相手の通信状態を確認してから通信を行うコネクション型の通信を行っている．それから，OS の切替を依頼する機能が備わっていて，通信相手が休止中から復帰してデータの送受信ができるようになっている．本研究で実装した OS 間通信は，コネクション型の通信ではない．また，アプリケーションの切替を依頼する機能は備えていない．本研究で実装した OS 間通信がコネクションレス型の通信を行うことは，コネクション型の通信に比べて信頼性に欠けるが，オーバーヘッドを小さくできる利点がある．また，本研究で実装した OS 間通信がアプリケーションの切替を依頼する機能を備えることに関しては，通信のリアルタイム性の確保につながるため有用である．

8. おわりに

本論文では，ECU 統合向け OS である DUOS における OS 間通信の要件や仕様，その性能評価について述べた．OS 間通信の要件では，OSEK COM 仕様や AUTOSAR COM 仕様といった既存の車載システムにおける通信方式をベースとすることや，OSEK 仕様や ITRON 仕様の通信機能に適合させるために，通知機能や待ち状態への遷移機能を追加することについて説明した．そして，OS 間通信の仕様では，その要件に基づいて決定したメッセージ通信，状態変数通信，状態変数グループ通信の仕様について詳しく説明し，OS 間通信の評価では，実装した OS 間通信のデータ送受信処理時間やプログラムサイズの評価に関して述べた．

本研究では，OS 間通信を実装することによって，ECU 間で行っていた通信を DUOS においても代替することが可能になった．また，実機を用いて，OS 間通信のデータ送受信処理に要する時間や，OS 間通信機能のコードとデータサイズを測定し，組込みミドルウェアとして許容できる範囲内にあることを確かめた．

今後の展開としては，DUOS と共に OS 間通信に対するメモリ保護対応やマルチコア対応の設計，実装，評価を行っていく予定である．その評価の際に，今回の測定結果は OS 間通信のメモリ保護対応やマルチコア対応による性能への影響を知る上で，基準値として有用である．

謝辞 本研究を進めるに当たり，ご協力くださった名古屋大学大学院情報科学研究科附属組込みシステム研究センターのメンバー各位に厚く御礼申し上げます．

参考文献

1) OSEK/VDX: OSEK/VDX Operating System Version 2.2.3 (2005).
2) (社)トロン協会: μITRON4.0 仕様 Ver. 4.02.00 (2004).
3) 松原豊，本田晋也，富山宏之，高田広章: 時間保護のためのリアルタイムスケジューリングアルゴリズム，情報処理学会研究報告, Vol.48, No.SIG8, pp.192-202 (2007).
4) 松原豊，本田晋也，富山宏之，高田広章: リアルタイムアプリケーション統合のための柔軟なスケジューリングフレームワーク，情報処理学会研究報告, Vol.49, No.10, pp.3508-3519 (2008).
5) 松原豊，本田晋也，高田広章: 割込み処理を含むリアルタイムアプリケーション統合のための階層型スケジューリング，情報処理学会研究報告, Vol.2009-EMB-14, No.7, pp.1-9 (2009).
6) OSEK/VDX: OSEK/VDX Communication Version 3.0.3 (2004).
7) AUTOSAR: Specification of Communication Version 3.0.2 (2008).
8) Emblix: Linux における RTOS とのハイブリッド構成に関する仕様書（第 1 版）http://www.emblix.org/doc.html;apr;2010.
9) 金田一勉: Linux on ITRON ハイブリッド構造の実装, Interface 別冊付録, CQ 出版社 (2002).
10) 江口悠利，中川智尋，太田賢，竹下敦: 携帯端末向けサスペンド機能利用型マルチ OS 環境における OS 間通信方式，情報処理学会論文誌, Vol.2008, No.32, pp.215-220 (2008).

Vol.2010-SE-168 No.3
Vol.2010-EMB-17 No.3
2010/6/1

1 OSEK/VDX: OSEK/VDX Operating System Version 2.2.3 (2005).
2 (社)トロン協会: μITRON4.0 仕様 Ver. 4.02.00 (2004).
3 松原豊, 本田晋也, 富山宏之, 高田広章: 時間保護のためのリアルタイムスケジューリングアルゴリズム, 情報処理学会論文誌, Vol.48, No.SIG8, pp.192-202 (2007).
4 松原豊, 本田晋也, 富山宏之, 高田広章: リアルタイムアプリケーション統合のための柔軟なスケジューリングフレームワーク, 情報処理学会論文誌, Vol.49, No.10, pp.3508-3519 (2008).
5 松原豊, 本田晋也, 高田広章: 割込み処理を含むリアルタイムアプリケーション統合のための階層型スケジューリング, 情報処理学会論文誌, Vol.2009-EMB-14, No.7, pp.1-9 (2009).
6 OSEK/VDX: OSEK/VDX Communication Version 3.0.3 (2004).
7 AUTOSAR: Specification of Communication Version 3.0.2 (2008).
8 Emblix: Linux における RTOS とのハイブリッド構成に関する仕様書（第 1 版）
http://www.emblix.org/doc.html;apr;2010.
9 金田一勉: Linux on ITRON ハイブリッド構造の実装, Interface 別冊付録, CQ 出版社 (2002).
10 江口悠利, 中川智尋, 太田賢, 竹下敦: 携帯端末向けサスペンド機能利用型マルチ OS 環境における OS 間通信方式, 情報処理学会論文誌, Vol.2008, No.32, pp.215-220 (2008).