組込みシステムに適したコンポーネント指向開発環境

Component Based Development Environment for Enbedded Systems.

上野　真路†
Shinji UENO
†有限会社ユー・システムズ
U-SYSTEMS INC.
ueno@usys-assoc.org

本稿では、組込みシステムに適したコンポーネント指向における開発環境について提案する。主に複数のプロセッサ構成による分散システムを対象にした、分散コンポーネントフレームワークについて述べる。また本開発研究による成果目標を、幅広い一般技術者のための実用システムの開発としており、システム導入のハードルを低くするためグラフィカルな開発環境を基本としている。試験的ではあるが導入例についても紹介する。

1 はじめに

組込み分野において、比較的要求仕様に余裕がある制御対象にまで、複数のプロセッサを用いた分散制御が用いられるようになってきている。このことの背景のひとつには、チップの低価格化、出荷後の新機能拡張性付加、ネットワーク技術の浸透、開発効率の重視などが考えられる。このトレンドの先には、ＰＣの世界で普及している分散型コンポーネント指向による優れた開発環境が、組込み分野でも出現することと容易に推測される。実際その傾向が見うけられる。しかしながら、組込み分野での分散コンポーネント環境の導入には、ＰＣ界での技術（WindowsDCOM、CORBA、JavaBeansなど）をそのまま移植することは無理が多いと考えられる。組込分野は定義にもよるが、限られたリソースの１チップボードのシステムから、よりＰＣに近い環境まで種々考えられる。これら幅広いターゲットシステムに対応できるように、次の２項目が特に考慮された組込みに適した分散コンポーネント仕様の設計、そして実装コードの試作開発を行った。

１、さまざまなプロッセッサ上へ移植・動作可能なこと（軽量シンプルであること）

２、リアルタイム性の確保（時間管理機能）

＊本開発研究は独立行政法人情報処理推進機構（IPA）による、2003年度/2004年度未踏ソフトウェア創造事業の支援を受けている。

2 想定するニーズ

本開発が現時点で対象としている、主な実用的なニーズは以下である。

1. ロボットなどの機械制御システムにおける、分散制御ネットワークシステム。
2. より一般的な組込み分散制御、自立分散制御における協調制御、省配線化、CPUパワー分散のためのネットワークシステム。

3 各ニーズに対する現状と課題

現状、ニーズ１には実用的なニーズが多くある。また多くの分散システムの独自規格が存在しており、プロセッサチップ単体ベースの比較的小さなシステムの分散システムでは、私は混沌とした状態であると考えている。ＣＡＮ、Ｉ２Ｃなどは、チップ内に通信制御回路が搭載されることにより普及しているといえるが、しかしながら多品種のプロセッサチップが存在し活用されている組込みの世界では、ある意味限られたチップ上での閉じた規格であると、捉えることが出来るとも考えている。一方、組込といわれる世界と比較して大きなシステムを対象とするＦＡ・ＰＡの世界では統一規格の導入がかなり進んでいるようである。

ニーズ２に関しては、チップの低価格化・高性能化とともに、組込みシステムへの要求仕様の高度化が進んでいる現状、その要求に効率的に対応するた

めに、組込みシステムでも分散化の対応が進みつつある。今後、より一層リソースが制限される組込みシステムでも、効率的な分散システムの構築が可能な、汎用性・移植性に優れたオープンソースの統一仕様が期待されていると考える。

4 開発内容

分散システムを構築する目的を詰めてゆくと以下の２つに大別できると考える。

ア： データの共有

イ： CPUパワーの分散

しかしながら、分散コンポーネント指向においては、両者の明確な区別はなくなり同一のフレームワークで対応できると考えている。組込みシステム用に最適化した分散コンポーネントシステムを設計するにあたり、次のように基本となる仕組みを明確にすることを行い、リアルタイム性・軽量であるシステム開発に取組んだ。

「分散コンポーネントシステムは、いわゆるＲＰＣ（Remote Procedure Call）を使ったデータ（関数の戻り値・引数パラメータ）の通信接続を行う別の表現の一つである。加えて、通信する際のデータのシリアル変換・復元処理のためのマーシャリングに、分散コンポーネントウェアの本質がある」と、私は考える。

図１に、プログラムコードの利用例（概略イメージ）を示す。一方のCPU上にあるオブジェクトへ接続するに、自信のCPU上にプロキシを生成しそのプロキシと、相手方CPU上に存在するスタブとがRPCにより通信接続することで、同じプロセス内コールに近い形で利用できるようになる。

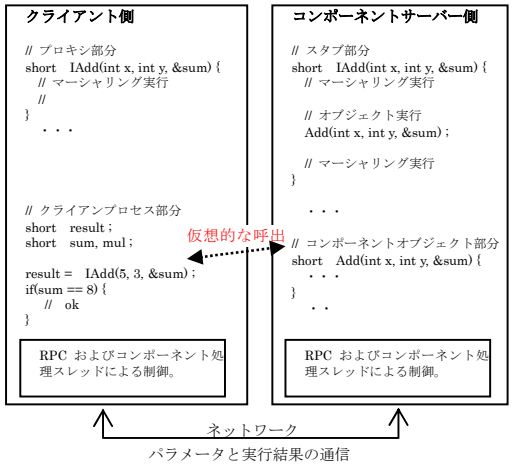

図１ クライント － サーバー間内での
プログラムコード利用例の概略

4.1 さまざまなプロッセッサ上へ移植・動作可能とするための仕様

ここでWindowsDCOMなどと違い、軽量な仕組みとするための工夫（適切な条件付け・制限）が必要とされる。また、次に説明する、リアルタイム性確保のための仕組みを導入することでもコード量の削減、仕組みの単純化に活かした。通信接続に使うハード仕様は、ほぼすべてのプロセッサに搭載されているといえるUARTを基本とすることにより、多種のCPUが存在し使われる組込み世界において、低コストに通信接続が可能となると考えている。一般仕様といえるUARTはFPGAへも容易に導入できるため、CPUチップのみではなくより多くのデバイスを用いた分散制御ネットワークが、比較的容易に可能とすることが出来る。今回、電気的な仕様については、図２に示したRS-485を用いた２線式LANを採用した。また、同一基盤上などの小エリア内のLAN構築には、図３に示すようにTTLレベルの信号を用いたより簡易的な構築が可能でもある。

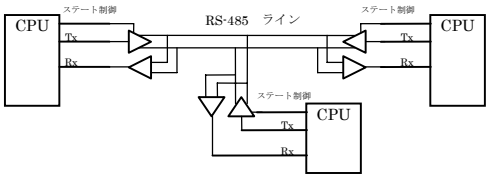

図２： RS-485 による LAN システム

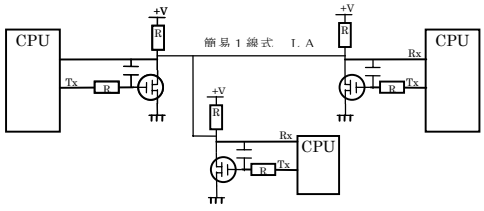

図３： 簡易的な LAN システム

4.2 リアルタイム性（時間管理機能）について

図４は、４つのＣＰＵを用いた分散システム例である。ＣＰＵ２（サーバー）・ＣＰＵ３（サーバー）

がインスタンスを持つ分散コンポーネントオブジェクトを、ＣＰＵ１（クライアント）から対応するインターフェイスを通じ利用している構成となっている。ここでは、前述したスタブとプロキシにより、オブジェクトのパラメータをマージャリングして通信している。分散オブジェクトのコールは、自身のメモリ上の関数コールと比較して、オーバヘッドが大きく、また、同時に返信されてくる応答時間には不確定要素が多くある。このことに対する簡易かつ明瞭な対策としてCPU０を設けた。表２は、オブジェクトA・オブジェクトB・オブジェクトCの実行処理所要時間と、CPU１からの利用要求度・頻度について表している。表１のように、実際にクライアントが制御情報としてコールしたい周期と、オブジェクトの処理時間には大きな差がある場合が多く、クライントプログラムが求める頻度以上に、インターフェイスとオブジェクトを繋げても（RPCによるスタブ⇔プロキシ通信）LANの帯域を浪費するばかりである。そこで、CPU０により無駄なRPC接続が行われないように、通信スケジュール管理をしている。CPU０内には、あらかじめテーブルデータとして、表１のような情報のほか、分散システム上のどのCPUにどの分散オブジェクトをもち、誰がそのオブジェクトを参照するかなどの情報を持たせている。そのテーブルデータをもとに、各分散オブジェクト及びその参照オブジェクトへ、順序よく接続処理を実行して行く。（すなわち、時間管理が可能となっている）。

4.3　機械制御などでよくある制御プログラムのパターン例を用いたシステム概念について

図５は、ひとつのプロセッサ内に、３つのタスクを走らせ、互いに協調してひとつの制御をおこなっているところを示したものである。ここで、各タスクを３つのプロセッサに分散させたシステムを考えると、図６のように、それぞれのプロセッサ（タスク）をコンポーネントプロックとして表現可能である。

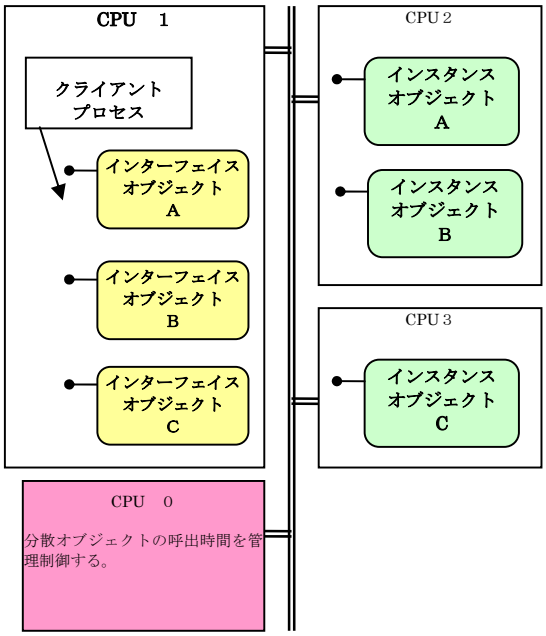

図４： 分散システム例

表１

	実処理 所要時間	クライント側が求める接続周期 １回／t
オブジェクトＡ	1msec 以内	t＝ 200msec
オブジェクトＢ	1msec 以内	t＝ 1 sec
オブジェクトＣ	100msec 以内	t＝ 500 msec

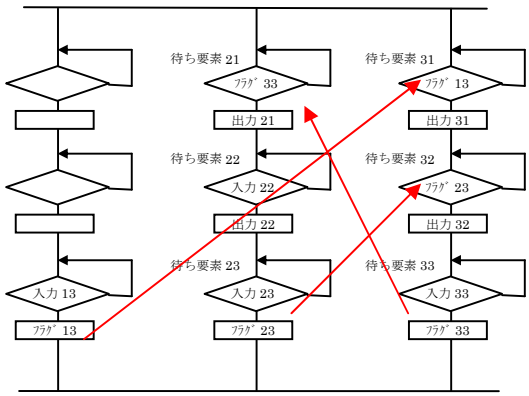

図５： タスク間の連携イメージ

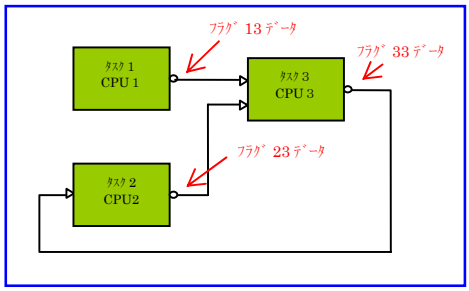

図６：グラフィカルなコンポーネントによる表現

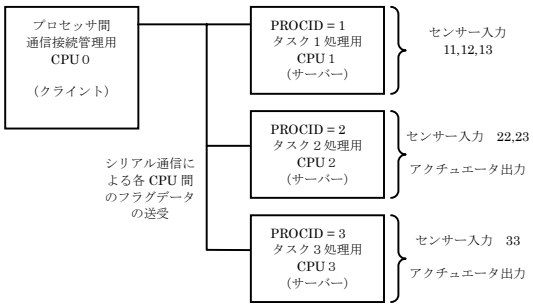

図７： 分散システムのプロセッサ構成

図７は、新たに各コンポーネント（プロセッサ）間の通信接続スケジュール管理に、CPU0を加えたシステムである。前述しているがシリアル通信ライン上においては、物理的に常に１つの接続しか成立しない。よって各CPUが任意のタイミングで送信接続を試みようとすると、さまざまな処理が必要となってくる。その処理をマイコンリソースのみで解決しようとすると無理があるため、通信スケジュール管理を行わせるCPU０を追加し、クライアント的な動作をさせている。ここで、図６に示されたブロック表記が、システム上の各コンポーネント接続状態を示すグラフィカルなプログラムとなる。

4.4 本件で扱う分散コンポーネントの定義・仕様および指針について

１、規定する通信接続インターフェイスを持つ実行可能な実装オブジェクトをもつものとする。（すなわち、規定したインターフェイスを持っていることが条件であり、ハードまで含んだ装置全体であってもよい）。

２「メソッド」「イベント」へのインターフェイス。（プロパティはメソッドとして扱う。イベントは原則発するだけで返答を求めない。またイベントには、任意長のデータを添付可能とする）

３、原則として、コンポーネントはサーバー（受動）的な動作をするものとする。

４、主な開発言語はＣ言語とし、Ｃ＋＋などオブジェクト指向言語は避ける。

4.5 オブジェクト識別IDについて

1. PROCID： プロセッサID：1byte

ネット上に接続されている、プロセッサ識別用

2. CLSID： コンポーネントクラスID：2byte

同一プロセッサ上に実装された複数のコンポーネントオブジェクトのまとまりの識別記号。原則として、グローバルユニークID扱いとする。

3. OBJID： コンポーネントオブジェクトID:1byte

CLSIDにより識別される、オブジェクト集合内の各オブジェクト識別用

4. EVNTID： 2byte

イベント識別

4.6 コンポーネントのグラフィカルなユーザーインターフェイスについて

図８： メソッドのグラフィカルインターフェイス

図８は、グラフィカル開発環境での、クライアン側のメソッドへのインターフェイスプロキシと、サーバー側のインターフェイススタブである。現時点では、標準インターフェイス（short型の入力８個・出力８個）のみ開発した。ブロック左側の入力端子には、分散接続されたオブジェクトの識別IDを設定する端子である。また、左側には、時間管理機能のためのパラメータ端子があり、RPCエンジンがスケジュール管理するためのパラメータとして活用するように設計している。そこにはコンポーネント接続の不要な接続を減らすための接続希望サイクルパラメータ、実際の通信接続サイクル時間（msec）の出力、通信エラーカウンター出力端子である。

図９は、イベント送受のためのインターフェイスプロキシと、インターフェイススタブである。イベントには任意長のデータを持たせる仕様としたが、１３個のshortパラメータを持たせたものを標準インターフェイスとして開発した。またmodeパラメー

図４： イベントのグラフィカルインターフェイス

タにより、イベント送信の手動／自動切替など行える仕様としている。

4.7 開発システムの検証

図１０は、研究開発に用いた検証システムの構成の1つであり、プロセッサを３つ使っている

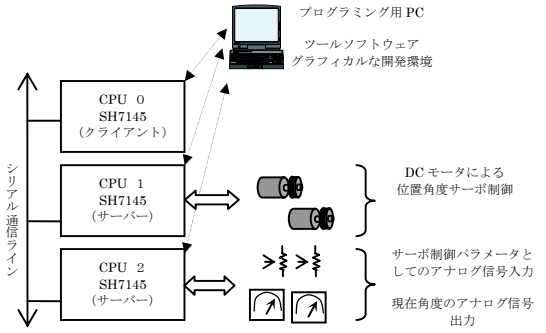

図１０： 検証に用いたシステム構成

● CPU１は、ｷﾞﾔ付DCモータの角度位置サーボ制御行うためのコンポーネントとなっている。

● CPU２は、外部からのボリューム電気信号を取り込み、CPU１でのサーボ制御用パラメータとし、同時にCPU1から実角度のパラメータを受け取りアナログメータへ出力している。

● CPU０は、主にクライアントとして通信スケジュール管理などを行っているが、同時に制御角度目標値としてSin波演算コンポーネントを実行し、CPU2へそのパラメータと供にイベント送信している。

図１１には、上記制御をさせた各CPUのグラフィカルプログラムを示す。

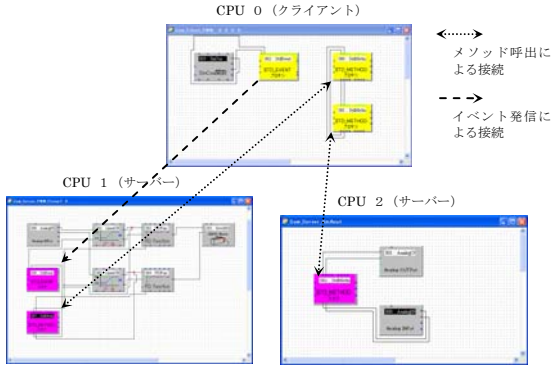

図１１： 検証に用いたプログラム画面

5 まとめ

組込みを主な対象としたコンポーネント指向開発環境を提案した。本開発研究は実用的なシステムを強く意識し行われている。また、コンポーネント指向による開発作業は、コンポーネント部品開発と、その部品の組合わせによるアプリケーション開発の２つに分化してくる。今後もコンポーネントフレームワークの機能強化改良とともに、有用なコンポーネントの平行開発を続けて行く。今回提案した開発環境の開発のきっかけは、工業製品に必須といえる組込システム向けに適した同様なツールは当初見受けられず、自ら開発しこのようなツールの必要性を問いたいと思ったことにある。日本から有力なツールを発信したいという強い思いのもと、継続開発に力を注ぐ思いである。最後に、導入例についても紹介する。ユーザーにとって導入し易いグラフィカルな開発環境・開発済みの有用な制御コンポーネント類などの特徴を有することから、岐阜工業高等専門学校電子工学科の実験授業用教材として採用された。

参考文献

[1] 上野真路、遠藤登、稲葉昭夫、分散型組込制御システムのComponent BasedによるGUI開発ツール ,SWEST5予稿集 pp.18-22, (2003).

[2] 上野真路、遠藤登、稲葉昭夫、"分散型組込制御システムのComponent BasedによるGUI開発ツール ,"IPA 2003年 未踏ソフトウェア創造事業成果報告集.

[3] 上野真路、遠藤登、北川秀夫、福永哲也、畑中裕司、森貴彦、吉田昌春、教育用マイコン制御システムの開発研究, ソフトピアジャパン共同研究報告書2005 Vol.9, (2005)