OS/2 Warp のリアルタイム機能の拡張

5M－4

根岸　康，立沢　昌弘，本田　良司，石川　繁樹
日本アイ・ビー・エム (株)

1　背景

FA(Factory Automation) の分野では、FA 機器を制御するために PLC(Programable Logic Controler) と呼ばれるシーケンス制御専用装置を用いているが、近年では専用装置を用いずに PC(Personal Computer) から直接 FA 機器を制御する "パソコン PLC" が登場し、注目を浴びている [1]。一般にパソコンによる制御は価格や拡張性等メリットも大きいが、現状のシステムはこれらのメリットを十分に生かしていない。我々は、パソコン制御向けの OS の満たすべき用件について考察し、パソコン制御向けに OS/2 Warp のリアルタイム性能を拡張した OS(以下、OS/2 Warp RT と呼ぶ) の実現を行なっている。

2　パソコン制御の OS に必要な性能

2.1　パソコン制御のメリット

市販の PC 用カードやソフトウェアを利用して、使いやすく拡張性の高いシステムを安価に構築したいという要求が高まっている。現状の PLC の問題点、すなわちパソコン制御に対する要求は、大きく次の２つにまとめられる。

コスト
PC の価格は年毎に大きく下がっており、価格性能比が高い。ソフトウェアに安価な市販品が利用できる。

拡張性
PLC の入出力インタフェースや OS はメーカー独自のものが多く、市販の拡張カードやソフトウェアを利用することは難しい。PC のインタフェースは公開されており、市販の拡張カード、ソフトウェアを利用できる。

2.2　現状のパソコン制御の問題点

現在パソコン制御に利用可能な OS は、以下のように分類できる。

リアルタイム OS
LynxOS, QNX 等制御システム向け OS。通常 UNIX をベースに作成されている。リアルタイム性能は他の OS に比べると優れている。市販の拡張カードを利用する場合、デバイスドライバを作成する必要がある。DOS/Windows 用ソフトウェアは使用できない。価格は、開発環境を含めると百万から数百万円。

DOS/Windows
DOS および Windows3.1。スケジューリングをアプリケーション自身が行なうことにより、制御システムを実現する。このため、制御プログラムと DOS/Windows 用ソフトウェアを並列に動作させることは難しい。市販の PC 用のほとんどのカードを利用できる。

DOS 系汎用マルチタスク OS
Windows95、WindowsNT および OS/2 Warp。Windows95 を除きリアルタイム性は一応考慮されているが [2][3] 制御システムを実現するには不十分。特別なソフトウェア無しにシステムを実現でき、制御用ソフトウェアと市販の DOS/Windows プログラムを同時に利用できる。また、PC 用のほとんどの市販カードをデバイスドライバを新規に作成すること無しに利用できる。

UNIX 系汎用マルチタスク OS
Linux、FreeBSD 等 UNIX 系の汎用システム。リアルタイム性は特に考慮されていない。市販のカードはハードウェア市販のプログラムは、DOS 系と比べると少ない。

このように現状で利用可能な OS は、それぞれ長所、短所を持っている。ここでは、特にパソコン制御に期待が高い拡張性、すなわち市販カードおよび市販ソフトウェアの利用可能度に着目する。リアルタイム性能と拡張性との相関図を書くと図 1のようになる。

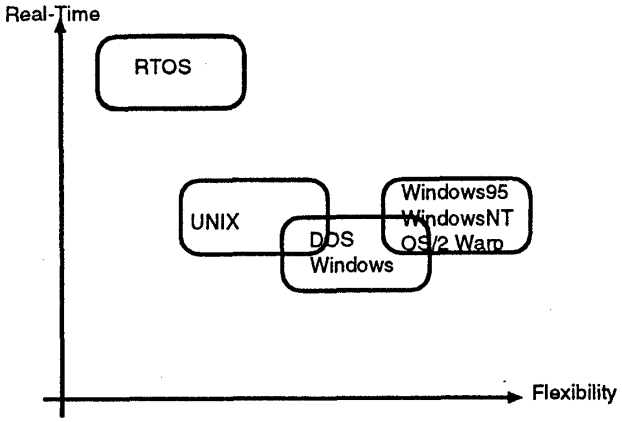

図 1: リアルタイム性能と拡張性の相関図

この中で用途に応じた OS を選ぶことになるが、図 1 中で左上の領域をカバーする OS がパソコン制御に適した OS ということになる。この左上の領域をカバーするためには、リアルタイム OS の拡張性を高める方法と、DOS 系マルチタスク OS のリアルタイム性能を高める方法があり、両方の試みがなされているが十分ではない。

我々は、OS の変化の速さを考えると前者の方法で DOS プログラムの互換性を維持していくのは困難と考え、DOS 系マルチタスク OS のリアルタイム性能を高

Real-Time Enhancement of OS/2 Warp
Yasushi NEGISHI, Masahiro TACHIZAWA, Ryoji HONDA, Shigeki ISHIKAWA
IBM Japan, Ltd

める方法を選んだ。

ベースとするOSにはDOS系マルチタスクOS 3つの内、OS/2 Warpを選んだ。最初にリアルタイム性が考慮されておらず、16ビットプログラムの実行中実行権の横取りができないWindows95を除外し、Windows NTとOS/2 Warpを比較した。この両者ともリアルタイム性を考慮しているが、実際の性能はパソコン制御には不十分であったので、必要とするメモリの量及びDOSプログラム実行の互換性を検討し、OS/2 Warpを選んだ。

3 OS/2 Warp RT

前章の検討結果に基づき、OS/2 Warpのリアルタイム性能を拡張し、OS/2 Warp RTと呼ぶシステムを実現した。本章では、実現したシステムの実現方針、変更箇所、得られたリアルタイム性能およびその制約について述べる。

3.1 実現方針

実現目標として、次のものを挙げた。

目標1 リアルタイムプロセスも通常のプログラムと同じ方法でプログラミングできるようにする ([3]参照)。

目標2 既存のデバイスドライバはそのまま利用する。

目標3 リアルタイムプロセスと通常のプロセス間の同期を可能にする。

目標4 変更箇所および与える影響を最小にする。

OS/2 Warp RTの実現方針として、次のようなものが考えられる。

方針1 1つのデバイスドライバとして実現する。

方針2 OS/2 Warpを1つのプロセスとして動かすOSとして実現する。

方針3 OS/2 Warpのカーネルの横取り性を順次高める方法で実現する。

方針1はリアルタイムプロセスのプログラミングにデバイスドライバ作成の知識を要するようになってしまい、方針2では変更箇所が多く既存のデバイスドライバを利用することが難しくなるので、我々は方針3をとることにした。

具体的な作業としては、リアルタイムプロセスの実行の様子をチェックするテストプログラムを走らせた上で、このプログラムの実行を妨害するテストプログラムを走らせ、リアルタイム性が守られなかった部分を調べ、必要な修正を行なう。変更の大半はOS/2のカーネルやデバイスドライバ中で排他的に実行が長く続く部分に実行を他に譲るための関数(Yield)を挿入することになる。

3.2 変更箇所

OS/2 Warpのカーネルおよびデバイスドライバに対し、現在までに行なった変更は次のようなものである。

1. 制御プログラム用に新たな優先度を設けた。
2. 実時間制約が守れなかった場合にプログラムに確実に通知できるようにした。
3. Yieldが実行された際に確実にリアルタイムプロセスが実現されるようにした。
4. カーネル中の横取り性が不十分な部分にYieldを挿入した。
5. デバイスドライバ中の横取り性が不十分な部分にYieldを挿入した。

全体の変更箇所は40箇所程度でOS全体には大きな影響を与えずに変更することができた。

3.3 リアルタイム性能

実現したOS/2 Warp RTのリアルタイムに関する性能を測定した。測定は、IBM ThinkPad530(CPU:DX/4 100MHz、メモリ20MB)に通常のOS/2 WarpおよびOS/2 Warp RTをインストールした環境で行なった。

測定の方法は、8m秒周期で動作するテストプログラムをリアルタイムプロセスとして動かした上で、負荷をかけるための典型的なプログラムを走らせて、テストプログラムの8m秒周期がどの程度守られているかを測定した。テストは周期2000回を5回繰り返して測定し5回の平均値を用いた。負荷をかけるためのプログラムには以下のものを用いた。

CPU負荷 無限ループを回るプログラム

ファイルI/O 500kバイト程度のファイルをコピーし続けるプログラム

コンパイル テストプログラム(C言語とアセンブラ言語で500行程度)のmake

以下に、実時間制約が守れなかった確率(%)および実行間隔の最大値(m秒)を示す。

負荷	OS/2 Warp	OS/2 Warp RT
CPU負荷	0.00% (-)	0.00% (-)
ファイルI/O	0.63% (32m秒)	0.01% (16m秒)
コンパイル	0.42% (16m秒)	0.00% (-)

3.4 今後の課題

汎用OSのカーネル横取り性を高めていくためには、多様な環境でテストを行なっていく必要がある。我々は、OSの実現の際にも開発中のOSを使う、βテストを行う等の方法でこの点に対処しているが、更に実際のシステムに利用しながら性能向上を図っている。

汎用OSを母体としたリアルタイム化であるため、リアルタイムアプリケーションを構築する際に、フェールセーフを考慮したプログラミングは必要である。例えば、割り込みが短期間に集中した場合等に対応する必要がある。またデバイスドライバの開発ガイドを守らず長期間排他的に実行を続けるデバイスドライバの利用は回避する必要がある。

4 まとめ

少ない変更でOS/2 Warpのリアルタイム性能を高めたOSを実現できた。これにより拡張性の高いパソコン制御システムを構築できる。今後の課題としては、より多くのテストを行うことによりリアルタイム性能をより高めていくこと、複数のリアルタイムプロセス間のスケジューリングの方法を改善すること[4]、およびリアルタイムプロセスと通常プロセスの同期の手段をより充実させていくことにある。

文献

[1] "パソコン PLC は実現するか," 日系メカニカル 1995.10.2 No. 464.

[2] "Real-Time Systems with Microsoft Windows NT, " Microsoft Technology Brief 1995.

[3] "OS/2 Warp Control Program Programming Guild," IBM Corp. Manual.

[4] John A. Stankovic, Macro Spuri, Marco Di Natale and Giorgio C. Buttazzo, "Implications of Classical Scheduling Results for Real-Time Systems," IEEE COMPUTER, Jun 1995.