オペレーティング・システム　39－4
（１９８８．６．１７）

多重OS「江戸」とそのウィンドウ・システムの開発

岡野裕之，堀素史，中川正樹，高橋延匡
東京農工大学　工学部

我々は，日本語情報処理に適した環境を目指し，フル２バイトコードを採用したOS／ｏ（omicron）を研究開発中である．多重OS「江戸」は，このOS／ｏの実行環境を提供する仮想マシンとして設計･開発されたソフトウェアである．ここで言う仮想マシンとは，複数のOSを同時に実行する機能（多重OS機能）と，抽象化したハードウェア･インターフェースをOSに提供する機能を指している．「江戸」の多重OS機能は，複数のOS／ｏ，あるいはOS／ｏと他のOSを同時に実行することを可能にするもので，これによってOS／ｏのデバッグや，他のOSからOS／ｏへの開発環境の移行を容易にすることができた．また，「江戸」によるハードウェアの仮想化によって，OS／ｏのマシン独立性を向上できた．特に，「江戸」はそのウィンドウ･システムによって，ビットマップ･ディスプレイとマウスを抽象化しており，OS／ｏでマンマシン･インターフェースの研究を行うことを容易にした．本論文では，背景となったニーズや，その内部構造について述べる．

Development of Hyper OS “edo” and Its Window System

Hiroyuki OKANO, Motofumi HORI, Masaki NAKAGAWA, Nobumasa TAKAHASHI
Faculty of Technology, Tokyo University of Agriculture and Technology, Koganei-shi, 184 Japan

A Hyper OS “edo” has been designed and developed as a virtual machine monitor primarily for OS／omicron, an operating system constructed to provide an unrestricted and coherent environment for Japanese information processing based on the full double-byte JIS code set. Edo offers multiple and abstract hardware interfaces for a plural set of OS／omicrons and other OS's. The multiple OS environment has made the CP／M−68K's utilities available for the development of OS／*o* and the shift from CP／M to OS／*o* smooth. The virtualized hardware by edo allows OS／*o* to be even independent of rather new devices like bit-map-display and mouse. The edo's window system abstracts them, so that the development of user interface of OS／*o* is made simpler and easier.

1．はじめに

我々の研究室では，フル2バイトコード化された，マルチタスクのオペレーティング・システム（OS），OS／o（omicron）を研究開発している［1,2,3］．OS／oのフル2バイト化は，ファイル名やディレクトリ名がフル2バイトであるだけにとどまらず，JIS C6226コードを用いることを前提とした2バイト単位の入出力や，文字コードと属性情報を分けて格納するための属性ファイルのしくみなど，「日本語OS」を強く意識したものになっている．OS／oのこの他の特徴としては次のものが挙げられる．

(1) 記述言語として言語Cを採用しており，そのコンパイラには，2バイト化され，リロケータブル，リエントラントなオブジェクトを生成する，Cat (C compiler developed at Tokyo University of Agriculture and Technology) を使用している［4,5,6］．

(2) 実記憶方式を採用している．

(3) アプリケーション指向のOSである．

(4) ターゲットプロセッサとして，広いアドレス空間を持つM68000シリーズを採用している．

アプリケーション指向というのは，いわゆる汎用OSが，実はどのアプリケーションにとっても不都合なOSであるという認識からとられた方針であり，OSの設計段階から特定のアプリケーションに的を絞るというものである．OS／oの場合，やはり当研究室で研究を行っている，日本語文書処理システム「浄書」(JOSHO) ［7］，オンライン日本語手書き文字認識JOLISなどを対象としている．

OS／oが対象とするアプリケーションとして，もう一つ，日本語入力エディタやCAIなど，マンマシン・インターフェースに関わる分野がある．OS／oは今まで，MC68000のSBC（日立製作所のシングルボード・コンピュータ）上で開発されてきたが，このハードウェアはマンマシン・インターフェースの研究をする場合には機能不足なものである（図1）．このため，OS／oを，ワークステーションのような，高解像度のビットマップ・ディスプレイと高いCPUパワーを持ったマシンに載せることが望まれた．また，市販のワークステーション上にOS／oを実現し，研究室外へ公開したいという要望もあった．OS／oをワークステーション上に実現しようとする際，次のような問題点があった．

(1) ビットマップ・ディスプレイやマウスのインターフェースを，どのような形でユーザ（アプリケーション）に提供するか．

(2) ワークステーションには普通，SBCなどにあったROMデバッガは存在しない．どのようにOSのデバッグをするのか．

上記 (1) の問題点を考える場合，まず言えることは，ユーザが直接ハードウェアを操作すべきではないということである．つまり，資源管理をするというOSの立場もさることながら，このようにしたのでは，アプリケーションの可搬性が著しく低下するということである．この問題は，既存のパーソナル・コンピュータ用のOSでは解決されておらず，何々OS用と銘打っていながらも，特定のマシンでしか動作しないソフトウェア・パッケージが氾濫している．これを解決するために我々がとった方法は，OS／oとハードウェアの間の層に，Hyper OSを置き，Hyper OSがハードウェアを仮想化したのち，さらにOS／oがその資源を管理するという二重構造である．これは，ハードウェアに直接関係する部分とその他の部分を切り分けることにより，保守性が向上するという利点を生む．そして，Hyper OSにデバッグ機能を付け，上記 (2) の問題を解決するという方向へも発展できる．

我々が今回開発した多重OS「江戸」は，このHyper OSに当たる．多重OS「江戸」は上記 (2) の問題を解決するために，デバッガを組み込むのではなく，複数のOSを同時に実行することにより，旧OSから新OSをデバッグするという方針を採用した．この，多重OSという名前は，複数のOSを同時に実行するという機能を指している．多重OS機能によって得られる利点として，この他に，旧OSから新OSへの開発環境の移行が容易になる，ということが挙げられる．我々はSBCでOS／oを開発する際に，OSとしてCP／M－68Kを使用してきたが，この開発環境をOS／oに移行する必要に迫られている．この時，「江戸」の多重OS環境は非常に有用である．

「江戸」の設計は，OS／oを動作させることを前提として行った．ただし，他のOSの動作も可能としている．現在，多重OS「江戸」初版はワークステーション2050（日立製作所）上に実現され，さらに第二版がワークステーション2050／32（同）に実現されている（表1）．すでに，CP／M－68K，OS／oが多重OS「江戸」上に移植され，その際，旧OSからのデバッグ機能の有用性が証明された．また，2050版の「江戸」上から，2050／32版の「江戸」上へのOS／oの移行も不自由なく行え，「江戸」によるハードウェアの仮想化の有用性が確かめられた．以下の章では，多重OS「江戸」の構造や，その評価などについて述べる．

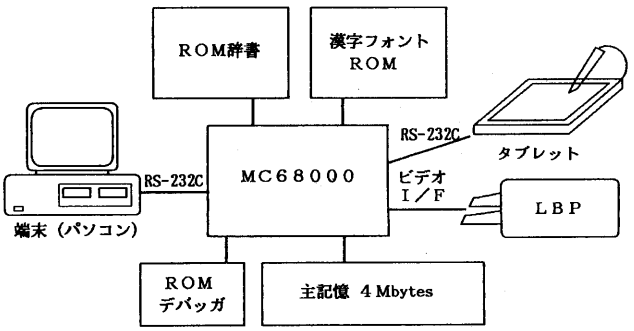


図 1　　SBCシステムの構成図

printed by 浄書 ［1］

表 1　多重ＯＳ「江戸」第二版のターゲットマシンの仕様

プロセッサ	MC68020 (20 MHz)
コプロセッサ	MC68881 (20 MHz)， ＵＲＲプロセッサ
主記憶	8 Mbytes (最大 16 Mbytes)
内蔵ハードディスク	88 Mbytes
フロッピーディスク	3.5 インチ (HD) 1台
表示能力	1120 × 780 (16 色)

2．多重OS「江戸」の構成

多重OS「江戸」は，多重OS機能（複数のOSを同時に実行する機能）を実現するカーネルと，ハードディスクやRS−232C，ビットマップ・ディスプレイなどのデバイスを制御し，その機能をOSに提供するデバイスドライバによって構成されている（図2）．以下，カーネルの動作原理，メモリ管理などについて述べたあと，デバイスドライバの概要を述べる．

2.1 多重OS「江戸」の動作原理

「江戸」がターゲットとするプロセッサは，OS／９と同じく，M68000シリーズである［8］．（ただし，MC68010より上位のもの．）M68000は，ユーザ状態とスーパバイザ状態という２つの特権状態を持っており，いくつかの特権化された命令（特権命令）は，ユーザ状態では実行不可となっている．特権命令をユーザ状態で実行しようとすると，特権命令違反例外が起こり，スーパバイザに制御が移るようになっている．「江戸」はこの機能を利用して，多重OS機能を実現する．つまり，OSをユーザ状態で走らせ，OSの実行する特権命令をエミュレートすることによって，CPU資源を各OSに分配する（図3）［10］．

OSに対して行われる例外処理は，すべて「江戸」がエミュレートする．「江戸」における例外処理の流れを図4に示す．図中のCOSCB，OSCBは，「江戸」の持つOS管理表であり，表２のような構造を持つものである．図４に示すように，COSCBはスーパバイザ・スタック中の一定位置に書き込まれており，そこからのポインタによって，その時点でアクティブとなっているOSのOSCBが示される．これらのOS管理表の構造は，「江戸」をマルチプロセッサで実現することを考慮して決定した．マルチプロセッサで実現する場合，OSCBはグローバルな管理表として一ケ所にまとめ，各プロセッサの持つCOSCBがこれをポイントし，参照するようにする．

例外が発生すると，図中左側にある実際のベクタ・テーブルがプロセッサによって参照され，「江戸」内にある各例外処理ルーチンへ制御が移る．「江戸」がOSに対して例外を発生させる場合は，各OSの持つベクタ・テーブルを「江戸」が参照し，例外をエミュレートする．このとき，例外を起こすべきOSが，その時点でアクティブでない場合，COSCB

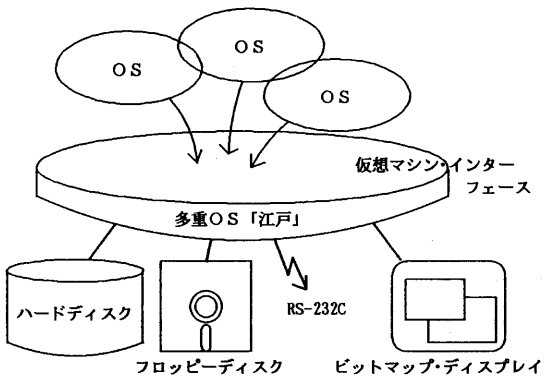


図 2　　多重ＯＳ「江戸」の構成

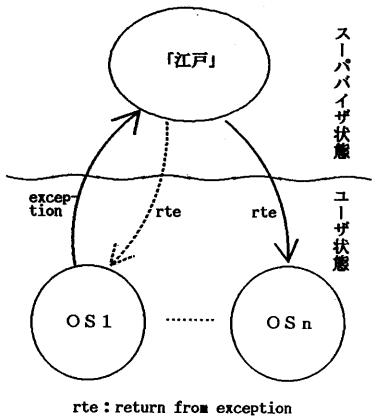


図 3　多重ＯＳ機能の概念図

を書き換え，アクティブOSを変更（OSスイッチ）してから，例外をエミュレートする．このように，「江戸」ではOSへの例外をすべてエミュレートによって行うため，割り込みデバイスの仮想化が可能という利点が生まれている．（これについては，あらためて述べる．）反面，すべてのtrapをエミュレートしなければならない，という欠点がある．

2.2「江戸」の提供するCPU環境

多重OS「江戸」初版はMC68010上に，第二版はMC68020上に実現されているが，「江戸」上で動作するOSに提供されるCPU環境は，これらターゲット・プロセッサと同一ではない．ここで言うCPU環境とは，次のようなことを指している．

- ○　レジスタ・セット
- ○　使用可能な特権命令
- ○　例外スタック・フレーム（例外発生時に積まれるス

タック）の内容

○ 「江戸」へのSVC用の命令

レジスタ・セットは，ユーザ状態でアクセスできるものに関してはすべて同一であるが，特権命令によってアクセスする一部のレジスタが省略されている．これは一部の特権命令（movec，moves 命令）のエミュレートを行っていないからで，これによって処理を容易にしている．また，第二版に関してはコプロセッサに関する特権命令もサポートしていないが，これは今後実現する予定である．

例外スタック・フレームが実際のプロセッサと異なるのは，「江戸」の大きな特徴と言える．具体的には，MC68000 が生成する例外スタック・フレームをそのままエミュレートしている．M68000 シリーズのスタック・フレームは，MC68000 と MC68010 以降のものとで仕様が変更されており，普通，MC68000 上で動作するOSをMC68010 以降のプロセッサに持ってきてもうまく動作しない．OS／ｏの開発機，および最初のターゲット・マシンが，MC68000 の SBC であったために，我々はOS／ｏを含めてMC68000 用のソフトウェアを多く所有している．そのため「江戸」は，これらのソフトウェアを生かすべく，プロセッサに依らず例外スタック・フレームをMC68000 仕様としている．

「江戸」では，ビット・パターン $ffff (dc −1) を，「江戸」へのSVC用の命令として使用している．この命令は，M68000 の命令セットの内，コプロセッサ用の命令として使用されているが，ユーザ定義のコプロセッサ用として用意されているものであり，「江戸」が使用しても差し支えないものである [9]．「江戸」へのSVCは，機能番号と引数を特定のアドレス・レジスタによって指したのち，dc −1 で設定した $ffff を実行することによって行われる．このときに使用されるアドレス・レジスタは，OSの起動時に指定できる．SVCは普通，次のようにして行われる．

```
*       SVCの例（A7を使う場合）
        move        引数３，－（A7）
        move．l     引数２，－（A7）
        move        引数１，－（A7）
        move        機能番号，－（A7）
        dc  −1
```

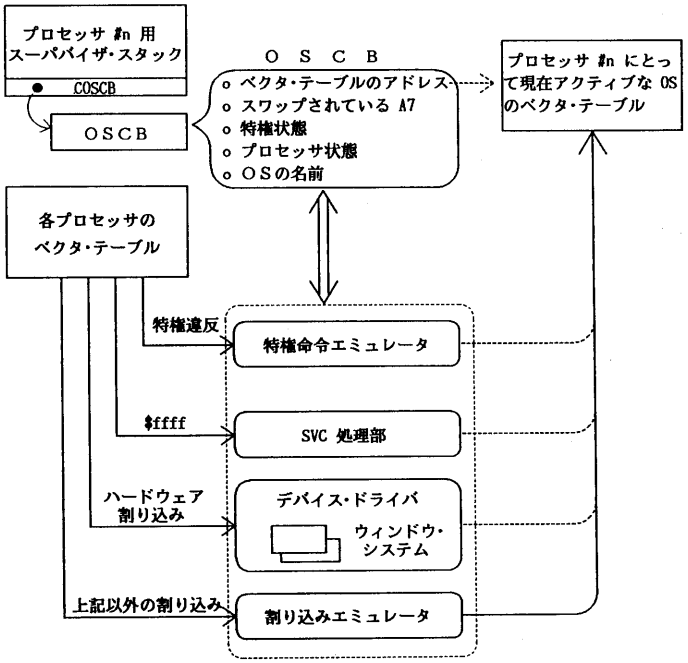


図 4　　多重OS「江戸」の構造

表 2　　COSCB と OSCB の内容

COSCB（Current OSCB）

位置（バイト）	名前	サイズ	内容
0	processor_ID	word	プロセッサ ＩＤ
2	active_OSnum	word	現在アクティブな OS のＯＳ番号
4	active_OSCB	long	現在アクティブな OS の OSCB のアドレス
8	top_of_OSCB	long	OSCB の先頭アドレス
１２	max_OS	word	起動可能な OS の最大数
３２	————	long×8	初期コンテキスト（A0-A7）

OSCB（Operating System Control Block）

位置（バイト）	名前	サイズ	内容
0	OS_name	word×17	OS の名前（JIS C6226 で１６文字＋null）
３４	OS_number	word	OS が起動される順に、0,1,2… と振られる番号
３６	processor_state	word	プロセッサ状態
３８	system_byte	word	SR の上位バイト
４０	swapped_USP	long	スワップされている USP
４４	swapped_SSP	long	スワップされている SSP
５２	vector_table	long	ベクタ・テーブルのアドレス
５６	SVC_pointer	word	SVC に用いるアドレス・レジスタ
６０	active_COSCB	long	COSCB へのポインタ
６４	————	long×16	非実行中の D0-D7/A0-A7

printed by 浄書 [3]

```
move        (A7)+, 戻り値
addq        #8, A7
```

2.3 メモリ管理

現段階の「江戸」では，メモリ管理を全く行っていない．「江戸」の下で動作するOSは，あらかじめ，互いに競合しないようにアロケートされていなければならない．「江戸」がメモリ管理を行わない理由として，次のことが挙げられる．

(1) OS／ｏが実記憶方式を採用しているため，「江戸」も実記憶方式を用いる．

(2) OSどうしが互いにのぞき合ったり，メモリを通じた通信を行ったりしたい．

上記（1）の理由から，実記憶方式を採用しているのであるが，実記憶の枠組みの中でも，DATを用いた空間の組み替えの機能をOSに提供するような選択もあった．しかし，上記（2）のような理由から，単一空間にOSを配置することにした．ただし，この場合でも，安全上の理由から，ページごとのアクセス属性をOSに設定させたり，OSの領域ごとのアクセス属性を設定させたりすべきである．しかし，現在はハードウェア上の制約から実現していない．

2.4 デバイスドライバ

「江戸」には，ハードディスク，RS−232C，ビットマップ・ディスプレイ，マウスなどのデバイス・ドライバが組み込まれており，ユーザ（OS）はSVCによってこれらを利用できる．デバイスドライバをアクセスする時のSVCの仕様が，すなはち「江戸」の仮想マシン・インターフェースである．デバイスドライバの内，ビットマップ・ディスプレイとマウスを管理している部分は特に，ウィンドウ・システムと呼ばれ，大きな位置を占めている．ウィンドウ・システムの詳細は次章に回し，ここではそれ以外のデバイスドライバについて記述する．

ウィンドウ・システム以外のデバイスとしては，現在次のものが実現されている．

- ○ キーボード
- ○ ハードディスク
- ○ フロッピーディスク
- ○ カレンダ
- ○ RS−232C

これらのデバイスの，仮想マシン・インターフェースの決定は，なるべくハードウェア・イメージを残すという方針で決定された．これは，「江戸」によるハードウェアの抽象化を低レベルなものにし，OSによる抽象化の余地を多く残した方が，OSにとって自由度が高いだろうという考えに基づいている．以下，この特徴を列挙する．

(1) キーボード入力では，キーコードが入力できる．

(2) キー割り込みが受けられる．

(3) ハードディスク，フロッピーディスクは，処理の終了報告を割り込みによって行う．

(4) ハードディスクのOSごとのパーティションは「江戸」で管理せず，各OSはハードディスク全体を自由にアクセスできる．

(5) フロッピーディスクはユニットのロック，アンロックの機能を提供し，OSごとの排他制御を行う．

上記（1）で言うキーコードとは，ハードウェアが固有に持つものである．キーコードが入力できることによって，“変換キー”など，JISコードに割り当てられていないキーの入力ができるようになるほか．ノートオン，ノートオフで別のキーコードを用意することによって，キーの上げ下げの検出も可能となる．（キーの上げ下げに関しては，ハードウェア上の制約から，現在は実現されていない．）「江戸」が提供するキーコードはハードウェア固有のものであり，標準キーコードのような規格は用意しなかった．というのも，キーボードは多種多様であり，標準化が難しいからである．無理をして標準化を行い，ハードウェア（キーボード）の選択の幅を狭めるよりも，ソフトウェアでキーコード・テーブルを持つなどして対処する方が得策と考えた．

上記（2），（3）にあるように，「江戸」の仮想マシン・インターフェースは，ハードウェア・イメージに合わせて割り込みを用いている．これによって実際のハードウェアとのギャップが少なくなり，慣れやすいインターフェースとなっている．

3．ウィンドウ・システム

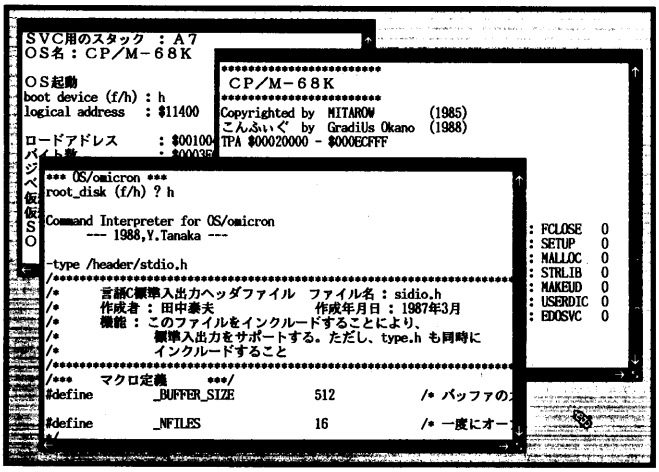
本章では，多重OS「江戸」が提供する仮想マシン・インターフェースで最も特徴的な，ビットマップ・ディスプレイとマウスのインターフェースを提供する，ウィンドウ・システムについて述べる．

3.1 ウィンドウ・システムの概要

「江戸」のウィンドウ・システムは，オーバラッピング方式を用いたマルチウィンドウを採用している（図5）．各ウィンドウには，OSが一つずつ持っている，仮想フレーム・メモリの一部が表示される（図6）．つまり，「江戸」のウィンドウ・システムは，各OSに一つずつウィンドウを開くもので，OSのタスクごとにウィンドウを開くものではない．これは，ハードウェアを仮想化する層と，フレーム・メモリを仮想化する層を分けるべきであるという立場による．つまり，「江戸」が提供する仮想フレーム・メモリは，あくまでも個々のハードウェアの差異を吸収しただけの低レベルなものにとどめ，そのインターフェースはビットマップを直接操作するという形式をとる．そして，仮想フレーム・メモリの上で，OSが別のウィンドウ・システムを走らせ，さらに高度な仮想化を加えるわけである．このときの仮想化は，ネットワーク・システムのことなどを考慮し，直接ビットマップを操作させない方法になるであろう．OS／ｏ上のウィンドウ・システムは現在開発中である．

「江戸」のウィンドウ・システムの機能は，ウィンドウの

printed by 浄書 [4]



左図は，「江戸」の画面のハードコピーである．手前のウィンドウがＯＳ／ｏで，真中がＣＰ／Ｍ-６８Ｋ奥が「江戸」のウィンドウである．

「江戸」のウィンドウに対して，ＯＳの起動などのコマンドを入力できる．

図 5　　オーバラッピング方式のマルチウィンドウ

表示の他に次のものがある．

(1) キーボード入力をどのOSに向けるか決定する．

(2) ウィンドウ上で行われるマウスのクリック，マウスの移動をOSに伝える．

上記 (1) は，重なりの一番上になるウィンドウにキー入力を向けることにしている．また，(2) は，割り込みによってそのタイミングを伝えることによって実現している．マウスのクリックには，ウィンドウ上のそれの他に，ウィンドウの枠に書かれた矢印のクリックも含まれる．計4つある矢印へのクリックは，別々のイベントとして入力される．

3.2 マルチウィンドウの実現

「江戸」のウィンドウ・システムの実現に際しては，専用LSIのような特別なハードウェアは利用しなかった．これは，ターゲット・マシンのハードウェア上の制約にもよるが，すべてソフトウェアで記述することによる移植性の向上も意図している．この結果，ウィンドウの描画は，仮想フレーム・メモリから実フレーム・メモリへの，ソフトウェアによるイメージ転送の形で行われる．この描画動作は，二段構えのメモリ転送となり，ソフトウェアで行う場合かなり負担の重いものである．そこで，メモリ転送の量を最小に抑えるようなアルゴリズムをとった．

ウィンドウの管理は，ウィンドウごとにマスタテーブルという管理表を持つことによって行われる（図7）．この管理表には以下のような情報が格納される．

- ○ OS番号
- ○ OSの名前
- ○ 矩形領域管理テーブルへのポインタ
- ○ 仮想フレーム・メモリのアドレス，大きさ
- ○ 仮想フレーム・メモリ中の表示領域の位置
- ○ 表示画面におけるウィンドウの位置

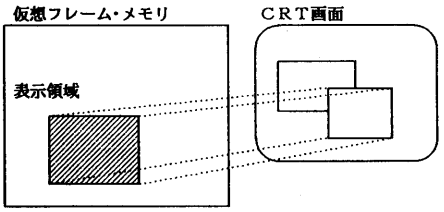


図 6　　仮想フレーム・メモリ上の表示領域

- ○ スクロールバーの管理テーブルへのポインタ
- ○ マウス・カーソルに関する情報

ウィンドウ間に重なりが生じると，重ねられた側のウィンドウは，複数の矩形に分割された形となる．一つのウィンドウについて，表示画面に現れている（重なりの上になっている）矩形領域の位置を保持しているのが，上記で示した矩形領域管理テーブルである．ウィンドウ・システムは，この矩形領域管理テーブルを参照することによって，メモリ転送の量を最小に抑えることを可能としている．たとえば，ウィンドウを拡大する時，元から表示してあった領域は書き換える必要がなく，広がった部分だけを書き換えればよい．このようにするため，拡大する前の矩形領域管理テーブルと，拡大後のそれを比較し，増えた矩形領域を把握している．

3.3 仮想フレーム・メモリのインターフェース

仮想フレーム・メモリのインターフェースは，我々が研究開発している日本語フォーマッタ浄書との相性がよいように決定した．浄書はLBPを出力装置に持つバッチ型の日本語フォーマッタであるが，この出力を「江戸」を通じてCRT画面に表示できれば，WYSIWYGへの発展なども望めるこ

printed by 浄書 [5]

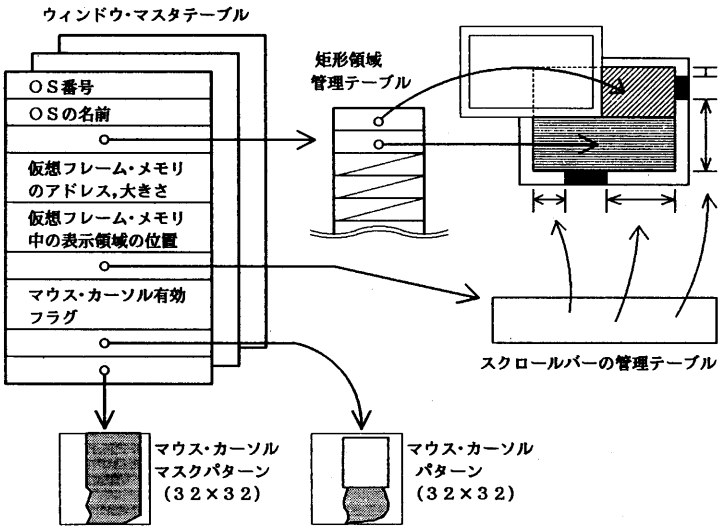


図 7　　ウィンドウ・マスタテーブルの構造

とになる．浄書のフレーム・メモリは白黒で4096×4096（2Mbytes）であるが，「江戸」の仮想フレーム・メモリはこれを満足できるよう，白黒で任意の大きさということにした．仮想フレーム・メモリのアドレスや大きさは，OSの起動時に指定する．

仮想フレーム・メモリに関するSVCには次のようなものがある．

- ○ 仮想フレーム・メモリのアドレス，大きさを得る
- ○ 仮想フレーム・メモリ上の表示領域の設定
- ○ 仮想フレーム・メモリ上の更新領域の設定

表示領域とは，仮想フレーム・メモリ上のウィンドウに対応する部分であるが（図6），この位置を移動することにより，仮想フレーム・メモリの任意の位置を画面に映し出すことができる．普通，マウスの矢印クリックや特定のキーが押下されたタイミングでこれを行う．マウスの矢印クリックによって，ウィンドウ・システム側が表示領域を移動（スクロール）してしまうのではなく，OSに割り込みをかけて矢印クリックの旨を伝えるだけにとどめている．これによって，矢印クリックというイベントを，OS側が適当に解釈でき，自由度が高くなる．上記の更新領域とは，仮想フレーム・メモリ上で書き換えを行った領域のことで，この領域をSVCで宣言してはじめて，仮想フレーム・メモリの内容がウィンドウ表示に反映される．

4．多重OS「江戸」の評価

本章では，実際に「江戸」上にCP／M－68K，OS／ｏを移植した結果得られた感想などを述べたあと，「江戸」の問題点について述べる．

4．1 多重OS「江戸」の効果

「江戸」上にCP／M－68K，その後OS／ｏを移植してみて，「江戸」の多重OS機能が非常に有用であることが実感できた．OS／ｏのデバッグには，CP／M－68K上のデバッガが利用でき，メモリダンプやディスアセンブルが行えた．さらに，「江戸」のHyper OSという性質から，OSに割り込みが起こったかどうか，特に多重例外（ソフトウェア例外と外部割り込みが同時に起こること）が起こったかどうかをモニタすることができ，これが非常に有用であった．また，メモリを介したCP／M－68K，OS／ｏ間の通信が暫定的に行われており，ファイル転送が可能となっている．これは，開発環境の移行を行う上で重要な機能である．

今後の多重OS機能の活用法として，OS間通信の実験をしてみることが挙げられる．つまり，OS間通信を一つのマシンだけで行うわけである．たとえば，TCP／IPをつかってOS間通信をしようとする場合，「江戸」にIPモジュールを組み込み，それがネットワークにつながっていると見せ掛けて，同一マシン上の他のOSと通信させることができる．この場合，

- ○ データグラムは失われるかも知れない
- ○ データグラムは分割されるかも知れない
- ○ データグラムは発信順で到着するとは限らない

などの，IPの性質を「江戸」がシミュレートすれば，OS上にあるTCPモジュールのコーディング，デバッグを効率的に行える［11］．

4．2 多重OS「江戸」の問題点

「江戸」の問題点として，次のようなことが挙げられる．

(1) trapをエミュレートによって行うため，オーバヘッドが大きい．

(2) 仮想フレーム・メモリが白黒だけを対象としている．

(3) 例外スタック・フレームがMC68000仕様である．

(4) アセンブリ言語のみで記述されており，保守性が悪い．

上記(1)の問題は，「江戸」が本質的に抱える欠点である．どれ程のオーバヘッドがあるかと言うと，言語Cのコンパイル時間で測定した場合，約一割のオーバヘッドがあった．(2)の問題は，仮想フレーム・メモリのインターフェースが簡潔であるという利点とうらはらである．また，(3)の問題は，MC68020用のOSを，「江戸」に移植する場合などに表面化するであろうが，現時点ではMC68000のソフトウェア

printed by 浄書 [6]

をそのまま継承できるという利点になっている．(4) については，いずれ言語Cによる再記述が必要と思われる．アセンブリ言語で記述した理由というのは，言語Cコンパイラ Cat の生成するオブジェクトが特殊なコンテキストを持っているため，割り込みを扱うプログラムを書きづらかったことや，効率のよいプログラムを書きたかったからである．コンテキストの問題は，Cat が自前のものであるという利点を生かし，「江戸」にあったものに作り変えようと考えている．

5．おわりに

多重OS「江戸」によって，我々は以下の利点を得た．

(1) 新しいデバイスへの対応

マウス，ビットマップ・ディスプレイなど，現在のワークステーションには当たり前のデバイスでありながら，OSレベルでのインターフェースが規格化されていなかったデバイスを規格化（仮想化）し，OS／οのハードウェア独立性を高めた．

(2) OSのデバッグ環境を提供

多重OS機能により，旧OS上のデバッガから新OSをデバッグするという環境が提供された．また，OS／οにOS間通信の機能を付加する場合には，同一マシン上にOS／οを2つ走らせ，実際にそれぞれが通信してデバッグすることが可能となった．

「江戸」のこれからの課題としては，マルチプル・プロセッサへの対応や，ローカル・エリア・ネットワークへの対応がある．

謝辞

本研究について御配慮頂いた（株）日立製作所神奈川工場篠崎雅継主任技師ほかの方々に深謝する．

参考文献

[1] 高橋延匡，並木美太郎，武山潤一郎，中川正樹：“OS／οのアーキテクチャと第一版の実現”，情報処理学会オペレーティングシステム研究会資料 24－11，1984．9

[2] 鈴木茂夫，中川正樹，高橋延匡：“OS／ο第二版におけるタスク管理”，情報処理学会第36回全国大会予稿，1988．3

[3] 田中泰夫，鈴木茂夫，中川正樹，高橋延匡：“OS／ο第2版のファイルシステム”，情報処理学会第36回全国大会予稿，1988．3

[4] 並木美太郎，屋代寛，田中泰夫，篠田佳博，藤森英明，中川正樹，高橋延匡：“OS／omicron用システム記述言語C処理系Catのソフトウェア工学的見地からの方式設計”，電子情報通信学会論文誌，Vol．J71－D No．4，pp652～660，1988．4

[5] 田中泰夫，中川正樹，高橋延匡：“OS／ο用言語Cコンパイラ catの日本語化の方式とその実現”，情報処理学会第34回全国大会予稿，1987．3

[6] 鈴木茂夫，小林伸行，田中泰夫，中川正樹，高橋延匡：“OS／omicronにおける日本語プログラミング環境”，情報処理学会コンピュータシステム・シンポジウム，1987．11

[7] 里山元章，中川正樹，高橋延匡：“OS／omicronにおける文書出力システム浄書（JOSHO）”，情報処理学会第33回全国大会予稿，1986．10

[8] M68000マイクロプロセッサ ユーザーズ・マニュアル，CQ出版社，Motorola Inc．，1984

[9] MC68020ユーザーズ・マニュアル，CQ出版社，Motorola Inc．，1986

[10] J．P．BUZEN，U．O．GAGLIADI：“The evolution of virtual machine architecture”，National Computer Conference，Vol．42，pp291～299，1973

[11] Postel，J．：“Internet Protocol”，RFC791，USC／Information Sciences Institute，1981

printed by 浄書 [7]