マイクロコン68000用モジュラーOS

筒井良義* 横畑静生* 前田多可雄* 矢部栄一** 石田正浩***
(株)日立製作所　システム開発研究所* 武蔵工場** 那珂工場***

1. はじめに

16ビット・マイクロコンは、その高性能化に伴い、簡単な機器制御から、複雑なリアルタイム制御システムにまで広範囲に利用されるようになってきた。これに伴い、使い易い，コンパクトなリアルタイムOSへの要求も高まっている。今回68000を対象に、ビルディング・ブロック方式により各種システム構成に最適なOSが構築できるモジュラー型のリアルタイムOSを開発した。以下、本論文では、本OSのねらい、構成および結果について報告する。

2. 開発のねらい

本モジュラーOSの設計方針は以下の通りである。

(1) ビルディング・ブロック構造によるモジュール化設計：

リアルタイム・システムでの68000の利用要求は，(a)従来8ビット・マイクロコンが使われてきた分野においてより高性能なマイクロコンに転換する，(b)従来ミニコンが使われてきた分野において安価な高性能マイクロコンにリプレースする，などがあり、これらによりシステム形態も機器組み込み形のシステムからプログラミング機能を必要とするシステムまで多岐にわたる。そしてそれらのOSに対する性能，機能および容量などへの要求も多様となっている。そこで本モジュラーOSではOSをモジュール構造として、モジュールの選択により各システムに最適なOSを構築可能とするビルディング・ブロック方式とする。

(2) モジュールの選択はオブジェクト・モジュールのレベルで行なえるようにする。

(3) ROM（Read Only Memory）化可能とする。

(4) ポータビリティの向上：

ユーザ実機への移植性を容易にする。このため、MMU（Memory Managout Unit）制御など，ユーザ実機に依存する部分は別モジュールとする。

(5) 保守性の向上：

保守性の向上とポータビリティの向上を目的に、ソース・プログラムは、マシン依存部を除き高水準言語Cで記述する。

(6) リアルタイム環境での汎用OSの共存：

マイクロコン用汎用OSの分野ではCP/M*，MS-DOS**，UNIX***といった有力なOSが存在し、これらのOSの下で利用できる流通ソフトウェアの数も増大している。リアルタイムOSにおいても、これらの流通ソフトが活用できると，プログラミング用等に利用できユーザの利点は大きい。そこで本モジュラーOSでは、マイクロコン仮想計算機方式を実現し、リアルタイム環境に汎用OSを共存させることとした。これによりリアルタイムOSでこれらの流通ソフトウェアの利用を可能とする。

---

*CP/Mはディジタルリサーチ社の登録商標。**MS-DOSはマイクロソフト社の登録商標。***UNIXはBell研のOSの名称。

3. モジュール化設計

3.1 2階層モジュール化方式

モジュール化設計を行なう上では、

(1) 機能の着脱が容易であること。

(2) モジュール化に伴うオーバーヘッドの増加が少ないこと。

が必要である。さらにマイクロコンではOSの機能をROM(Read Only Memory)に入れ、ROMチップ単位に機能選択ができることも要求される。

OSのプログラム構造は、大きく2つの構成法，すなわち1)〝Message－passing System〟と2)〝Procedure－calling System〟とに分類することができる。[3] Message－passing Systemでは、プログラム間の通信や同期はメッセージやイベントをSENDやRECEIVEといったシステム・マクロ(OSコールまたはシステム・コール)を使って行なわれる。一方，Procedure－calling Systemでは、プログラム間の通信や同期は、通常のProcedure Call(手続き呼び出し)機構によって高速に行なわれる。Message－passing Systemは、プログラム間には共通データがなく、メッセージを介するという粗な結合となるため，ビルディング・ブロック構成をとるシステムの設計に適している。しかしながら通常Message－passing Systemは、Procedure－calling Systemに比較して、そのオーバヘッドは大きい。[4]

そこで、本OSのモジュール化設計では、両者の利点を活し、機能の着脱性と高速応答性を確保するため、Massage－passingおよびProcedure－Callingの両方式を組合せた「2階層モジュール化方式」を開発した。本方式を図1に示す。

この方式ではモジュールは、以下の2種類のモジュール，すなわち

(1) ブロック・モジュール

(2) サブ・モジュール

からなる。ここで，システムの機能選択は、ブロック・モジュールの選択により行ない、各ブロック・モジュールの詳細決定はサブ・モジュールの選択により行なう。

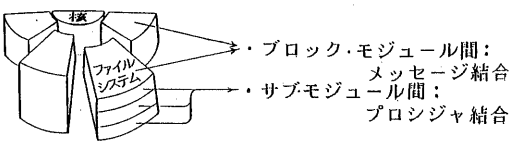


図1 2階層モジュール化方式

(1) ブロック・モジュール

図1において示した各ブロックが、本方式でのブロック・モジュールである。ここでリアルタイム核も1つのブロック・モジュールである。リアルタイム核を除く他のすべてのブロック・モジュールは，リアルタイム核によって制御されるシステム・タスクとして実現している。これらは，さきに述べたMessage－passing Systemを構成している。ブロック・モジュール間はメッセージを介して通信するという粗な結合となるためブロック・モジュールの着脱は容易となる。ブロック・モジュールの分割は、1つのブロック・モジュールがOSを構成する上での基本となるサブ機能を果すようにして行なった。

(2) サブ・モジュール

各ブロック・モジュールは、サブ・モジュールと呼ぶモジュールで構成する。1つのサブ・モジュールはいくつかのサブルーチンからなり、サブ・モジュール間は、Procedure Call構造で通信する。

3.2 システム生成方式

２階層モジュール化方式に基づくモジュール選択およびシステム生成方式を図２に示す。図２に示したようにシステムの大きな機能構成は、必要とするブロック・モジュールの選択によって決まる。最小のシステム構成はリアルタイム核のみからなり、他のブロック・モジュールは選択されない。論理ファイル・システムが選択された場合には、ファイルとして使用するI/O装置用物理ファイルシステム（：ブロック・モジュール）の選択も必要となる。リアルタイム核を除くブロック・モジュールは、タスクとして構成するので、選択されたブロック・モジュールの結合にはリンケージ・エディタは必要なく、単にエディション・テーブルと呼ぶハードウェアおよびソフトウェアの環境条件を設定するテーブル（ユーザ作成）に選択されたモジュールを登録するのみでよい。この方式はＲＯＭベースのシステムの構成にとくに有効である。たとえば、あるブロック・モジュールに変更・修正が行なわれても、システム全体を作りかえることなく、対応するＲＯＭチップのみを交換することにより対処できる。

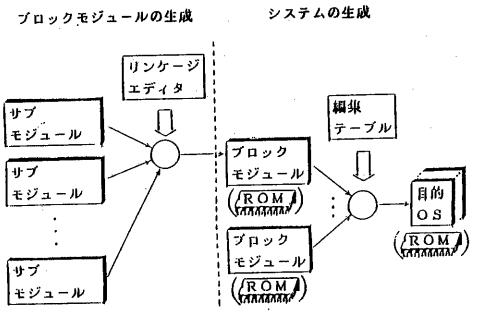


図２ システム生成方式

一方、各サブ・モジュールの間はProcedure Callで結合されるので、選択されたサブ・モジュールの結合はリンケージ・エディタを用いて行なわなければならない。

3.3 ＯＳの構造と利用形態

上記モジュール化方式に基づく本モジュラーＯＳの利用形態と各種システムへの適用を図３に示す。本ＯＳは大きく(1)リアルタイム・カーネル，(2)ファイル・システム，(3)セカンダリＯＳ　の３つのブロック・モジュールからなる。具体的には、ファイル・システムは１つのＦＭＴ（File Management Task）と、各デバイスに対応するＦＤＴ（File Driver Task）から構成される。セカンダリＯＳは、仮想計算機上にリアルタイム環境と共存して実行される汎用ＯＳのことである。

基本となる利用形態はブロック・モジュールの選択に応じて、

(1) リアルタイム核モード：
リアルタイム核からのみなる基本システム。

(2) リアルタイム核＋ファイルモード：　リアルタイム核とファイル・システムからなり、情報処理を含むリアルタイム制御ができるシステム。

(3) ２次ＯＳモード：　リアルタイム環境と共存して汎用ＯＳの利用が可能なシステム。

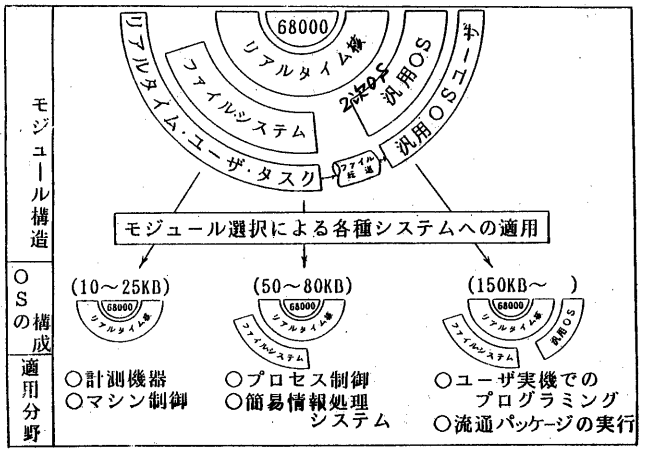


図３ モジュール構造と各種システムへの適用

の3つである。

4. 各ブロックの機能概要

(1) リアルタイムカーネル： リアルタイム・カーネルは本OSの核部である。リアルタイム・カーネルの概要を表1に示す。マシン・インタフェース・モジュール，各I/Oドライバ，各システムコール（OSマクロ）がそれぞれカーネルにおけるモジュール選択単位となる。

表1 リアルタイム・カーネルの概要

Items	Contents
Interruption	Process, I/O, Timer, Abnormal
Task Management	1)Multitasking 2)Intertask synchronizing 3)Asynchronous message communications between tasks using pseudo interrupt upon receiving message
Resource Management	1)Resource Allocation/Release 2)Monitoring of resource usage
I/O Management	1)Standard I/O : Console, Printer 2)User-coded I/O routine for the extended I/O 3)Monitoring of the status of I/O
Timer Control	1)Task invocation by delay and periodic modes 2)Specified time-of-day mode
Debugging Facilities	1)Display and change of memory contents 2)Start/stop 3)Display of table contents 4)Break point services 5)Events trace

(2) ファイル・システム： 以下のファイル編成をサポートしている。a)デバイスファイル，b)名前付ファイル；連続ファイル，シーケンシャル・ファイル，索引付シーケンシャル・ファイル，c)RAMファイル。ファイル・システムにおいては，各ファイル編成に対応するルーチンがモジュール選択単位となる。

(3) セカンダリOS： セカンダリOSは，一種の仮想計算機システムとして構成されており，現在CP/M－68Kを搭載している。セカンダリOSの利用法は，以下が可能である。

(a) リアルタイム処理と並行して，セカンダリOS下でプログラム開発やデバッグを行なう。(b) オンライン系ファイルを読み，セカンダリOS下でのユーティリティを使って処理，解析する。(c) セカンダリOSでのアプリケーションを実行する。

5. マイクロコン仮想計算機方式による汎用OSの共存方式

5.1 実現方式についての検討

2章 (6) で述べた目的を実現する方式，すなわち，汎用OSの下での豊富な流通ソフトウェアをリアルタイム・システムで利用できるようにするための方式として，大きく以下の2つの方式が考えられる。 (1) 汎用OSをリアルタイムOSの下で実行させる方式（2次OS方式），(2) 汎用OSのプログラム・インタフェース（OSコールまたはシステム・コール）をエミュレータと呼ぶ変換プログラムで模擬することにより汎用OSのアプリケーション・プログラムが実行できる環境を作りだす方式（エミュレータ方式またはバーチャル・オペレーティングシステム[6]）が考えられる。

2つの方式はそれぞれ利点，欠点を持っているが，本モジュラーOSでは，汎用OSをそのまま利用できること，リアルタイム・カーネルが大きくならないことなど比較的小規模システムに適していることから2次OS方式を採った。

5.2 2次OSの構造

2次OSの実現に当っては，以下の方針に基づいて設計を行なった。

① 本モジュラーOSはリアルタイムOSであるので，2次OSの実行により，リアルタイム処理の応答性に悪影響を与えないこと。

② 汎用ＯＳの核部は，通常バイナリで提供されるので変更を加えることは一般には不可能である。したがって汎用ＯＳを２次ＯＳとして搭載する上で汎用ＯＳの核部には変更を加えなくてもよい方式とすること。

③ モジュール化を推進するとの方針から２次ＯＳの着脱が容易であること。

この方針に基づき，２次ＯＳの実現方式としては，２次ＯＳをリアルタイムＯＳの下での１つのタスク（：１つのブロック・モジュール）として実現する。この方式は仮想計算機方式の分類ではタイプⅡバーチャルマシンに分類される方式である。[1),2)] ６８０００では Memory Mapped I/O 方式をとっているので入出力制御命令は特権命令でなく，常のメモリ参照命令（Move命令）である。したがってI/O命令を仮想計算機を制御するＶＭＭ（Virtual Machine Manitor）においてキャッチしそのシミュレーションを行なうことは困難である。

一般に，ＣＰ/Ｍなどマイクロコン用の汎用ＯＳは，ユーザ・マシンへの移植性をとくに重視して設計されている。このようなシステムでは，周辺I/Oとのインタフェース部は独立したモジュールとして作成される。そしてユーザにそのインタフェースが公開されているのが普通であり，ユーザで自由に変更できる。この部分はＣＰ/ＭやＭＳ－ＤＯＳではＢＩＯＳ（Basic Input Output System）と呼ばれている。ＶＭＭで入出力制御命令をシミュレーションするという方法をとらず，ＢＩＯＳ部を変更する方式をとった。すなわち，ＢＩＯＳ部で直接機械語のI/O制御命令を発行するのではなくリアルタイム・カーネルまたはファイル・システムに対し入出力システム・コールを発行するようＢＩＯＳ部を変更する方式とした。これによりＶＭＭの構成は，I/O制御命令のシミュレーションがないため，簡単な構造となった。この簡略化タイプⅡバーチャルマシンに基づいた２次ＯＳの構成を図４に示す。[5)] ２次ＯＳタスクは①ＶＭＭ，②ＢＩＯＳ，③２次ＯＳ核から構成され，これらは，リンケージ・エディタにより結合され１つのタスクとなる。なおＶＭＭの規模はＣ記述部１Ｋstep・アッセンブラ記述部１.２Ｋstepである。なお，本ＯＳでは，ＣＰ/Ｍ－６８Ｋを標準の２次ＯＳとしている。

5.3 ２次ＯＳの制御方式

(1) ２次ＯＳの優先度

２次ＯＳの実行によりリアルタイム・タスクに悪影響を与えないようにするため，２次ＯＳタスクは，リアルタイム・タスクに対して優先度の最も低いタスクとして実行させることを原則とする。

(2) ２次ＯＳのI/O装置の割当て

２次ＯＳ用の二次メモリには，ファイル・システムが管理するファイルを割当てることにした。ファイル・システムを介在させる大きな理由は，ファイル・システムのプロテクション機構を利用することにより，リアルタイム・プログラムが使っているファイル領域が，２次ＯＳ下のユーザ・プログラムにより誤ってアクセスされ破壊され

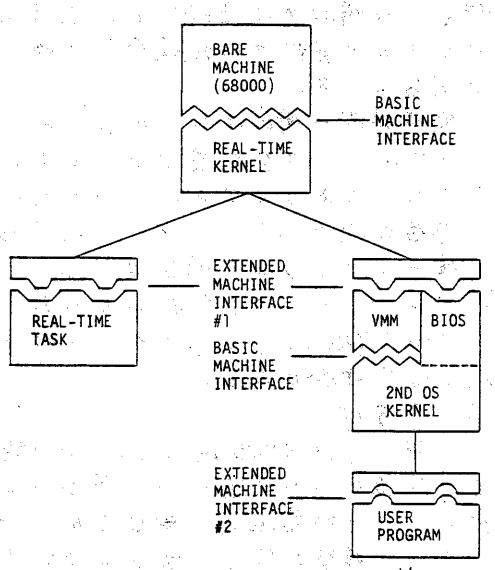


図４ ２次ＯＳの構造：簡略化タイプⅡバーチャルマシン

るのを防ぐためである。I/O割当ての定義は、エディション・テーブルで行なう。

(3) 特権命令のシミュレーション

タスク内で特権命令を実行すると、特権違反のマシン割込みが発生するが、この割込みは、"例外処理宣言システム・コール"によって例外処理ルーチンを登録しておくことにより、特権違反をおこしたタスクでキャッチできる。この例外処理ルーチンであるTMMは、仮想上の実行モード・レジスタをもち、仮想上の実行モードがシステム・モードならば、特権違反を起した特権命令のシミュレーションを行なう。

(4) TRAP命令のキャッチ

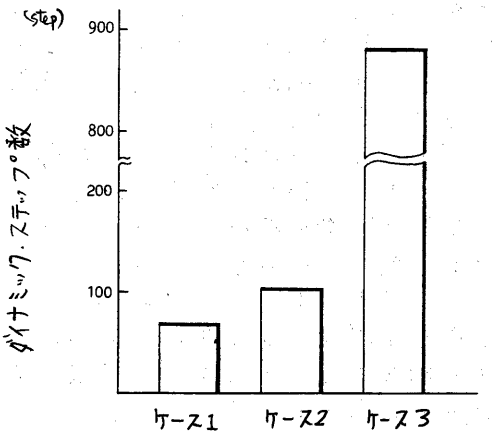
システム・コールの発行に利用されるTRAP命令は、特権違反のキャッチと同様の方式で"TRAP処理宣言システム・コール"によって行なう。

5.4 性能に対する考察

代表的な特権命令である"RTE"(return from exception)のシミュレーション・オーバヘッドを図5に示す。ケース3(status registerにトレースビットがセットされる場合)では、通常の場合の10倍の約900ステップもの時間を要している。しかしながら本方式ではこのシミュレーションは、2次OSタスクの実行として処理されるので、リアルタイム・タスクの応答に対して悪影響を与えることはない。

CP/M-68Kを2次OSとして実行させた場合の処理時間を、直接68000の上で実行させた場合と比較した例を図6に示す。

処理時間の増加は約30%である(リアルタイムタスクの負荷を0としている)。ただしこれはCコンパイラを実行させた場合の例であり、コンソールI/Oを多く使うプログラムの場合は、CP/M-68Kが1キャラクタ単位の入出力を行なっている関係上、オーバヘッドはこの数倍になる場合がある。



ケース1: 実行モードに変更がない場合
ケース2: 実行モードに変更がある場合
ケース3: 実行モードに変更がありさらにトレースモードがセットされる場合

図5 RTE命令のシミュレーションオーバヘッド

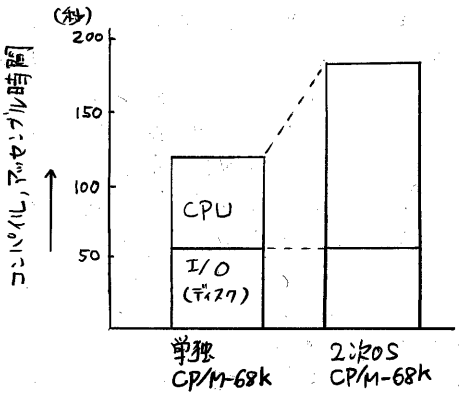


図6 2次OS(CP/M-68k)の性能
例: 2万ステップのCプログラムのコンパイル,アッセンブル時間

6. むすび

本稿では、68000用モジュラーOSの特徴および構造について述べた。本OSはROMベースの核巻組込み用OSからプログラミング機能を持つシステムまでをビルディングブロック方式で実現できる。とくに2次OSの搭載により、リアルタイ

ム環境と共存して汎用OSの下での流通プログラムを実行させることができ、リアルタイム・タスクの作成やパソコンとしての利用が可能となる。

7. 参考文献

1) J.P. Buzen, et al., "The Evolution of Virtual Machine Architecture," NCC, 1973, PP. 291-299

2) 大町, "仮想計算機システムの概要と技術課題," 情報処理学会, 計算機システムの制御と評価研究会資料 18, 1983, 2, 4, PP. 1~7.

3) H.C. Lauer, et al., "On the Duality of Operating System," Proc. 2nd International Symp. on Operating System," No. 2, Apr. 1979, PP. 3-19.

4) C. Crowley, "The Design and Implementation of a New UNIX Kernel," NCC, 1981, PP. 256-271.

5) S. Tsutsui, et al., "Universally Interfaceable Modular Operating System for 68000 Microcomputers," COMPCON FALL '83, Proc. PP. 453-460.

6) D.E. Hall, et al., "A Virtual Operating System," Comm. of the ACM, Vol. 23, No. 9, PP. 495-501.