3V-8

実時間オペレーティングシステムR²の通信管理方式

杉村 邦彦　白濱 和人　友田 和伸 ((株)ダイヘン)
大久保 英嗣　津田 孝夫　楠田 修三　小林 正典(京都大学工学部)

1. はじめに

産業用ロボットを代表とするFA(ファクトリーオートメーション)システムの高機能化と、システム保守の容易性の要求により、産業用システムは複数プロセッサによる機能分散制御形態をとることが一般的となってきた。これはまた、CPU価格の低下によるところも大きい。こういったシステムでは、全プロセッサに渡るプログラム間でタイミングを取りながら、総合的に作業を進めていかなければならないという複雑さがある。そこで実時間オペレーティングシステムR²は、このような複数プロセッサ上に分散されたシステム制御プログラムにおける、タスク間の同期と通信機能のサポートを行うことによって、アプリケーションプログラム記述の負担を軽減させている。本稿では、その設計思想と実現法について述べる。

2. R²の同期と通信制御の特徴

産業用ロボットなどの、各種産業用システム制御装置に組み込むことを目的としたR²は、機能分散制御システムにおける通信制御として、次の3つを重視している。

(1) 複数の通信装置の同時実行を可能とする。

(2) 多種多様の通信ドライバの作成を容易にする。

(3) 通信装置の存在をアプリケーションプログラム側に意識させない。

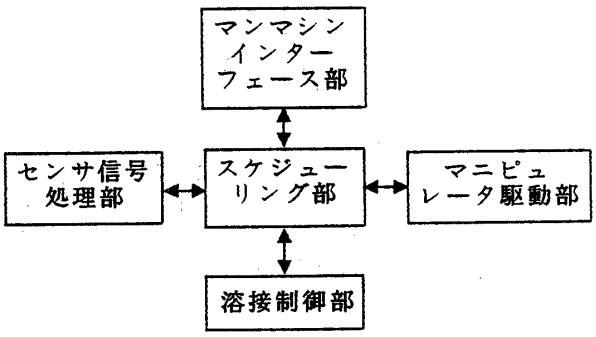


図1 機能分散制御システム

機能分散システムにおいては、複数の機能間で同期と通信を行う必要があるため、1つのプロセッサが複数のプロセッサと接続される可能性がある(図1)。この時リアルタイムシステムとしては、複数のプロセッサとの通信を同時に進める必要がある。このためR²では、通信ドライバをOS核に持たせている。これは入出力ドライバと同様の考え方であるが、通信機器としての固有の処理があるため、入出力ドライバから独立させている。タスクが、同期又は通信の要求を発行すると、通信ドライバに制御が渡る。通信ドライバは他のタスクと並行に動作することができるので、このタスクがまた別の通信を要求することができる。既に動作中の通信装置に対して通信要求を行った場合は、当該通信装置に対して通信要求のキューが作られる。

通信装置と通信方式は、各システムの要求によって様々であるため、通信ドライバは、ユーザが設計しなければならない。そこでR²では、通信ドライバの設計を容易にするため、通信ドライバ自体をC言語で記述することを可能としている。これによって、他システムへの通信ドライバの移植も容易となる。さらに、通信ドライバとR²核とのインターフェースを容易にするため、各種ライブラリ関数を用意している。

ユーザのプログラムを構成するタスクには、全システムに渡って一意的に定義された論理番号が付けられる。アプリケーションプログラムでは、この論理番号を直接指名することによって、タスク間の同期と通信の記述を行う。すなわち、通信機器を意識させない抽象化された表現によって、タスク間の同期と通信を記述することができる。これにより、通信の相手タスクが、同一プロセッサ上あるいは他のプロセッサ上どこにあっても、同じ形式の記述となるため、移植性の高いアプリケーションプログラムを実現することができる。

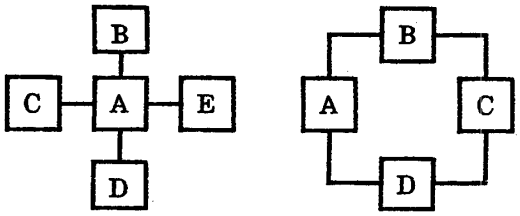
3. 通信可能なシステムの形態

R²が通信可能なシステムの形態を図2に示す。R²では、通信ラインによって直接接続されたプロセッサ間においてのみ通信を可能としている。例えば、図2(a)で、プロセッサAとプロセッサB上のタスクは通信可能であるが、プロセッサB上のタスクがプロセッサC上のタスクを、直接指名して通信することはできない。この場合は、プロセッサA上のタス

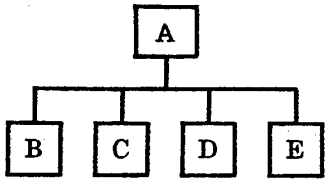
The Architecture of Communication Control in Real-Time Operating System R²

KUNIHIKO SUGIMURA, KAZUTO SHIRAHAMA and YASUNOBU TOMODA (Daihen Corporation)
EIJI OKUBO, TAKAO TSUDA, SYUZO KUSUDA and MASANORI KOBAYASHI (Kyoto University)

クが一旦通信を中継しなければならない。



(a) スター結合　　(b) リング結合



(c) バス結合

図2 通信可能なシステム形態

4. システムコール

R²の同期と通信制御におけるシステムコールを表1と表2に示す。

表1 同期用のシステムコール

1. signal (task_no, signal_data)
2. wait (signal_data, condition)
3. test ( )
4. reset (signal_data)

表2 通信用のシステムコール

1. send (task_no, msg_ptr, msg_len, wait_flag)
2. receive (msg_ptr, wait_flag)
3. reply (task_no, reply_ptr, reply_len)
4. forward(from_task, to_task, msg_ptr, msg_len)
5. createmailbox(mail_ptr, mail_len, capacity, reply_ptr, reply_len)

表1のsignalとwaitによって、タスク間で16ビットのシグナルの送受を行い、同期のタイミングを取る。testは自タスクのシグナル受信状況を確認するために用いられる。シグナルの待ち条件として、ビットのANDまたはORが指定できる。メッセージの送信は、表2のsendによって行われる。受信側のタスクは、receiveを発行することによって送られてきたメッセージを受信する。引数wait_flagによって、メッセージの受信があるまでタスクの実行を一時停止させるか否かを選択することができる。sendの引数wait_flagは、受信側からの返信メッセージを必要とする時に指定される。この時送信側のタスクは、メッセージが返信されるまで待ち状態となる。受信側からの返信の発行は、replyによって行われる。forwardは、返信の発行を子タスクに任せる時に使用される。メッセージの送信、返信を受けるタスクは、いずれもあらかじめcreatemailboxを発行して自タスク用のメイルボックスを作成しておく必要がある。

5. 同期及び通信制御と通信ドライバ

同期及びメッセージ通信で指定された相手タスクの番号により、R²はそれがどのプロセッサ上のタスクであるかを判断する。そして他のプロセッサ上のタスクである場合には、当該プロセッサとの通信に使用する通信装置を割り出す。通信装置が決定すると、対応する通信ドライバに対し通信要求を発行する。このイメージを図3に示す。

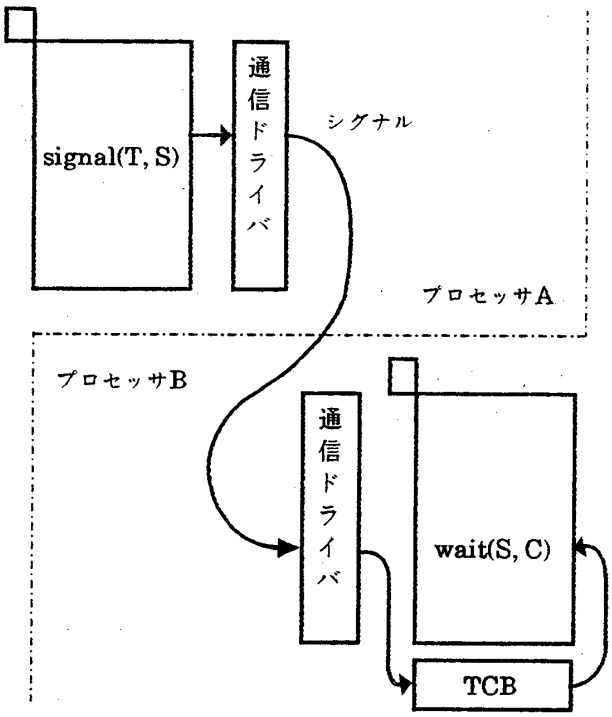


図3 プロセッサ間タスクの同期イメージ

6. おわりに

以上のアーキテクチャは、プログラマから煩雑な通信装置の概念を排除し、シンプルなプログラム設計の実現に寄与するものと確信している。現在R²は、RS232Cによる通信を標準的な通信手段として通信プロトコルを決め、そのための通信ドライバを持っているが、パラレルバス等による、より高速な通信手段のサポートも行っていく予定である。