TECHNOLOGY REVIEW

UNISYS 技報 通巻 31

1991年11月発刊 Vol. 11 No. 3

# 特集：ワークステーション



日本ユニシス

ISSN 0914-9996

企業内情報システムにおいて，メインフレーム集中のシステムからネットワーク・コンピューティング・システムへの変革が進んでいる．佐藤博の**ネットワーク・コンピューティングの特質とそれに基づくシステム開発アプローチの変革**は，ネットワーク・コンピューティング・システムをベースにした情報システムの構造について，外形的側面・内面的側面から論じるとともに，この変革に応えるために企業内情報システム部門はどのような役割の変化を求められるかを述べている．

EXOS（EXcellent Office System）は，エンドユーザを対象にした統合オフィスシステムである．河合昭男の**EXOS統一操作環境の実現**は，EXOS開発の最大の目標であった“使いやすさと操作の統一性”の問題について，ワークステーションPW²上のアプリケーション開発でどのような手法で解決したかを述べている．

近年，ワークステーションがパーソナルコンピュータの性能向上を背景に，ユーザインタフェースにGUIを提供した環境が一般的になってきた．佐野浩之の**GUI環境とアプリケーション開発——統合運用システム（IOF）におけるGUI適用を背景として**は，GUIの概念をMS OS/2*のPMを基に解説し，さらにシリーズ2200•1100システムのIOFにおけるGUIの適用例・GUI適用時の考慮事項を紹介している．

* MS OS/2，PMは米マイクロソフト社の登録商標である．

田村太一の**PM環境下でのアプリケーション・プログラム開発事例**は，ワークステーションPW² EXOSのコンポーネントであるEXSD（スクリーン•デザイナ）の開発事例を通して，MS OS/2のPM環境下で稼働するイベント駆動型アプリケーション・プログラムを開発する上で考慮しなければならない事項について報告している．

ワークステーション上ではハードウェアの高機能化と高性能化に伴って，GUIを備えたソフトウェア・プロダクトが整備されつつある．佐々木茂の**グラフィック環境下のターミナルシステム**は，MS OS/2のPM上で走行するシリーズ2200・1100用のターミナル・システムであるインフォメーション・サービス・システムについて，その概要と設計のねらいについて記述している．

情報量の増大・業務の多様化に伴う複数コンピュータによる分散処理へのシステム体系の移行，高性能ワークステーションの普及・ネットワーク技術の進歩による分散処理システムの構築，の動きの中で，水平分散を実現していた統合データ交換システム（DDA）に対してワークステーションへの展開が求められた．北川達朗は，**垂直分散型ネットワークにおける統合データ交換環境の構築**の中で，構築時の経緯等を述べ，今後主流になると考えられるインフォメーション・ネットワークに必要とされる統合環境に言及している．

ワークステーションの機能拡充による費用対能力比は年々向上しているが，性能に比べ操作が簡便化されているとは言えない．千葉啓善は，**ワークステーションによる簡易入力支援機能**の中で，エンドユーザのデータ入力簡便化を目的として開発した，ローカルヘルプ・システムの概要を紹介している．

FINESSE-Jは，統合化された金融機関営業店システムの概念であるFSA（Financial Systems Architecture）のアプリケーションソフトウェアを具現化したものである．天野剛の**FINESSE-Jの特質と操作性——カスタマイズの視点から**は，FINESSE-Jの持つソフトウェアパッケージとしての操作性を，「カスタマイズ•モジュール」を取り上げて評価している．

近年，高性能EWSに対する期待が高まっている．吉田欣司・唐下勉の**S8400シリーズ・エンジニアリング・ワークステーションの開発**は，日本ユニシスが米国ユニシスと共同で開発したS 8400シリーズについて，ハードウェア面から見た高速・高性能を実現するための構造・工夫・拡張性等について述べている．

# 特集「ワークステーション」の発刊によせて

中　村　　脩

「ワークステーション」のコンピュータ産業に占める重要度は，近年とくに高まっている．データクエスト社の調査によれば国内市場は，1994年まで30～40%の成長を続け，1994年には6000億円に達すると見込まれている．

「ワークステーション」は，一般にハードウェアの商品カテゴリの名称になっているが，元来は「コンピュータと対話しながら作業を進める場(＝環境)」という利用形態に焦点を合わせた商品群である．たとえば，XEROX社のStarワークステーション，UNIX*ベースの高性能ワークステーション等がある．また，オフコンやメインフレームのいわゆる「端末」でも，エンドユーザが主体的にコンピュータと対話し，コンピュータの上の処理を組み立てて自らの作業を進める場合に，そのための種々の機能を付加するとき，それを「ワークステーション」と称する．それは，「中央(センタ)」に制御される「端末(ターミナル)」の位置づけではなく，「中央(サーバ)」にサービスを要求する自律的な「利用者(クライアント)」である．

また，業務用途の区分によっても，オフィス業務用(オフィス・ワークステーション)，エンジニアリング業務用(エンジニアリング・ワークステーション)，コンピュータ・ソフトウェア開発用(プログラマーズ・ワークベンチ等)，知的作業支援用(AIワークステーション)等に分かれる．そこに用いられるオペレーティング・システムもMS-DOS**，MS OS/2**，UNIXあるいはメーカ固有のもの等多様である．これは前述したように，本来的に「ワークステーション」は利用形態に着目した名称であるからである．

このように「ワークステーション」が包含するハードウェアとしての商品は多岐にわたるが，本特集では主としてオフィス・ワークステーションに焦点を合わせて構成するとともに，エンジニアリング・ワークステーション(EWS)に対する期待に応えるべく米国ユニシスと共同開発したS 8400シリーズ開発の紹介も行っている．

オフィス・ワークステーションには，従来のオンライン業務の遂行からワープロでの文書作成業務まで多種多様な利用形態が要求される．また，ユーザも管理職，専門職，事務職等，多岐にわたり，その習熟度，使用頻度等もさまざまである．さらに，オフィス業務という性格上，「通信」は不可欠である．

このような背景から，GUI(グラフィック・ユーザ・インタフェース)を中心にした統合操作環境とLAN(ローカル・エリア・ネットワーク)がワークステーションの重要な要素技術となってきた．これらは，本号の論文の柱の一つである．また，エンドユーザ指向のパッケージ・ソフ

---

* UNIXオペレーティング・システムはUNIX System Laboratories, Inc.が開発し，ライセンスしている．

** MS-DOS，MS OS/2は米マイクロソフト社の登録商標である．

トウェアも重要である．統合オフィスシステムの「EXOS」，金融営業店支援アプリケーション・パッケージ「FINESSE-J」等の論文を集録した．当社では，昨年10月に90年代を担う新世代のソフトウェア・アーキテクチャとしてUA(Unisys Architecture)を提唱し，オープンシステムを中心とした新しい商品提供の枠組みを発表した．パーソナル・コンピュータ/ワークステーションはその中でワークステーション・グループとして位置付られ，ホストコンピュータ(インフォメーションハブ)，サーバーコンピュータとの連携において，90年代の情報システムの中核をなすものと考えられている．

今後とも利用形態を直視し，その要求に応える商品をタイムリに提供することに一層の努力を続けていきたいと考える．

(オープンシステム推進本部 本部長)

# ネットワーク・コンピューティングの特質とそれに基づくシステム開発アプローチの変革

# The characteristic of Networked Computing and Changes in Approaches to the Development of Networked Computing Systems

佐　藤　　博

**要　約**　企業内情報システムにおいて，メインフレーム集中のシステムからネットワーク・コンピューティング・システムへの変革が進んでいる．それは，エンドユーザ自らが情報のつくり手になっていくという情報システムの開発主体の拡大をその本質に含んでいる．

開発主体の拡大は，システム開発アプローチの変革をその前提としている．そこで，従来は情報のつくり手であった企業の情報システム部門の役割として，エンドユーザが新しい開発アプローチに基づいて望む情報を，円滑に得ることのできる環境を整備することが求められる．

**Abstract**　When it comes to in-house information systems, transitions are obviously in progress from mainframe-oriented systems to networked computing ones. This essentially implies the extension of main developers of information systems; namely, the fact that end users themselves are destined to be developers of information systems. And the extension of major systems builders entails, as its precondition, changes in approaches to systems development efforts.

In this context, the corporate information systems departments who have traditionally been systems developers are now required to straighten up and enhance the environment that makes it possible for end users to have easy access to any information they desire according to new systems development approaches.

## 1. はじめに

ネットワーク・コンピューティング・システムは，LANで相互に結合した複数のコンピュータが操作者から見ると，あたかも一つのコンピュータ・システムであるかのように機能する複合システムである[1]~[4]．またダウンサイジングとは，従来メインフレームのみで行っていた処理（の一部）をマイクロプロダクト上での処理に置き換えることを指す．その場合一般には，一台のマイクロプロダクトで置き換えることは，物理的な容量・性能から無理であることから，ネットワーク・コンピューティング・システムの形態をとる．したがって，ダウンサイジングとネットワーク・コンピューティング化とは同じ意味に用いられることが多い．

しかし，強いて両者のニュアンスの違いを挙げるならば，ダウンサイジングは，システムコストの削減にその焦点が当たっているのに対して，ネットワーク・コンピューティング化は，後述するように経営管理の単位組織に情報処理機能を「ダウンサイジング」するという，情報処理の主体の拡大・移転に焦点を当てている．

本稿では，ネットワーク・コンピューティング・システムをベースにした情報システムの構造について，まず外形的な側面から論じ，次に内面的側面を「データ格納」

と「処理」に分けて論じる．最後に，この変化に応えるために企業内の情報システム部門はどのような役割の変化を求められるかを述べる．

## 2. ネットワーク・コンピューティング・システムの外形的特徴と優位性

ネットワーク・コンピューティング・システムの定義には必ずしも定まったものはないが，本章では一般に用いられている定義を解説し，次にネットワーク・コンピューティング・システムの出現の背景を企業内情報システムにおけるその優位性の側面から明らかにしていく．

### 2.1 ネットワーク・コンピューティング・システムの定義と特徴

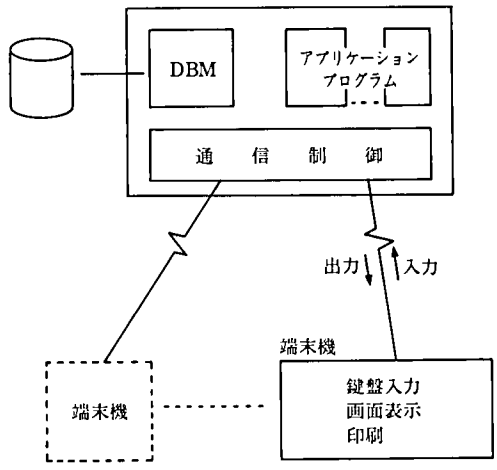
前述したようにネットワーク・コンピューティング・システムとは，LAN で相互に結合した複数のコンピュータが操作者から見ると，あたかも一つのコンピュータ・システムであるかのように機能する複合システムである．ここで「複数のコンピュータ」は「クライアント」と「サーバ」に機能上分類される．クライアントは操作者が（操作の）対象としているコンピュータで，それは通常，パーソナル・コンピュータ（PC），あるいはワークステーション（WS）である．サーバは，個々の PC あるいは WS（つまりクライアント）では実現できない機能，あるいはクライアント間で共有する機能について，LAN を経由して代替実行する．サーバは，スーパー PC あるいはハイエンド WS，ミニコンピュータ，メインフレーム等で，代替実行する機能に応じてさまざまなものが用いられる．一つのネットワーク・コンピューティング・システムに複数のサーバが用いられることもあるし，また物理的には一台の WS がクライアントとサーバを兼ねることもあり得る．

このように操作者はクライアントを通して，単体の PC/WS を越えて LAN のサーバをも自分の PC/WS の資源の延長として利用できる（single image）．サーバ上のこのような機能を PC/WS 側の操作者に提供可能にする基本ソフトウェアを「ネットワーク OS」と総称する．すなわち，ネットワーク OS はクライアントに対して，共用のデータファイル，プリンタ，通信網，プログラムコードおよび CPU 等を提供する．

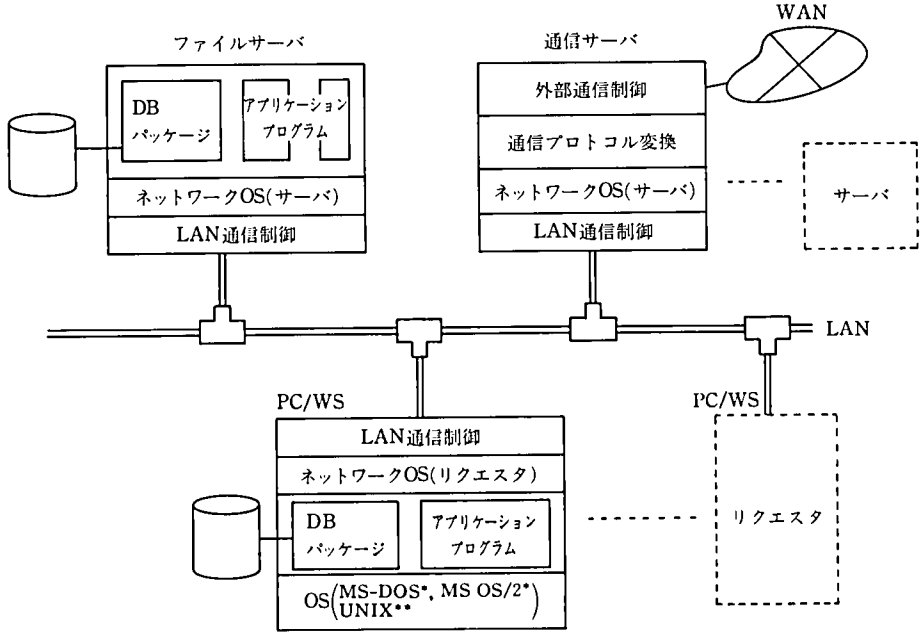
従来のメインフレーム集中のシステムとネットワーク・コンピューティング・システムとの形態上の比較を図 1 に示す．従来のメインフレーム集中のシステムでは，メインフレーム上の既定のアプリケーション・プログラムを使用している場合はもちろんのこと，タイムシェアリングで使用している場合でも，操作者は端末機を通して「メインフレームを呼び出して使用している」という意識を持つ．それに対してネットワーク・コンピューティング・システムでは，操作者は結果的にはメインフレームの CPU をタイムシェアしていても，クライアントとしての「PC/WS を使用している」と思える環境を提供する．すなわち，操作者は PC/WS の使い勝手（あるいは PC/WS の「文化」）の延長として結果的にメインフレームを使用する．

次に，ネットワーク・コンピューティング・システムの外形的特徴は次のようである．

1) オープンプロダクトを取り入れた複合システムである……データベース・サーバ，コミュニケーション・サーバ，文書作成用 WS，基幹データ収集用 WS 等，明確な役割を持ったコンピュータ群で構成される．すなわち一般には，膨大なデータ資源およびプログラム資産を維持するメインフレームと，特定機能に特化し




（メインフレーム集中のシステム）




（ネットワーク・コンピューティング・システム）

* MS-DOS, MS OS/2 は米国マイクロソフト社の登録商標である.
** UNIXオペレーティング・システムは UNIX System Laboratories, Inc. が開発し, ライセンスしている.

**図 1　メインフレーム集中のシステムとネットワーク・コンピューティング・システム**

Fig. 1　Mainframe oriented system vs network computing system

てオープンマーケット向けに製造され流通しているマイクロプロダクトで構成される．

2) マルチベンダ・プロダクトで構成されている……マイクロプロダクトのハードウェア・ベンダおよびソフトウェア・ベンダは，階層化した特定の機能の提供に特化する．したがって，複合システムは結果としてマルチベンダになる．

3) LAN で相互に結合されている……機能の階層化は同一プロダクト内の機能の独立性（凝集度）を高める．これは，ある機能階層（N）を提供するプロダクトにとって，N 層を経由して他のコンピュータ（具体的には PC/WS 等のクライアント）上の隣接する機能階層にサービスすることは，自分が存在するコンピュータ(具体的には UNIX*，メインフレーム等のサーバ)上の隣接する機能階層(N＋1）にサービスすることと原理的に同一と見なせる，ことを可能にする（図 2）．このように，LAN は複数のコンピュータを接続し，クライアントとサーバの間での機能分散を実現する．これは機能，性能および容量についての拡張性と柔軟性を増大させる．

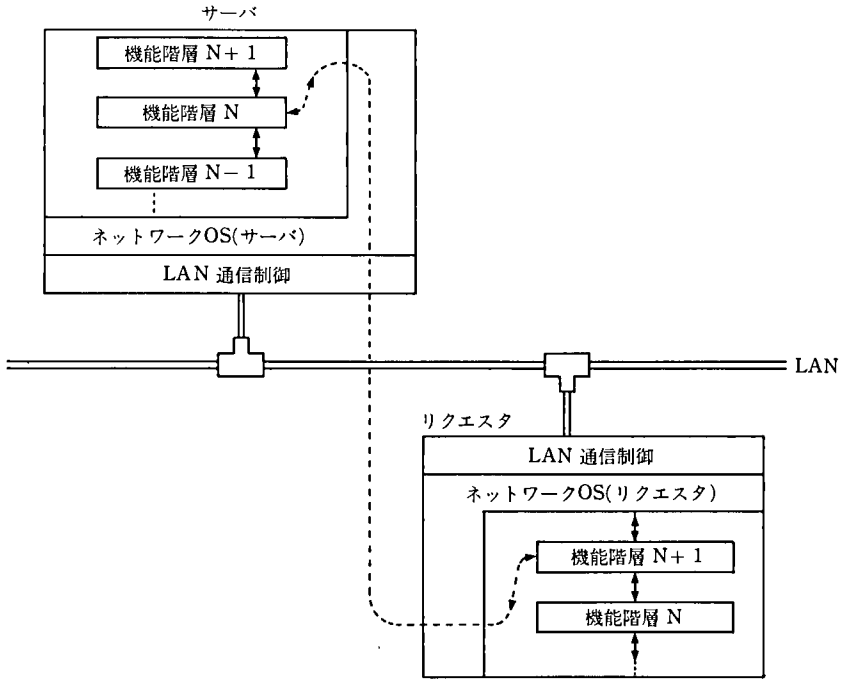



**図 2 LAN を経由した複数機関での機能の分担**

Fig. 2 Burden sharing of a function among the multiple computers connected with each other through LAN

4) GUI (Graphic User Interface) を用いている……GUI の適用によって WS/PC の操作者の習熟期間の短縮化，操作の正確性の向上が図れることは，多くの実験・研究で指摘されている[5]~[7]．PC/WS 上のアプリケーション・プログラム (AP) からの文字表示および描画の所要時間は，メインフレーム上の AP

* UNIX オペレーティングシステムは，UNIX Laboratories, Inc. が開発し，ライセンスしている．

に比べて 1/100～1/1000 倍である[8]．これは PC/WS では，AP が直接に表示機構にアクセスできるからである（図 3）．すなわち，システムの表示部・操作部を PC/WS に移行することによって GUI が可能になる．

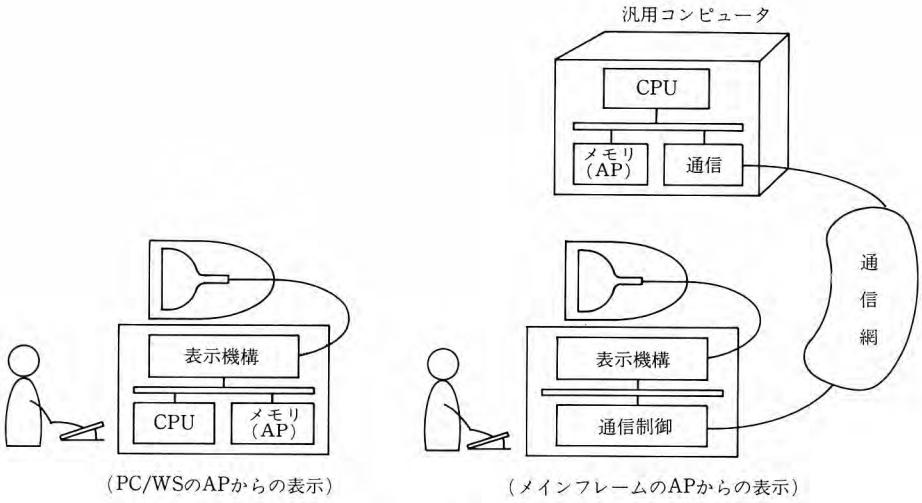



**図 3　PC/WS とメインフレームの表示機構の相異**

Fig. 3　Differences of the data display mechanism between PC/WS and the mainframe

## 2.2　ネットワーク・コンピューティング・システムの優位性

1)　コストパフォーマンスの優位性……マイクロプロダクトのメインフレームに対するコストパフォーマンスは 10 倍以上とされている[1][2][8]．これは，集積度が 2～3 年間に約 2 倍ずつ向上するという半導体製造技術の驚異的な進歩に負うところが多い．今後少なくとも 10 年はこの傾向が持続すると予想されている．

またそれに加え，オープン・システムの実現はベンダのより大きな市場の獲得を可能にし，それが大量生産による規模の効果を生み，規格化，事実上の標準化（de facto standardization）を進展させている．さらにこれらがメーカ間の価格競争を生み，マイクロプロダクトのコストパフォーマンスを促進している．

2)　特化したあるいは最新の機能の提供……機能の階層化の進展と，その階層化した機能の外部仕様の規格化あるいは公開は，ハードウェアおよびソフトウェアの開発範囲の局所化と製品寿命の安定化をもたらす．これは中小規模の IHV（独立系ハードウェア・ベンダ：independent hardware vender）および ISV（独立系ソフトウェア・ベンダ：independent software vender）による，需要が広範だが希薄にしか存在しない製品分野（niche）あるいは最新技術を適用した先端分野への参入，およびプロダクトの開発期間短縮化をもたらす．

3)　企業の単位組織の自律的戦略活動に対する適合……製品，サービス等の短ライフサイクル化，マーケットのグローバル化等により，企業内の単位組織の中小規模化とそれの自律的戦略活動が要請されている[9]．

LAN で結合された PC/WS およびサーバは，基幹系システムの端末機，通信ノード機としてのみならず，自己完結した一つのコンピュータ・システムとして単

位組織の占有システムにもなる．また，オープン・プロダクトの本来的特質からPC/WSの機器の増設，周辺機器の設置，ソフトウェア・パッケージの導入，外部通信網との接続等について，その組織の目的に合致する構成・機能に組み立て運用することが可能になる．

また，PC/WSのGUIをベースにしたエンドユーザ・コンピューティング環境の整備は，単位組織内での情報処理能力（情報リテラシ）を向上させ，情報処理の自己完結度を向上させる．

以上述べたネットワーク・コンピューティング・システムの外形的特徴と優位性の関係を図4に示す．

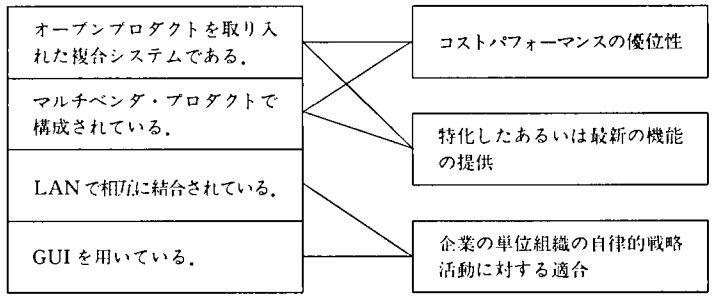



**図 4　ネットワーク・コンピューティング・システムの外形的特徴と優位性**

Fig. 4　The outward characteristic and merits of the network computing system

## 3. ネットワーク・コンピューティング・システムのデータ格納のダウンサイジング化

前章でネットワーク・コンピューティング・システムの優位性の一つとして，企業の単位組織の自律的活動への適合を指摘した．しかしこの適合は，コンピュータが計算機室を出てオフィスの中に入って来るだけで実現できることでないことは，言うまでもない．

工場における主要な生産要素は原材料，機械設備および人（訓練）であるが，情報の生産もそれにならえば，データ（＝原材料），ハードウェアおよびソフトウェア・パッケージ（＝機械設備）および人（訓練）である[10]．

本章では，企業の単位組織が多彩で高度な情報を生産するのに適合するために，それの原材料としてのデータはどのように整理して格納すればよいかを検討する．

### 3.1 データベース中心のシステム構築

従来のシステム構築アプローチでは，処理プロセスを設計の中心に置き，かつ処理プロセスに対する入力・出力がその外部仕様を決めていた．そしてデータファイルは，最終的な出力を得るための一時的なデータ格納と見なされていた．つまり原材料であるデータは特定の処理プロセスに強く結び付けられていて，「共通部品」あるいは「標準部品」という意識は薄かった．以降，本稿ではこのような開発アプローチをPOA (Process Oriented Approach＝処理中心開発アプローチ）と呼ぶ．

それに対して，新しい開発アプローチはシステムの中心にデータ格納を置く．すなわちデータは，発生・更新・消滅等のライフサイクルが定義され，それに従って保守

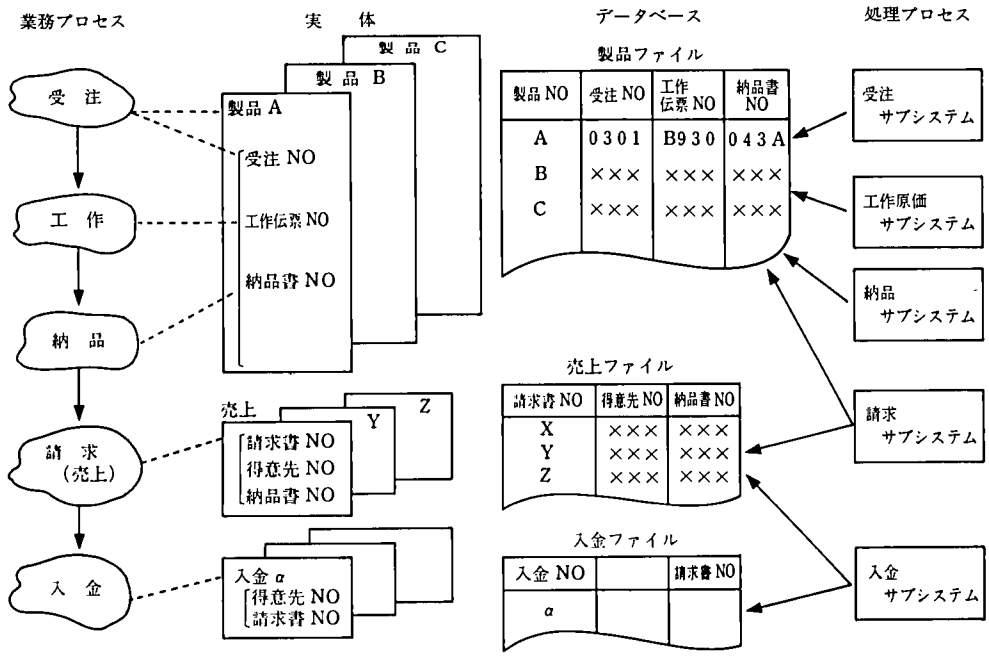



**図 5　新しい（DBOA）開発アプローチ**

Fig. 5　New (DBOA) development approach

されるべきであり，また参照・加工等に際して扱いやすい形で格納されるべきであることをシステム構築の基本に置く．このようなシステム開発アプローチをデータベース中心のシステム設計法（以降，DBOA＝Data Base Oriented Approach）と呼ぶ．以下にその概要を述べる．

DBOA は，システムが「計算の対象とする事物」（以下，実体 entity と呼ぶ．たとえば，「製品」という“もの”あるいは「差し立て」という“こと”）に着目する．そしてデータは実体の属性（attribute）であり，たとえば「製品コード」は実体「製品」の属性の一つである，との観点に立つ．すなわち，図 5 に示すように製品（実体）が差し立てされたことは，製品ファイル中の，対応する製品コードを持つレコード中の工作伝票 NO の項目が定義（＝入力）され，また製造が終了して納品の状態になったことは，納品書発行 NO の項目が定義（＝入力）されたことで示される．

従来のシステム開発アプローチ（POA）では，工作伝票ファイル，納品伝票ファイル等が別々に存在する．図 6 にそれを示す．

図 5 と図 6 を比べれば明らかなように，新しい開発アプローチではファイルは実体のN次の属性空間を形成していて，ファイルの 1 レコードはN次元空間の元とみなされるのに対して，従来のシステム設計法では 1 次の属性空間で 1 ファイルを構成している．それを図 7 に示す．

このようなファイルの構成法の相違は，当然に「処理」の相違を生む．以下に例で示す．

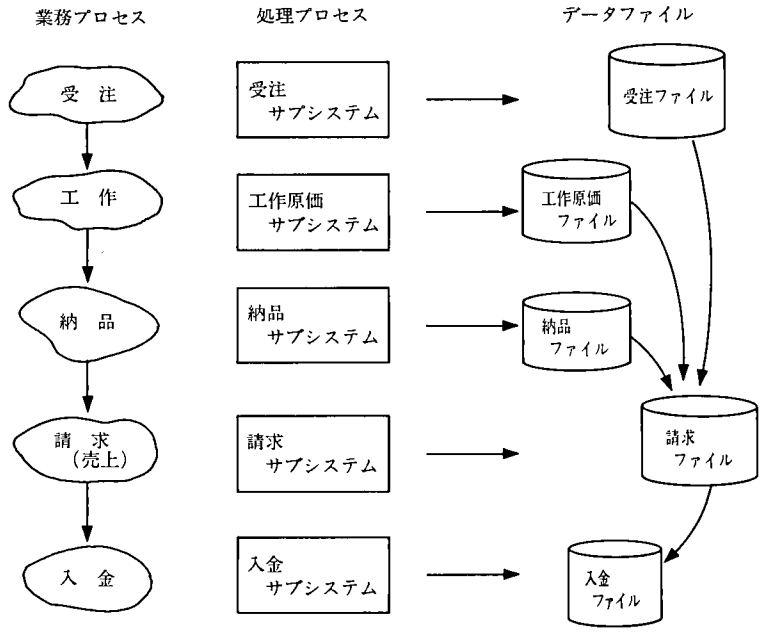



**図 6 従来の(POA)開発アプローチ**

Fig. 6 Traditional (POA) development approach

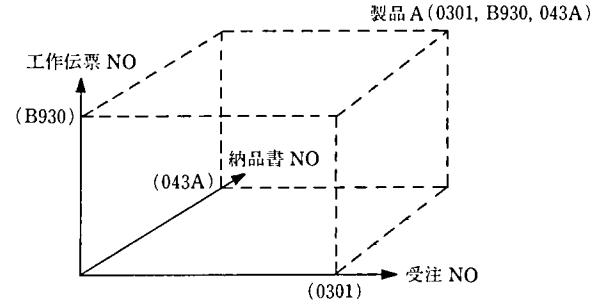



**図 7 製品ファイルのベクトル表現例**

Fig. 7 An example of vector representation of the entity

(例1) 先々月末日の仕掛かり製品原価を算出する.

DBOA：先々月末以前に工作伝票 NO が定義されていて，かつ納品伝票 NO が未定義かあるいは，先月以降に納品伝票が発行されたレコードを抽出することによってなされる.

POA：毎月末に工作伝票ファイルと納品伝票ファイルを突き合わせて仕掛かり品ファイルを作成しておく.

(例2) 当月の納品額合計を算出する.

DBOA：当月に発行された納品伝票 NO を持つレコードを抽出し，納品金額について合計をとる.

POA：納品書ファイルから納品金額の合計を求める．

この例からも明らかなようにDBOAのPOAとの主な相違は次のようになる．

① DBOAは中間ファイルを生成しない．

従来の設計法では，xx月分納品書ファイル等のように作成しようとするレポートに対応したファイルを生成する．それに対して新しい設計法では，レポートはデータベース上の「ビューの生成」であるからファイルは生成しない．

② DBOAは「締め」が無い．

従来の設計法では（中間）ファイルを生成するので，「締め切り日」を設定する必要がある．これに対して新しい設計法では，ファイルを生成しないのでデータを欲しい時点でビューを生成することになる．

### 3.2 DBOAとPOAの比較

磁気テープが外部記憶として用いられた時代は，POAは最良の，というよりも唯一のアプローチであった．高速大容量の磁気ディスクの使用および分散化処理を前提とするシステムにおいてはどうであろうか．以下にパフォーマンス，システムの柔軟性，データの透明性の側面から検討する．

1) パフォーマンス……同一の環境の下では処理のパフォーマンスはPOAのほうが有利であるとされる．それは以下の理由による．

POAではファイル中のデータは，一般に処理の対象となっているもののみで構成されている．また，データレコード中の項目もその処理に必要なもののみである．データの整列順はレポート上の出力順にあらかじめなされている．

それに対してDBOAでは，集合として「実体」を構成するすべての元（たとえば，完成製品，仕掛かり製品，不良製品，見積り中等の「仮想的な製品」を含むすべての「製品」）を処理対象ファイルに含み，それから処理対象データを抽出することになる．

しかし，ネットワーク・コンピューティング・システムのような分散システム環境では，一般にデータ量が「分散」の度合に応じて減少するので，DBOAのパフォーマンスについての不利は操作者にとっても問題でなくなる．

2) システムの柔軟性……システムの変更の要因として，およそ次の項目があげられる．

① データ入力の時期，内容あるいは，レポートの出力の時期，内容等の変更

② 計算内容の変更（たとえば税率の変更）

③ 原材料・製品種目/種別等の変更にともなうデータ項目の増減

POAは，その名が示すように各々の処理の設計からファイル構造を導き出す．処理の時期の変更は，前処理の出力結果が使えなくなる可能性があり，その場合，変更規模は当該処理だけにとどまらない．また，計算内容の変更，データ項目の変更は処理プログラムの変更に直接結びつく．POAの場合は，とくに同一の計算式が複数の処理プログラムに現れやすい．したがって，データ項目の増減も複数の処理プログラムの変更をもたらすことが多い．

一方，DBOAではデータのライフサイクルの制御機構は個々の業務処理プログラムから切り離されるので，これらの影響が局所化できる．

3) データの透明性……前述のように，DBOA はデータ保管について，企業内の「実体」を洗い出し，その実体の属性値としてデータを整理する．また，データのライフサイクルの定義とそれにもとづいた保守を行う．したがって，DBOA はシステムに関与する人々すべてに，システム内のデータについて同一の理解を可能とする．

これはあたかも工場において，工場の各現場で部品の標準化によって部品の格納場所，在庫管理者，発注点・発注量等の発注方式，等の手順の標準化が図られることに相応する．このような点からもコンピュータ・システムにおけるデータ格納の重要性が理解できる．

### 3.3 ネットワーク・コンピューティングと DBOA

DBOA およびそれに類する開発アプローチは，基幹系を含む情報システム一般に適用可能なものであり，ネットワーク・コンピューティングに強く関連するものではない．しかし，DBCA は前述のように，システムの柔軟性，データの透明性が POA に比べて大きく，単位組織の自律的活動に適合する特質を持っている．また，最大の課題であるパフォーマンスについては，全社集中から単位組織へ分散することにより，軽減されると考えられる．

このように，DBOA 開発アプローチはネットワーク・コンピューティング環境下では，有効な開発アプローチであると考えられる．

## 4. データベース・ソフトウェア・パッケージの利用

情報を作成・利用する人々が自らデータを入力しなければならないようなエンドユーザ・コンピューティングは，いわゆる OA 処理の域を出ない．真のエンドユーザ・コンピューティングは，基幹系のデータも含み企業内に格納したデータを，その存在場所を越えて，どこからでも操作者に特別の操作を要求することなくアクセス可能にするような仕組み，およびそれらのデータを加工するためのソフトウェア・パッケージの整備が必要となる．

システムをオープン・プロダクトで構成することの大きな利点の一つとして，多数の流通ソフトウェア・パッケージを利用できることが挙げられる．

前章では，情報処理システムの「原材料」であるデータをどのように格納していくかについて開発アプローチの視点からその変革の利点を述べたが，本章では「機械設備」に相当するソフトウェア・パッケージについて，データベース・ソフトウェア・パッケージを「処理」の中心に据えることの利点を論じる．

### 4.1 「データ」把握の階層

コンピュータ・プログラムの究極的な目的はデータの変換処理である．したがって，「処理」を遂行するプログラムの内容を把握するには，入出力データの性質をとらえることが鍵となる．文献[11]によれば，入出力データの性質は「入出力データの情報構造」「入出力データの表現方法」「入出力データのデータ・アクセス方法」の三つの階層に分けてとらえられる．本稿もそれに準じて考察を進める．まず，以下にこの三つの階層の概要を述べる（図 8）．

1) 入出力データの情報構造……「計算の対象とする事物」や「事物相互の関係」

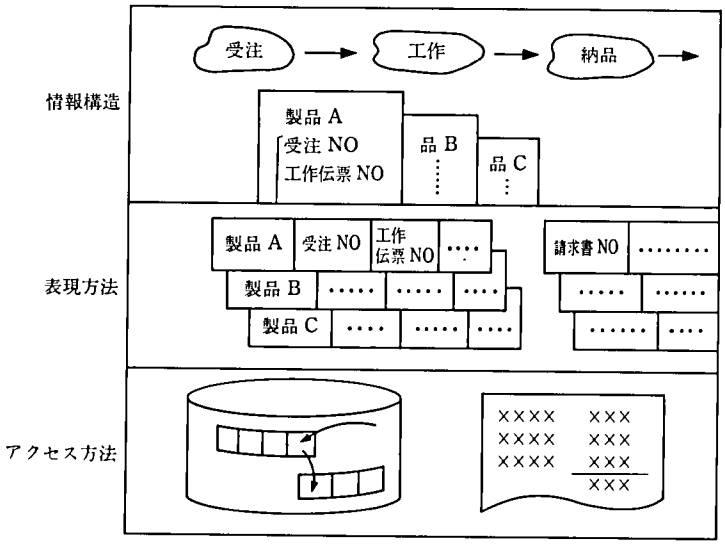



図 8　入出力データ把握の三階層

Fig. 8　Three layers of Input/Output data descriptions

の存在の枠組である．すなわち情報は，「製品」いう“もの”あるいは「差し立て」という“こと”等に対応し，それらを基準にしてその存在自体あるいは相互の関係が認識される．認識された“もの”あるいは“こと”（以下，「実体」という）は，「製品番号」，「工作指示書 NO」「個数」等の属性によって一意に識別され，あるいは状態が表現される．

2)　入出力データの表現方法……認識された「実体」とそれに対応する「属性」がコンピュータが扱う「入出力データ」となる．この入出力データのデータ項目の並び方や文字・数値・整数等のデータ形式を定め，実体との対応関係を定めたものを入出力データの表現方法という．

3)　入出力データのアクセス方法……データがファイルやディスプレイ，印書装置等に入出力される物理的な形式あるいはアクセス方式を規定する．すなわち，レコード長，入出力の区分，順/ランダム編成等の編成方式等を規定する．これを定めたものを入出力のアクセス方法という．

### 4.2　データベース・ソフトウェアの特徴

MAPPER 等の 4 GL，あるいは Informix*，ORACLE**等のリレーショナル・データベース・ソフトウェア（以下まとめて「パッケージ・ソフトウェア」と呼ぶ）は COBOL 等の 3 GL を用いてプログラミングするのに対して，大幅な工数の逓減をもたらす[11]~[14]．それは前節のデータの三つの階層について，パッケージ・ソフトウェアは次のような機能を有するからである．

1)　データのアクセス方法

①　データの物理的な入出力機構の隠蔽：物理的にどのように大容量記憶装置に

* Informix：米国 Informix 社の登録商標である．
** ORACLE：米国 ORACLE 社の登録商標である．

格納されているか，あるいは「削除レコード」も含み，レコード間の物理的な並びがどうであるか，レコード内の項目にデータが入りきれるか否か，等の制御はパッケージ・ソフトウェア側で行う．ユーザはそれらに関与しない．

② リカバリ機能：データ書き換え中にシステムがダウンした時のリカバリ（回復）をパッケージ・ソフトウェア側で一貫して行う．

2) データの表現方法

① 項目属性の指定：データベース上にデータ項目を定義するが，データ項目の属性として「項目名」「データ型」「データ長」等が指定できる．これは原材料の規格を指定することに相当する．すなわち，データ層に係る情報を扱っていると考えられる．ここで大切なことはデータ項目の指定は，単にデータの規格を述べているだけでそれがどう加工されるべきかということ，すなわち処理とは切り離されているということである．これが前述の「データの透明性」を確保するのに寄与している．

3) データの情報構造

① データの関係演算：複数のデータファイルを突き合わせて，照合，転記，連結（ファイルを「よこ」方向に結合させる），合併（ファイルを「たて」方向に結合させる）等の処理を関係演算というが，データベース・ソフトウェアは一般にこのような機能を有する．

また，ファイルを検索し，条件に合致するレコードのみを選択する，あるいはレコードに位置づける等の機能を有する．

② 導出計算式の属性化：データベースの「項目」に対応するデータには，「1次データ項目」と「2次データ項目」の2種類がある．「1次データ項目」とは外部からの入力によるもので，「2次データ項目」はシステムの内部で生成されるデータである．たとえば，項目の「数量」，「品名コード」は受注伝票の入力によって得られ，また「単価」は「品名コード」をキーにして単価表から得られ，「金額」は「単価」×「数量」で得られるとすれば，「数量」「品名コード」は1次データ項目，「単価」「金額」は2次データ項目である．2次データ項目を「導出項目」ともいい，1次データ項目から導き出す計算式を導出計算式と呼ぶ．

一部のとくにPC系のデータベース・ソフトウェアでは，項目の「属性」として導出計算式を定義することができる．図9にワークステーションPW²の「EXLIST」の例を示す．

導出計算式の項目属性化によって，従来はCOBOL等のプログラム内部に持っていた2次データ項目の生成のロジックを切り離して，データ項目定義の一部とすることができる．この意味で「データ・カプセリング」に一歩近づいているといえる[15]．

③ データの「情報化」機能：データの抽出，集計，関数当てはめ等の解析，分析のための機能，またそれらの結果の表形式レポートの出力，ビジネスグラフの出力，等の機能を有する．

④ セキュリティ管理機能：使用者に対し，「参照」「更新」等の区分，あるいはファイルの機密区分，使用者の権限等に応じてセキュリティのチェックを行う．

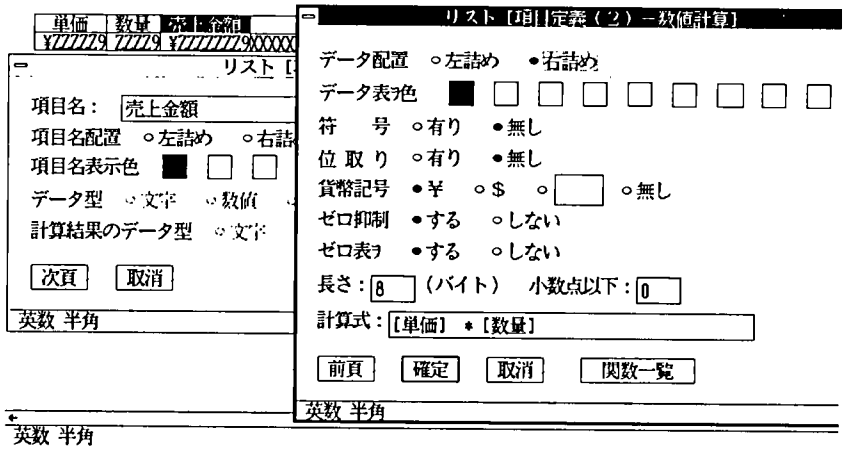



**図 9　導出計算項目の指定の例 (EXLIST)**

Fig. 9　An example of the derived data definition

### 4.3　データベース・ソフトウェアの利点

従来のプログラム作成法でも，アクセス層はライブラリ等で論理化がされていたが，情報層とデータ層は分化されていた．(混然として一つのプログラム・ロジックの中に埋め込まれている．)

データベース・ソフトウェアは，まずアクセス層についてはユーザから完全に隠蔽されている．次にデータ層については，ユーザがデータベースのスキーマの定義をするのみで，データ層に関する処理はデータベースが行う．すなわち，ユーザは情報層の操作のみに注力できる．しかも，情報層の基本的な処理は前述の検索，照合等の「データの関係演算」「導出計算式」あるいはグラフ表示・印書等の「情報化機能」等のツールが用意されている．

その結果，次のような利点が得られる．

1)　「業務処理」に注力できる……前述のように「論理レベル」でデータを扱えるので，コンピュータのハードウェアあるいは基本ソフトウェアに関して深い知識なしに，データベース・ソフトウェアの利用が可能になる．

これは，パソコン上でワープロ・ソフトウェアを使う場面を考えてみれば直ちに了解できる．ユーザは MS-DOS* 等の知識なしに使いこなしている．同様に，データベース・ソフトウェアについてもユーザは「どのように動かすか」にではなくて，「どのように利用するか」により注力できる環境を作り上げつつある．

2)　プログラム作成の逓減……ファイル同士の関係演算によるデータファイルの操作，条件によるデータの絞り込み，検索等の機能を活用することによりプログラミング労力を大幅に軽減できる．また，帳票作成・画面作成ユーティリティによるプログラミングの大幅な削減等が図れる．

3)　きめの細かい情報が得られる……利用者自らがデータの加工を行うことによって，利用目的により合致した情報が迅速に得られる．

* MS-DOS：米国マイクロソフト社の登録商標である．

## 5. システム部門の役割の変化

以上述べたように，ネットワーク・コンピューティング・システムへの移行は単に現行システムをLAN上のPC/WSに移行するのではなく，データ処理の設計の枠組の変革である．

まず部品の標準化にあたるデータファイルの設計とライフサイクルの定義，保守基準の設定を行う．そのポイントは「実体」とその「属性」の把握にある．工業製品では，モデルチェンジがあっても大部分の部品は変更なく使用される．同様に，実体と属性が的確に把握され，それをもとにデータベースが構築されるならば，組織変更，手続き変更に際し，データベースの変更はあるにしてもごくわずかなものに留まるであろう．組織が変わろうとも，手続きが変わろうとも，「作業者」，「製品」，「受注」，「差し立て」等の「実体」と「属性」のそれ自体の枠組は，企業活動自体が変わらない限り変わることはない．(変わるのは「属性」の取り得る値の範囲，たとえば，「製品種類」等である．実体としての「製品」は，たとえばサービス業に事業転換しないかぎりなくならない．)

次に，生産設備に相当するハードウェアおよびソフトウェアの選定・導入を行う．そのポイントは，エンドユーザ・コンピューティングへの適合性の評価にある．得たい情報が戦略的なものであればあるほど，その情報の構造は不明確である．したがって多くは「発見的な」解析アプローチを必要とするので，このような情報処理の主役はエンドユーザとなるであろう．解き明かした情報構造をエンドユーザ自らがそれを「データ表現」して処理する，そしてより高次の情報構造の解析につなげる，というサイクルをとるだろう．

重要になってくると思われるのが人（＝訓練）である．企業のダイナミズムを増すためには，コンピュータの活用を自律的な組織にとっての基礎的な「読み書き能力」(リテラシ）とすることが求められる．ネットワーク・コンピューティングでは，ネットワーク上のコンピュータ資源を「PCの文化」でアクセスできるため，リテラシの主たる内容はPC上のソフトウェアの使用法・活用法，および情報構造の解析手法にあると考えられる．

このように，これからの情報部門の役割として，「エンドユーザが作りやすい環境の提供と教育」が重要な位置を占めると考えられる．

## 6. おわりに

情報システムのネットワーク・コンピューティング・システムへの移行は，本質的には情報の作り手と使い手の同一化を意味している．それは，あたかもプロの運転手のいるバスから自家用車への乗り換えを意味する．情報システム部門の役割も「お客をスケジュール通り目的地へ安全に運ぶこと」から，「プロでなくても使いこなせる車の整備，運転の教育等」になるであろう．

しかしながら，ネットワーク・コンピューティングは未だ緒についたばかりの技術であり，課題も多い．世界最大のコンピュータ・トレード・ショウの一つであるCOMDEXの91年春のテーマはネットワーク・コンピューティングであったが，そこで多くの米国のネットワーク・コンピューティング先進ユーザは，開発の迅速化，投下資

本の早期回収化，全社データベースへの容易なアクセス等の優位性とともに，次のような課題を指摘していた．

- 分散データベースが未だ弱い
- ソフトウェアの配布機能が弱い
- CASE が用意されていない

われわれメーカは，オープン・プロダクトを主軸にしてそれをより価値あるものにする商品の開発，サービスの提供に努めたい．

---

**参考文献**

[1] Buisiness Week/November 26, 1990 "Rethinking the computer with superchips, the network is the computer"
[2] 日経エレクトロニクス,「分散コンピューティング環境の最前線」日経 BP, 6-11, 1990.
[3] 日経コンピュータ,「クライアント・サーバー　システム」日経 BP, 4-23, 1990.
[4] 日経コミュニケーション,「90 年代のコンピュータ環境を構築するネットワーク・コンピューティング」日経 BP, 1991.2.18.
[5] PC WEEK,「ユーザインタフェースの臨床実験　GUI は明らかに生産性を高める」1990.8.15&22
[6] B. Schneiderman,「ユーザ・インタフェースの設計」, 日経マグロウヒル, 1987.
[7] 守屋慎次,「ユーザインタフェース技法」情報処理 Vol. 29, No. 10, Oct. 1988.
[8] 佐藤博,「情報システムにおけるマイクロプロダクトの位置づけに関する考察」
[9] 「1990 年代の OA＝次世代オフィス・システムに関する調査研究報告書」(社)日本オフィスオートメーション協会　H 2.5.10
[10] 堀内一,「データ中心システム設計」オーム社, 1988.3.
[11] 橋本正明,「データ中心のプログラム仕様記述法」井上書院, 1988.
[12] 岩波講座　情報科学 8「情報の構造とデータベース」オーム社, 1988.7.
[13] C. Date "An Introduction to Database System", Addison-Wesley Publishing Company, 1986.
[14] 大賀節雄,「データベースと知識ベース」オーム社, 1983.
[15] 堀内一,「データ中心型システム設計」, 情報処理学会研究報告 88-IS-21.

**執筆者紹介**　佐　藤　　博 (Hiroshi Sato)

1967 年武蔵工業大学卒業. 同年日本ユニシス(株)入社. 9000 シリーズ, S 80 のアセンブラ・プロセッサ, 通信制御ハンドラ等の基本ソフトの開発を経て, 1976 年よりワークステーション・システムの開発に従事, 現在 OA ソフトウェア 2 部長. 武蔵工業大学非常勤講師, 技術士(情報処理), 中小企業診断士, 情報処理学会会員.

# EXOS統一操作環境の実現

## An Implemented Unified Operation Environment for the EXOS

河合昭男

要 約 EXOSはエンドユーザを対象とした統合オフィスシステムである.「使いやすさ」は,開発当初から最大のプロダクト目標の一つであった.EXOSのワークステーション側の環境基盤は,PW² ワークステーション+MS OS/2* である.マウス操作を基本としていることが特徴である.

EXOS統一操作環境の実現のために用いた手法は,「UIM」という統一操作のためのモジュールを開発したことおよび「EXOS設計作法」という外部仕様設計基準書を作成したことである.「EXOS設計作法」には,「EXOS操作憲法」と名付けた基本設計方針とEXOS操作の法律と位置付けられる全体に共通な具体的操作が記述されている.

本稿の目的は,この「憲法」に従ってどのように統一操作環境を実現していったかを具体例を挙げながら述べ,「憲法」の有効性を示すことである。具体例として,とくに重点をおいたオブジェクト指向操作について詳細に述べる.

**Abstract** The EXOS is an integrated office system designed for use by end users. 'Ease of use' has been one of the product's greatest goals since the beginning of its development. The environmental platform for the EXOS workstation is made up of a PW² workstation plus MS OS/2, and characteristically based on mouse operations.

The techniques for the implementation of the EXOS unified operation environment are represented by having newly developed a modular tool kit named 'UIM' intended for unified operation and by having drawn up the external specifications design guide called 'EXOS Design Methods.' The latter guide includes basic designing policies named 'EXOS Operation Constitution' and also describes detailed basic operations which, positioned as the laws of EXOS operation, are communal to the whole.

The purpose of this paper is to show how useful the 'constitution' is by focusing on the way in which the unified operation environment has been implemented according to the 'constitution' while quoting some examples. Specifically, detailed descriptions are given about the object-oriented user interface which has commanded the author's special attention.

## 1. はじめに

ワークステーションDS7上のアプリケーション開発に携わった反省から,使いやすさや操作の統一性の問題について,今回ワークステーションPW²上のアプリケーション開発でどのような手法で解決したかを述べるのが本稿の目的である.

一般にGUI (Graphical User Interface)[5] と呼ばれている操作環境が,これからのワークステーションの標準操作環境として注目されている.その開発環境はツールキットとスタイルガイドからなっている (図1).アプリケーションのユーザ・インタフェース部分は各アプリケーションが独自に作成せず,ツールキットと呼ばれる共通

* MS OS/2は米国マイクロソフト社の登録商標である.

ライブラリを用いる．こうすることにより，開発者にとって作業が減少し，利用者にとっては異なるアプリケーションでも操作が同一となる．しかしツールキットのみで操作を統一することはできない．たとえば，メニューバーに並べる機能の配列や文言については，ツールキットとは別に規則を決める必要がある．これらの規則を記述したものをスタイルガイドと呼ぶ．

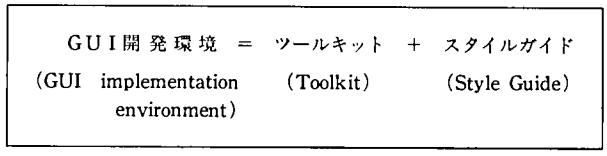



**図 1　GUI 開発環境の構造**

Fig. 1　The structure of GUI implementation environment

今回の EXOS (Excellent Office System, 統合オフィスシステム) 開発では，OS として MS OS/2 を採用した．ユーザ・インタフェース(UI)の基本ツールは，MS OS/2 のプレゼンテーション・マネージャ (PM)* として整備されている．では，この環境下で各アプリケーションが自由にデザインして統一操作環境が実現できるだろうか．ウィンドウの形や基本操作は，PM の下ではとくにアプリケーションが意図して変えない限り同一となる．しかし，われわれに与えられた課題は EXOS という統一操作環境を実現することである．PM だけの弱いしばりでなく，もう少し統一性を強調したやや強いしばりが必要である．その手法は以下本稿で述べるが，「UIM (User Interface Manager)」という統一操作のためのモジュールを開発したことと，「EXOS 設計作法」という外部仕様設計基準書を作成したことである．これらを GUI の図式にあてはめると，図 2 のようになる．

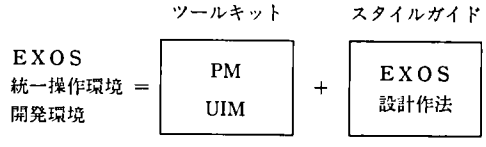



**図 2　EXOS 統一操作環境開発環境**

Fig. 2　The implementation environment of EXOS unified operation environment

## 2.　「使いやすい」とは

EXOS はエンドユーザを対象とした統合オフィスシステムである．誰にでも簡単に使えなければならない．「使いやすい」とはどのように考えればよいか．それは，「コンピュータを意識せず，本来の仕事に専念できること」という言葉に集約されている．

「使いやすい」とは何かは，逆に「使いにくい」とは何かを考えた方が理解しやすい．利用者側から見た阻害要因を次に列挙する．

1)　マニュアルを熟読しないと最低限の操作すらできない．

とくに，一般のパソコン・ソフトウェアではアプリケーションごとに操作体系がまったく異なり，たとえばワープロを覚えたから表計算もすぐに使えるという

* MS OS/2, PM は米国マイクロソフト社の登録商標である．

わけにはいかない．

2) 特殊キーに重要な機能が割り当てられている．

ここで問題になるのは，ファンクションキー等に重要な機能が割り当てられていて，それが画面に表示されない場合である．このようなアプリケーションには通常固有のキーマットが添付されているが，アプリケーションごとにキーマットを使い分けないと操作できない．

3) 操作ミスが許されない．

とくに，操作ミスでデータが消去されてしまい復元の方法がない．

これら利用者側の不満に対する開発側の考え方は以下のようであろう．

1) および 2) については，操作を同じにできればよいが，別々の開発者が作る製品の操作を統一すべき基準がない．他社製品と性能を競うには，機能の他操作性も重要であり，各々で工夫を行い結果として操作体系がまったく異なってしまうことがある．3) については，予期できない誤操作を利用者が行った場合の対応まで考えてソフトウェアの開発を行うのは困難である．結果として，「操作ミスをしないようにマニュアルを良く読んで，大事なデータを消してしまったりしないようにしてください」ということになる．

このように，今までは使用者の要求と開発側の問題との開きは大きかった．今回の EXOS 開発では，以下の章で具体的に述べるような手法で統一操作環境の実現を行い，これら開発側の問題を解決した．

## 3. 統一操作環境実現の手法

### 3.1 プレゼンテーション・マネージャ（PM）

EXOS は MS OS/2 PM を前提としている．PM によりアプリケーションの Look & Feel はかなり統一できる．主なものは，ウィンドウの形や操作，プルダウン・メニューによるアプリケーションの操作，ダイアログボックスによる操作員との対話を行う操作等である．PM は一種の GUI であるが，スタイルガイドがなくアプリケーションの自由度が高いため，GUI としては低レベルなものであり，むしろ GUI のツールとして位置付けられるものである．そのため，PM 上に統一操作環境を構築するには，独自のスタイルガイドとツールが必要である．実際，New Wave [5] 等でも PM 上に独自 GUI 構築を試みている．

### 3.2 U I M

一般的に UIMS（User Interface Management System）[4] と呼ばれているものがある．対話型システム開発に占める UI 開発コストが数 10%以上も占め[4]，使いやすさを求めるほど，その割合が上昇していく．そのため，アプリケーションの核と UI 部分を切り分け，UI の開発/実行環境を UIMS として提供して，アプリケーションの生産性と使いやすさの向上を狙うものである．一般的に GUI と呼ばれているものは，使いやすさをマウス/ウィンドウでグラフィカルに実現するもので，考え方は UIMS と同じである．

EXOS の UIM（以下誤解のない限り単に「UIM」と記述する）は，統一操作環境実現のために PM 上に構築したプログラムである．

UIM のプログラム構成は，以下に示す常駐および非常駐の実行プログラムと EXOS の各プログラムから呼び出すことができる共通 API (Application Program Interface) からなる．

1) 実行プログラム

① 常駐プログラム

ウィンドウ管理……ウィンドウ一覧の表示，整列，前面に表示等

キャビネット一覧の起動

UIM ツールの起動……電卓，時計，カレンダ等の起動

UIM ボタン……メニューの起動等カスタマイズ可

警告アイコン……メッセージが表示できない状態を操作員に通知する

メール着信アイコン……メール着信を操作員に通知する

② 非常駐プログラム

電卓，時計，カレンダ等の UIM ツール

2) API

① UIM 常駐プログラムとのインタフェース

② メッセージ表示

③ メール着信アイコン表示

### 3.3 操 作 憲 法

操作憲法は EXOS のユーザ・インタフェース基本設計思想で，主に対話型システム設計八つの黄金律[1] と TRON 作法[2] を参考に作成した．これらは共に PM の操作と矛盾しないもので，次のように定義した．

**EXOS 操作憲法**

第 1 条　操作の一貫性
第 2 条　オブジェクト指向操作
第 3 条　モードを意識しないで操作できること
第 4 条　コンピュータで判断がつくことを操作員に行わせてはならない．
第 5 条　初心者から熟練者まで満足できる操作法
第 6 条　リアルタイム性
第 7 条　逆操作を許す．
第 8 条　エラーの処理を簡単にさせる．
第 9 条　一度に多くのことを記憶することを要求しない．

EXOS 操作憲法に定義した項目は大部分一般に言われている常識的なことであるが，あらためて憲法と名付けて定義したことに意味がある．明示しておかないと，開発優先となって忘れられることが多いからである．第 2 条と第 3 条のみが，われわれ開発担当にとって新しい概念である．

操作憲法はどのように EXOS で実現されているかを見る．

第 1 条　操作の一貫性

アプリケーション開発上最も大切な原則である[6]．

EXOS は一つのアプリケーションの操作法を覚えれば，他のアプリケーションの操作法もマニュアルを見ないで推測しながら一通りは使える．

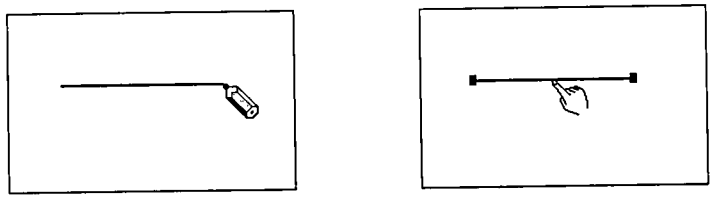
第2条　オブジェクト指向操作

オブジェクト指向操作については後述する．

第3条　モードを意識しないで操作できること

モード設定の禁止としたかったが，実現困難と考えられたため一歩譲った．モード設定がなぜ良くないかというと，モードは開発者の都合で安易に作ることが多い（モード設定の誘惑[6]）が，操作員にとっては現在のモードを意識しながら操作しなければならないのは使いやすいとはいえないからである．

EXOS 図形作成 EXDRAW では，マウスポインタの形を変えることによりこれを実現している（図 3）．鉛筆のポインタは図形描画モードを表し，手のポインタは選択モードを表す．ポインタは画面の中で最も操作員の注意を引くところなので，現在のモードをとくに意識することなく線を描く操作と図形選択操作を誤ることはほとんどない．



(a) 描画モード　　(b) 選択モード

**図 3　ポインタによるモード表示の例**

Fig. 3　Show current mode using pointer

第4条　コンピュータで判断がつくことを操作員に行わせてはならない

「その操作はできません」や「その操作の前に……を行ってください」等のメッセージを排除した．その操作が行えないよう，メニューの色を薄くする等考慮した．たとえば，「切り取り」操作を行う前には「貼り付け」メニューは薄くなっていて選択することはできない．

第5条　初心者から熟練者まで満足できる操作法

多様なユーザニーズを満足させるため，次のような機能を提供している．

1)　複数の操作方法

①　キーとマウス

メニュー選択はキーとマウスどちらでも操作可能とした．

アプリケーション内部は必ずしも完全にどちらか一方で操作できるようにはなっていない．

マウスで文字入力したり，キーで図形を描くのが使いやすいとは考えにくい．

②　短縮操作

上級者に対してはメニュー階層のショートカットを可能とした．

たとえば，カット/ペーストはプルダウンメニューを表示しないでも CTRL+INS と SHIFT+INS のキー操作で実行できる．

2)　ユーザ・カスタマイゼーション

① 業務選択等のメニューはユーザが自由に作ることができる.

② ログインにより自分の操作環境で使用することができる.
UIM ボタンやベースフォルダ等利用者単位に設定できる.

第6条　リアルタイム性

「操作員の操作には，常に即座に何らかの反応を返し，見捨ててはならない.」のが憲法である．時間のかかる処理には，ポインタを砂時計にするかメッセージを表示した．しかし，実際に使ってみてリアルタイム性は EXOS の一つの問題点となっている．アプリケーションのみならず，OS とハードウェアまで含めて今後の課題である．

第7条　逆操作を許す

「編集」プルダウンメニューの「復元」メニューを選択することにより，すべての操作について，その操作を行う前の状態に戻れるようにした．また，どうしても復旧できない致命的な操作については，操作員の確認をとるようにした．これで，操作ミスによりデータがなくなるような事故を防ぐことができる．

第8条　エラーの処理を簡単にさせる

エラーメッセージはエラーの原因と対処法を表示するよう努めた．

第9条　一度に多くのことを記憶することを要求しない.

G. Miller は 1956 年に発表した論文の中で，「人間の短期記憶領域は 7±2 である[1] 」と言っている．操作員は，本来の自分の仕事に専念できるよう，オブジェクト指向操作やモードレス操作等によりコンピュータを意識しないで操作できることを狙った．

## 4. オブジェクト指向操作

オブジェクト指向操作[3] が注目されている．オブジェクト指向操作は人間の自然な思考に近く，使いやすいと言われている．PM がオブジェクト指向操作であり，EXOS もオブジェクト指向操作を目指した．

### 4.1 オブジェクト/アクション方式

今までの操作では，たとえばファイルの削除をコマンドラインから入力する場合，「A＞DEL FILE」というようにコマンド（アクション）を先に指定してからその操作対象（オブジェクト）を指定する操作が普通とされてきたが，人間の思考法はその反対で，「何」（オブジェクト）を「どうする」（アクション）という順序が自然である．とくに，マウス操作を基本としたグラフィカルな操作環境の下では，コマンドを意識せず操作できるため，人間の自然な思考法にそった操作が一層大切となる．

その典型例は直接操作[1] である．たとえば EXOS では，キャビネット間の複写操作はオブジェクトをマウスでドラッグするのみで実行できる．利用者はまったくコマンドを意識する必要はない．しかしながら直接操作の実現には限界がありコマンドを完全に隠すことはできない．次に，今までの典型的操作と EXOS で採用したオブジェクト指向操作を実例を挙げて比較する．

(例1)　カット/ペーストの操作例

カット/ペーストは共有メモリを介して画面のデータを複写する機能である．

この操作について今までの典型的操作としてDS7イメージ処理U-IMAGEとEXOSイメージ処理EXPAINTを比較する．

DS7のU-IMAGEでは，次のようにコマンドを先に選択する操作となっている．

① 「読取り」メニューを選択する．
② 複写する対象（from）を指定する
③ 「貼付け」メニューを選択する．
④ 複写先（to）を指定する

EXOSのイメージ処理EXPAINTではこの操作は，次の2パスのオブジェクト/アクション操作となっている．

① 複写する対象（from）を指定する（オブジェクト）
② 「編集」プルダウンメニューの「読取り」メニューを選択する（アクション）
この操作で，指定したデータがクリップボードに読み込まれる．
③ 複写先（to）を指定する（オブジェクト）
④ 「編集」プルダウンメニューの「貼付け」メニューを選択する（アクション）
この操作で，クリップボードのデータが指定した位置に貼り付けられる．

(例2) 文字列複写操作例

文字列の複写例としてPC9801* 上の一太郎** と，EXOSワープロEXWORDを比較する．

一太郎では，次のようにコピーコマンドを先に選択する操作となっている．

① 「コピー」メニューを選択する．
② 複写するもとの文字列を選択する．
③ 複写先を指定する．

EXOSのEXWORDでは次のような操作となる．

① 複写するもとの文字列を選択する．
② ツールボックスの「複写」アイコンを選択する．
③ 複写する位置を指定する．

この操作では③の後のコマンドは省略でき，カット/ペーストより操作は簡単である．

(例3) 図形複写操作例

図形の複写例としてPC9801上の花子*** とEXOS図形作成EXDRAWを比較する．

花子では，一太郎同様コピーコマンドを先に選択する操作となっている．

① 「コピー」メニューを選択する．
② 複写するもとの図形を選択する．
③ 複写先を指定する．

EXOSのEXDRAWでは次のような操作となる．

① 複写するもとの図形を選択する．
② 「複写」ボタンを選択する．

* PC9801は日本電気(株)のパーソナルコンピュータである．
** 一太郎は（株)ジャストシステムのワープロソフトウェアである（登録商標)．
*** 花子は（株)ジャストシステムの図形作成ソフトウェアである（登録商標)．

③　複写する位置を指定する．

(例 4)　図形の属性指定の操作例

一度描画した図形の属性変更を行う操作例として，前述の花子と EXDRAW を比較する．

花子では次のようにコマンドを選択したのち対象を指定する．

①　「変更」メニューを選択する．

②　ダイアログでカラー，ハッチングパターン等の属性を指定する．

③　変更する対象となる図形を指定する．

EXDRAW では次のようなオブジェクト/アクション操作を行う（図 4)．

①　変更する対象となる図形を選択する．

②　ツールボックスを使ってカラー，ハッチングパターンの属性を指定する．

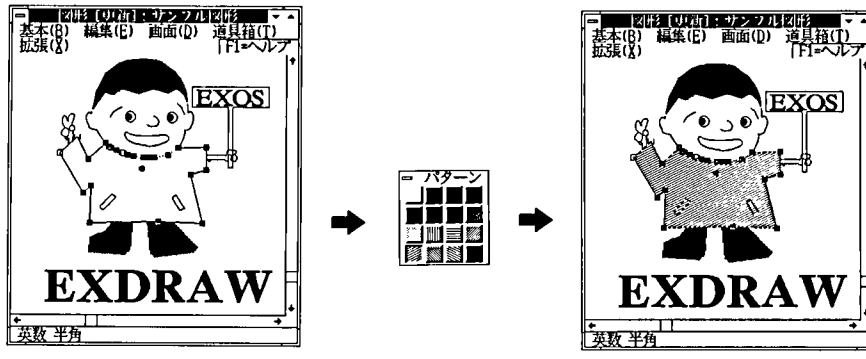



図 4　ハッチングパターンの操作例

Fig. 4　Change hatching pattern

以上の例で明らかなように，今までの伝統的操作では「どうする＝コマンド」を先にしているが，EXOS では「何＝操作対象」を先に指定しているのが大きな特徴である．

伝統的操作の特徴はガイドライン・メッセージが表示されることである．たとえば複写コマンドを選択すると「複写元を指定してください」と表示され，この操作が終わると「複写先を指定してください」と表示される．EXOS ではこのようなメッセージは表示されない．その理由は，操作対象が先に選ばれた状態では操作員がそれをどうしたいのかコンピュータには不明であり，適切なメッセージが用意できない．次にコマンドを指定した時点ではメッセージを表示することは可能である．しかしここで，たとえば「複写先を指定してください」というメッセージが本当に必要かどうかである．この時点で操作員はすでに複写元と複写コマンドを指定しており，次は複写先を指定するしかなく自然に操作できるのである．

## 4.2　キャビネットとオブジェクト

一般のパソコン・ソフトウェアには必ずファイルメニューがあり，その中には「新規作成」と「オープン」の機能がある．EXOS の各オブジェクトクラスに対応するアプリケーションには「ファイル」メニューがない．ではどのように「新規作成」と「オープン」の機能を実行しているか，まず本節ではオブジェクト指向の「オープン」に

ついて述べ，次節でオブジェクト指向の「新規作成」について述べる．

今までの典型的なワープロ等のアプリケーションでデータの更新を行うための操作は，まずプログラムを起動し，次にデータを読み取るという操作が普通であった．オブジェクト指向操作では，操作員はデータを指示するのみで適当なプログラムが自動的に起動され，データが読み込まれる．EXOS ではこれを次のように実現した．

文書処理，図形処理，イメージ処理等のアプリケーションと対応して，文書，図形，イメージ等のクラスを設定した．それぞれのアプリケーションで作成したデータをそれぞれのクラスに属するオブジェクトと呼び，キャビネットが管理を行う．

キャビネットは階層構造で，以下ドロワ，フォルダと呼ばれる．操作員があるフォルダに保存されているオブジェクトを更新するためには，そのフォルダを開いてオブジェクト一覧を表示し，適当なオブジェクトをマウスでダブルクリックする．後はキャビネットがそのオブジェクトのクラスを見て，対応するアプリケーションを起動してデータを渡す．つまり，操作員はオブジェクトを指示するのみで，どのアプリケーションを起動するかを考える必要がないわけである（図 5）．

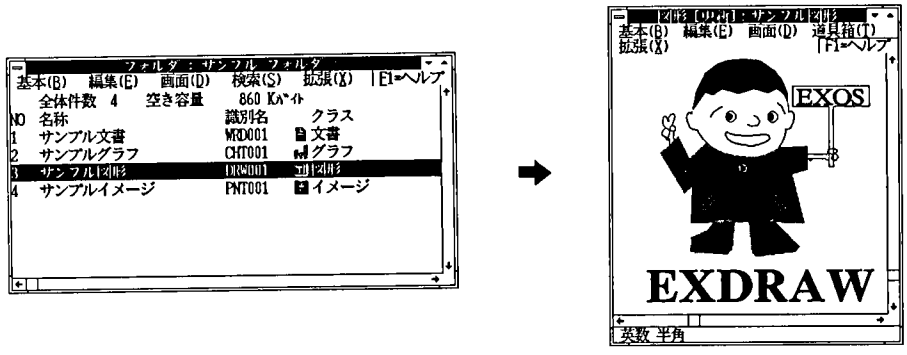



図 5　キャビネットからのオブジェクト更新操作

Fig. 5　Open a object in a cabinet

### 4.3　オブジェクトの新規作成

オブジェクト指向操作の基本は「何」を「どうする」である．新規作成の操作は，この「何」がまだ存在しない状態である．では，どのような操作がオブジェクト指向操作といえるのだろうか．

たとえば，オブジェクト指向操作の一つの代表として J-STAR* の新規作成を参考にしてみると次のようになっている．デスクトップ上に「未使用文書」というオブジェクトが存在し，これを自分のフォルダ等にコピーしてからコピーした「文書」を「更新する」という操作を行っている．これは，オブジェクト指向操作としての一貫性が追求されている操作体系であることは確かである．最初にコピーするという操作が，一般的な今までの操作と異なり，慣れるまで抵抗を感じる人が多い．

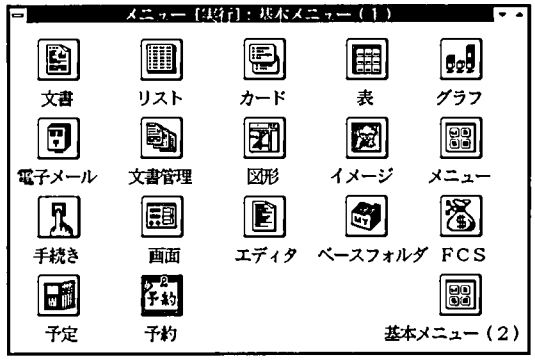
オブジェクト指向プログラミング言語 Smalltalk を参考にしてみると，オブジェクトの新規作成は，オブジェクトクラスに「new」というメッセージを送ることにより行われる．これを操作に直すと，オブジェクトクラス一覧を表示してそこに「新規作成」

* J-STAR は富士ゼロックス(株) のドキュメント作成システムである（登録商標）．

メッセージを送るような操作になる．

EXOS では「新規作成」を 2 通りの操作方法で実行できるようにした．

1)　「基本メニュー」からの起動……文書，イメージ，図形等のオブジェクトクラスがアイコン表示されていてマウスでダブルクリックする（新規作成メッセージを送る）とそれぞれのアプリケーションの新規作成画面となる（図 6）．




この例ではオブジェクトクラス以外のメニューも混在しているが，それらを指示すると単にプログラム起動となる．

**図 6　基本メニューからのオブジェクト新規作成**

Fig. 6　Create new object using a menu

2)　キャビネットからの起動……キャビネットの「基本」プルダウンメニューの「新規作成」を選択すると，オブジェクトクラス一覧ウィンドウが表示される．適当なアイコンをマウスでダブルクリックするとそれぞれのアプリケーションの新規作成画面となる（図 7）．

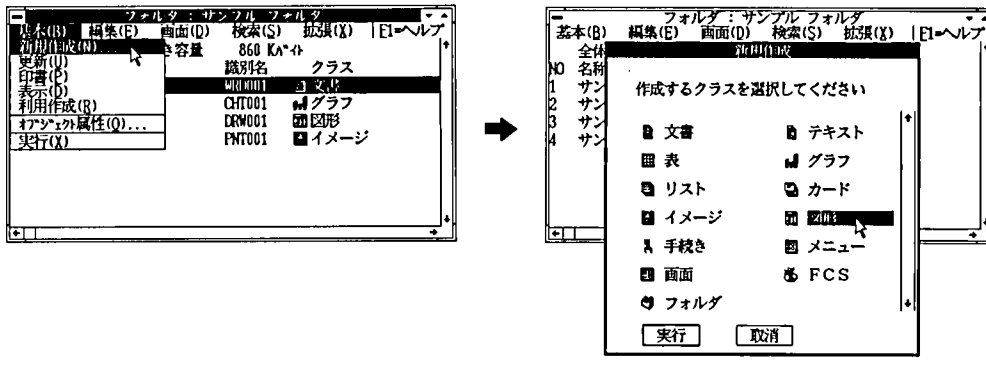



**図 7　キャビネットからのオブジェクト新規作成**

Fig. 7　Create new object using a cabinet

1) 2) の操作の違いは，オブジェクトを「保存」する場所に現れる．操作 1) では利用者ごとに定義されているベースフォルダに保存され，操作 2) では起動したフォルダに保存される（どちらの操作でも保存時にフォルダ変更することができる）．

2) の操作では，先に「新規作成」コマンドを選択してからオブジェクトクラスを選択しているため，オブジェクト/アクション方式に反する操作となる．しかし，実際に使用してみると，「このフォルダ」に作成するということがオブジェクトを新規作成する前に指定でき，ベースフォルダに保存する時以外は，1) に比べて使いやすい．

## 5. 評　　価

EXOSの全般的使用感はどうであろうか．簡単に使えるという感じがする．使ってみて好奇心が刺激されておもしろいと思う．このような点がエンドユーザ向け対話型システムとして重要な点である．

また，EXOSでは本稿2章で述べた3項目の阻害要因は，次のようにクリアされている．

1) PMとEXOSの最低限の操作を覚えれば，アプリケーションのマニュアルはほとんど読まなくても一通りは使用できる．
2) メニューバーで操作ができる．アプリケーションごとのキーマットは過去の遺物となった．
3) 原則としてすべての操作は復元可能で，復元できない操作に対しては必ず確認メッセージが表示される．初心者でも安心して使える．

## 6. おわりに

オブジェクト指向操作は本当に使いやすいのか．たとえば，「ガイドライン・メッセージが表示されないのは初心者には不親切」という意見がある．オブジェクト指向といえども，最低限のレベルに達していることが前提である．マウスによるウィンドウやプルダウン・メニューの操作，マウスドラッグによる範囲選択操作，オブジェクト/アクション方式の操作等が最低限の操作である．これらの基本操作はMS OS/2に限ることなく，WINDOWS* でもほとんど同一である．また，マッキントッシュ等まったく別のウィンドウ・システムの操作とも類似点が多く，一つのGUIの操作を知っていると異なる他のGUIの操作にもすぐに慣れることができる．そして，これらの「オブジェクト指向操作」がこれからのワークステーションの標準操作となりつつある．

最後に，開発を振り返って，設計作法を作るのは困難な作業であった．憲法については一般論であり，とくに反論もなくまとまったが，法律については具体的操作法を規定するものであり，さまざまなアプリケーションの都合や伝統的操作法との相違があり，何故その方法が最適であるかを証明することはできない．ここで憲法は重要な役割を果たした．しかし憲法も万能ではない．設計作法は，オブジェクト指向操作の理論のみを追求した机上の空論であってはならない．実際のアプリケーションでの実現可能性と使用感を確認して設計作法にフィードバックしていくことが大切なことである．

本稿が，他の対話型システムの開発に役立てば幸いである．

---

参考文献 [1] B. Schneiderman,「ユーザー・インタフェースの設計」，日経マグロウヒル，1987.
[2] 坂村健,「BTRON MMI統一プロジェクト」，第2回TRONプロジェクト・シンポジウム資料，Mar. 1987.
[3] 日経バイト,「押し寄せるオブジェクト指向の波」，日経BP，No. 59，June 1989.
[4] 守屋,「ユーザインタフェース技法」，情報処理，Vol. 29 No. 10，Oct. 1988.

* WINDOWSは米国マイクロソフト社の登録商標である．

［5］ 日経コンピュータ別冊，「マルチメディア時代のユーザー・インタフェース」，日経 BP，July 1989.
［6］ Apple, Inside Macintosh, Addison Wesley, 1988.

**執筆者紹介** 河 合 昭 男（Akio Kawai）

昭和 22 年生．45 年大阪大学理学部数学科卒業，46 年日本ユニシス（株）入社．EXEC 保守，シリーズ 1100 性能評価，無人化システム開発を経て 59 年よりマイクロ部門に移り，DS7 イメージ処理，EXOS 開発を担当．情報処理学会会員．

# GUI環境とアプリケーション開発 ——統合運用システム(IOF)におけるGUI適用を背景として

# The GUI Environment and Its Application Development ——The Incorporation of GUI Functions into the Integrated Operating Facility (IOF) System——

佐野浩之

要約 近年，ワークステーションやパーソナルコンピュータのような小型のコンピュータの性能向上を背景に，グラフィカル・ユーザ・インタフェース（GUI）を提供した環境が一般的になってきた．

統合運用システム（IOF）においてもそのユーザ・インタフェース部分にGUIを適用したパーソナルコンピュータを採用することとなり，そのためのプログラムを開発してきた．この中で「集中コンソール」という従来型のユーザ・インタフェースをGUI化して組み込む作業を行った．

本稿ではMS OS/2*のプレゼンテーションマネージャ(PM)を基にGUIの概略を紹介し，集中コンソール機能の開発を通して，GUI環境のアプリケーション開発のうちでもとくに既存機能のGUI適用を行う場合に考慮すべき事項について論じる．

**Abstract** On the strength of recent performance improvement in such small-size computers as workstations and personal computers, the computer environment where graphical user interfaces (GUI) are supported has been more and more popular.

Based on the project to make available the Integrated Operating Facility (IOF) system whose user interface provides GUI functions through a newly allowed connection of personal computers, the author has developed some programs to meet the requirements. His efforts have involved both making operable in the GUI environment formerly developed functions supporting a conventional type of user interface called 'interconnect consoles,'and incorporating them into the IOF system.

Besides briefly presenting the GUI functions offered by the Japanese-language MS OS/2 Presentation Manager, this paper discusses, based on the author's experience in the development of the interconnect consoles functions, some important points worth considering especially when existing applications have to be enhanced for new GUI availability, which is one of the application development efforts for the GUI environment.

## 1. はじめに

近年，ワークステーションあるいはパーソナルコンピュータと呼ばれているような小型のコンピュータのハードウェア，ソフトウェア両面における性能の向上がめざましい．とくにその処理速度，ディスプレイへの表示能力の向上は顕著である．このような背景からこれらの小型コンピュータのユーザ・インタフェースにグラフィカル・ユーザ・インタフェース(Graphical User Interface，以下GUIと記す)を用いること

*MS OS/2，PMは米国マイクロソフト社の登録商標である．

が主流となり始めている．

UNISYSシリーズ2200・1100システムで稼働する統合運用システム（Integrated Operating Facility，以下IOFと記す）では，その情報表示や操作を行う端末にMS OS/2を搭載したPW²を用い，その環境下でGUIを用いたマンマシン・インタフェースを実現している．

本稿でははじめにGUIの概念を，MS OS/2プレゼンテーションマネージャ(PM)を基に解説する．次にIOFにおけるGUIの適用例を紹介し，その適用作業において得られたマンマシン・インタフェースに対してGUIを適用する場合に考慮すべき事項等を記述する．

## 2. グラフィカル・ユーザ・インタフェース

### 2.1 グラフィカル・ユーザ・インタフエース

現在主流となっているGUI環境には，UNIX*系のX-Windows**，NeWS**，SunView**等，パーソナルコンピュータ系のMacintosh***，MS-Windows***，およびプレゼンテーションマネージャPM等を挙げることができる．

近年，GUI環境をマルチタスク環境でのユーザ・インタフェースとするシステムが増えてきている．並行稼働しているタスクごとにウィンドウと呼ばれる矩形を割り当て，それぞれのタスクの稼働状態をディスプレイ上で一望できるようにしている．各ウィンドウはディスプレイの任意の位置に重なり合うように表示されるが，ユーザはそれらをディスプレイ上を稼働させたり大きさを変えたりすることが自由にできる．また，稼働していることがわかればよい場合等は，アイコンと呼ばれる，そのタスクをシンボライズする図形からなる小さな矩形に変えてディスプレイの片隅に置くことも可能である．

そしてこれらウィンドウに対する操作はすべてマウスというポインティング・デバイスを用いて行うことができる．

さらに，データやコマンドの入力のために通常のウィンドウの他に，メニュー・バーやポップアップ・ウィンドウが用いられキーボードからの入力を受け取るが，この時にもボタン，リストボックスといったマウス操作を考慮した図形が用いられる（図1）．

GUIのもう一つの側面としてWYSIWYG（What You See Is What You'll Get）と呼ばれる手法があり，ウィンドウ内にはビットマップ・イメージやベクトル指定による図形の表示，マルチフォントを利用した多種多様な文字表示が行われ，グラフィックの印刷に対応したプリンタを用いた場合，ウィンドウに表示されたままのプリントアウトを行うことも可能となっている．さらに将来的にはビデオ画像や音声等にも対応したマルチメディア表現も期待されている．

### 2.2 GUIプラットホーム

前節で挙げたGUIのための道具立て，すなわちウィンドウの表示，メニューやボタン等のシンボル(図2)，マウスやキーボードからの入力の受け付けは操作環境として

*UNIXオペレーティングシステムは，UNIX System Laboratories, Inc.が開発し，ライセンスしている．
**X-WindowsはM. I. T., NeWSはソニー(株)，SunViewはサンマイクロシステム社の各々登録商標である．
***Macintoshはアップルコンピュータ社，MS-Windowsはマイクロソフト社の各々登録商標である．

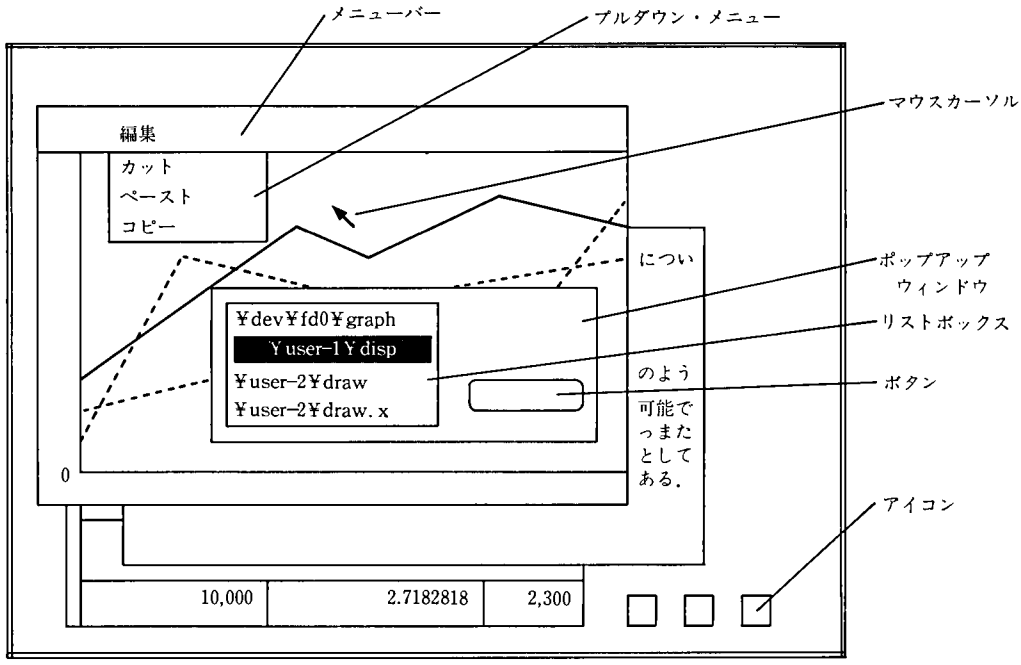



**図1　マルチウィンドウの概念**

Fig.1　Abstraction of multi windows

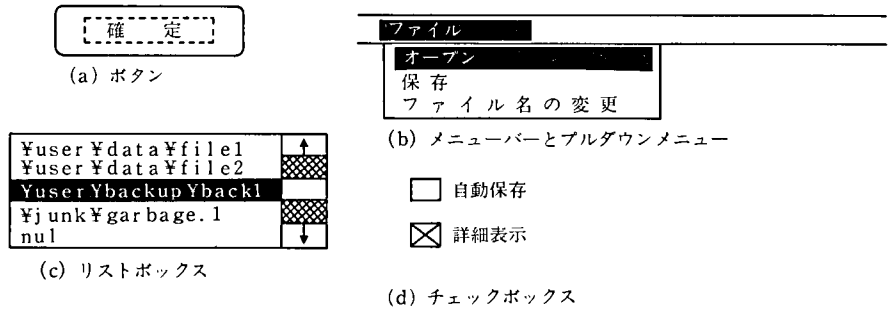



**図2　典型的なGUIに用いられる図形の例**

Fig.2　Examples of symbolic figures for GUI

提供される．そしてアプリケーション・プログラムに対してはシステムコールとしてこれらへのアクセス方法が提供されている．このようにオペレーティングシステムのサブシステムとして組み込まれたGUI機能は，アプリケーションの入出力に関するプラットホームと見ることができる．

本節ではこの意義について述べる．

アプリケーションでグラフィックを用いた入出力を行おうとした場合，表示される図形のデザインからプログラミングによるインプリメントまですべてアプリケーション開発者が行う必要があった．そして複数のアプリケーションで類似の機能を持った図形を用いる場合であっても，各々のアプリケーションごとに改めて作成し直す傾向があった．さらに新たな機能をインプリメントする場合，それに対するデバッグ作業

も付随して発生する．

このように従来のアプリケーションの開発では，そのアプリケーションの本質的な処理部分に加えて，グラフィカル表示とそれらに対するユーザのアクセスを処理する部分のインプリメントとデバッグが追加されるために膨大なマンパワーが必要となっていた．しかし，基本的なグラフィック図形に関する処理を環境側が提供することになると，適切な引き数を用いたシステムコールを行うだけでよく，またデバッグ作業の負担も環境側に吸収されることになるため，アプリケーション開発者の負担が軽減されることになる．

次に，従来はディスプレイに表示されるさまざまな図形や文言に対する意味付けはアプリケーションが独自に定義をしていた．たとえばアプリケーションのサブ機能を呼び出すためにメニューを提供している場合，そのメニュー画面の呼び出し方にも特定のキーを押下するものもあれば，特定のコマンド文字列を入力するものもあるといった状況であった．また，そのメニューから選択を行う方式も項目番号を入力するもの，矢印キーを操作するものとさまざまである．このような状況は，とくにマルチタスク環境下で複数のアプリケーションを同時に操作しているユーザにとっては，思考の混乱を招く基である．

これに対して，GUI プラットホームでは多くのアプリケーションが必要とするであろうと考えられる機能を提供するとともに，アプリケーション開発者に対しては提供される機能の利用形態のガイドラインが提示されている場合が多い．このガイドラインに従って，アプリケーションが同一の方式でそれらの機能を利用すれば，必然的にアプリケーションを越えた共通の操作方法がエンドユーザに対して提供されることになる．ユーザにとって 2～3 のアプリケーションの操作方法を修得することで，他のアプリケーションの操作方法も類推によって短期間のうちに修得することも期待できる．

ただし，アプリケーションのプレゼンテーション機能の多くがアプリケーション間で共通化されるために，たとえばボタン形状を複雑な形にしたり，メニュー選択肢の配列を複雑化したり，多くの色を用いるといった，いわゆるアプリケーションの個性というものを出すことは，むずかしいことになる．

しかし，このことはアプリケーションの本質的な機能を洗練するということが他の類似アプリケーションとの差別化を図るということを意味することにもなり，好ましいことであるといえるであろう．

### 2.3 MS OS/2 とプレゼンテーション・マネージャ

IOF ではそのユーザ・インタフェース・ターミナルとして UNISYS PW[2]を採用している．そのオペレーティングシステムは MS OS/2 が採用され，IOF ユーザ・インタフェース機能（IOF/WS と名付けられており，以下そのように記す）はこのアプリケーションとして開発された．本節では，MS OS/2 およびその GUI プラットホームである PM について触れる．

OS/2 は米インテル社の MPU（Multiple Processing Unit）i80286*以降を CPU として搭載したパーソナルコンピュータのために米 IBM 社と米マイクロソフト社が開

*i 80286 は米国インテル社の登録商標である．

発したシングルユーザ・マルチタスク・オペレーティングシステムである．その特徴として，現在パーソナルコンピュータのデファクト・スタンダードとなっているMS-DOS*のコマンド体系とファイル・フォーマットを踏襲しており，さらにi80286から提供されたプロテクトモードを利用したメモリ管理機能，マルチタスク機能，そして仮想記憶機能を挙げることができる．

OS/2の拡張機能として組み込まれているGUIプラットホームがPMである．PMはその画面表示能力に，MS-DOSのGUI環境であったMS-Windowsの見え方（Look & Feelと言われることもある）を発展させたものとして位置づけられる．しかしMS-Windowsにおける疑似マルチタスク処理や仮想8086機能による，実質的にはリアルモードのタスク・スイッチングとは異なり，MS OS/2のマルチタスク機能を利用したプロテクト・モードのマルチウィンドウ機能を実現している．

PMとそのアプリケーションの関係は，まずコンピュータに接続されたハードウェアのうち，PMは入力機構であるキーボードおよびマウス，出力機構であるディスプレイを管理している．そして入力機構を通じて発生したユーザからの入力動作に起因する事象や，その結果発生する表示上のウィンドウの重なり具合等の変化から発生した事象をすべて「イベント」として認識する．次にPMは，そのイベントが関連するウィンドウ（マウスポインタの下に位置したウィンドウや，別のウィンドウが重なってきたウィンドウ等）を判別し，そのウィンドウに対して発生したイベントを「メッセージ」と呼ばれるデータ構造体に変換して通知する．アプリケーションはこのようなメッセージを受け取り，メッセージのイベント種別ごとの動作を行う．なお，これらのイベントは非同期に発生する．

したがって，PMのアプリケーションにはこのようなイベント・メッセージを受け取るごとに処理していく「イベント駆動型」と呼ばれるプログラミング・パラダイムに則ってプログラミングされることになる．図3にPMとそのアプリケーションの関係

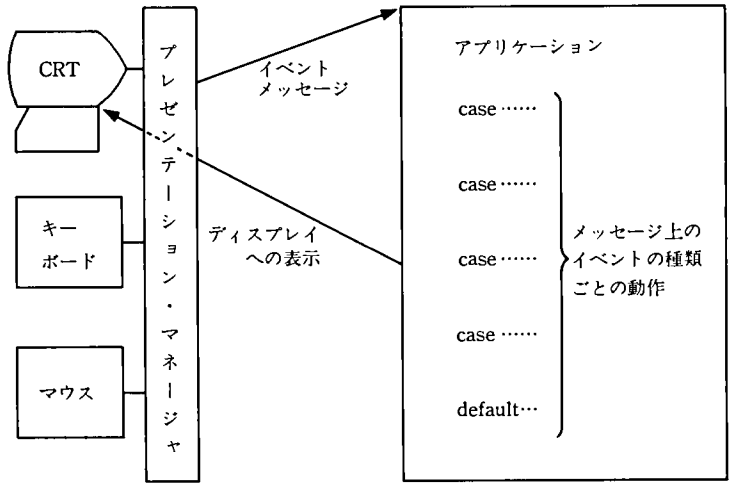



**図3 PMとアプリケーションの関係**

Fig.3 Relationship between PM and an application

*MS-DOSは米国マイクロソフト社の登録商標である．

を示す．図中では単純化しているが，実際にはイベントは PM によって細分化され約200 種におよぶメッセージが想定されている．ただしアプリケーションですべての種類のメッセージを処理する必要はない．

先に挙げた他の GUI プラットホームと同様に，PM もマンマシン・インタフェースのための多様な図形アイテムを提供している．しかし，MS OS/2 がターゲットとするマシンの標準的な表示能力が低いため（解像度 640×400～1120×750 ピクセル程度）それらのデザインは比較的地味であるとの評価もある．

## 3. IOF/WS の集中コンソール機能

### 3.1 IOF/WS の概要

IOF は UNISYS シリーズ 2200・1100 コンピュータの運用を統合して行うためのシステムであり，複数のコンピュータの稼働状況を管理したり，各コンピュータで動作しているプログラム群の運行状況を把握したりすることができる．IOF はいくつかのサブシステムから構成されている．IOF/WS はそれらのサブシステムのうち「集中監視制御」機能のユーザ・インタフェース機能として位置づけられ，IOF の管理する情報をオペレータに示し，あるいは必要に応じてシステムに対する操作を行うための端末として機能する．しかし，マンマシン・インタフェースの方式はホストから指示を受けるのではなく，その機能が IOF/WS 内にプログラムされ，ホストとはデータ本体のみを交換する分散処理形態を採っている．

集中監視制御機能はさらに「ホスト監視」，「運行監視」および「集中コンソール」の各機能から構成されるが，IOF/WS ではそれら各機能それぞれに対応する形で，数種類のウィンドウを用いたユーザ・インタフェース機能をサブシステムとして有している．

IOF/WS の内部構造およびホスト側 IOF との関係は図 4 のように概観することができる．

さて，IOF/WS の各サブシステムはそれぞれ GUI を適用した PM のアプリケーシ

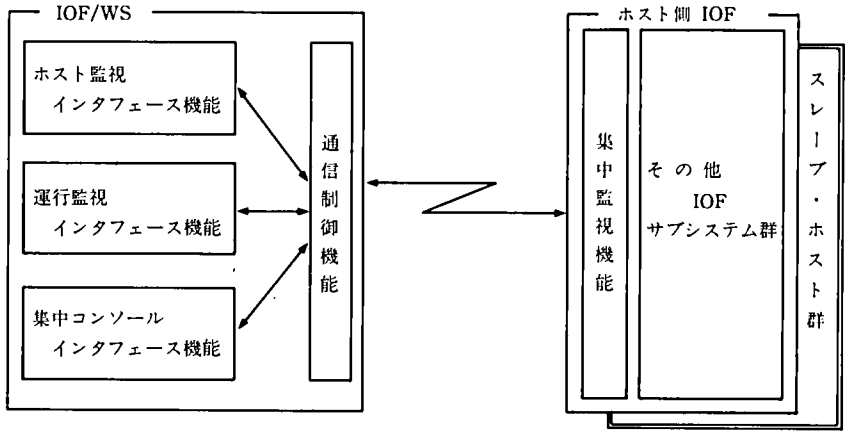



図4　IOF/WS とホスト側 IOF の関係

Fig.4　The relationship between IOF/WS and IOF on 1100 Host System

ョンである．そして「ホスト監視用インタフェース機能」および「運行監視用インタフェース機能」はIOF/WS開発時に新たに企画・開発されたものである．これに対して，「集中コンソール用インタフェース機能」は次節から述べるように，従来は他の方式によって提供されていた機能のユーザ・インタフェース部分をGUI化したものである．このような既存ソフトウェアへのGUI適用作業を通して，新規にGUIアプリケーションを作成する時に考慮すべき事項に加えて，いくつか考慮しなければならない項目があることに気がついた．

そこで本稿では，IOF/WSのうちでもとくにこの「集中コンソール用インタフェース機能」について，これに対するGUIの適用事例を記述していく．

### 3.2 IOFの集中コンソール機能

IOFでは，その集中監視制御機能の一つであるMONITOR/CONSとして集中コンソール機能を実現している．この機能は，従来は「統合オペレーションシステムCONSOLE1100」と呼ばれる，独立したプロダクトとして実現されていたものを下地にしている．

ここではCONSOLE1100について少し述べておく．シリーズ2200・1100コンピュータにはオペレーティングシステムに直結する形でシステムコンソールと呼ばれる端末が接続されている．ここにはオペレーティングシステムが発した，あるいはアプリケーション・プログラムがとくにシステムコンソールに向けて発したメッセージが順次表示されている．これらのメッセージの中には，オペレータによる応答をキーボードから行わねばならないものもある．

さて，単独のコンピュータを運用している状況下では問題にならないが，複数のコンピュータを通信回線を通じて接続し，それらが関連し合いながら運用されている状況においてはシステムコンソールも複数存在することになり，それらを同時に操作する必要性があったとしても容易ではない．CONSOLE1100システムでは接続されているコンピュータの一つを「監視ホスト」と位置づけ，その他のコンピュータではシステムコンソールのメッセージを途中で捕え，これを監視ホストへ回線を通じて送るようにする．監視ホストでは自機のコンソールメッセージや他のコンピュータから送られたメッセージを整理し，これらをUTS50端末に表示する（図5）．

IOFの集中コンソール機能はCONSOLE1100の機能を継承しつつ，UTS50端末をPW²に置き換え，そこに組み込まれたIOF/WSによって，マンマシン・インタフェー

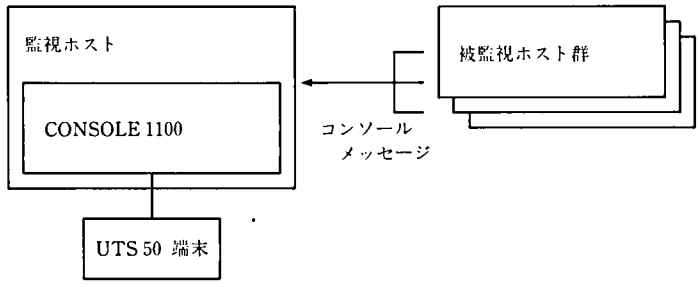



**図5 CONSOLE1100の概略**

Fig.5 CONSOLE1100 System

ス機能を分担すると同時に，そのLook & FeelにGUIを適用している．

### 3.3 集中コンソールへのGUIの適用

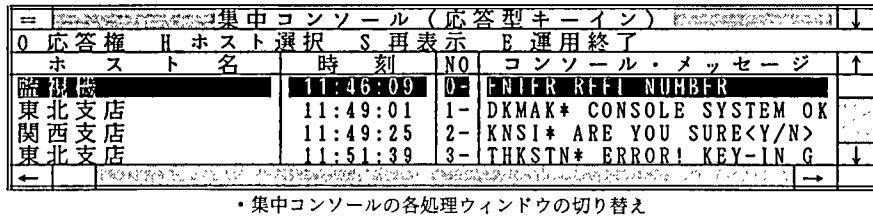
#### 3.3.1 GUIの適用

従来の集中コンソールはUTS50端末を採用していたことからもわかるとおり，キャラクタベースのユーザ・インタフェースを用いていた．これに他のIOF/WSの機能と同様にGUIを適用することとした．集中コンソールにおいては，オーバーラップ・マルチウィンドウの利用がその中心である．

具体的には，CONSOLE1100では一つの画面内に監視中の全ホストのメッセージが混然と表示されていた．そして応答を必要とするメッセージも同一の画面内に表示され，これらはオペレータが応答するまでディスプレイからスクロールアウトしないようになっていたため，未応答メッセージが増えてくると表示可能メッセージが減少する等，視認性があまり良いものではなかった．そこでホストごとに一つのウィンドウを割り当てることとした．これは，ホストごとに一つのウィンドウがあることで複数のホストを操作している実感をオペレータに与えることをねらった結果でもある．また応答待ちのメッセージのために，さらに一つ別タイプのウィンドウを割り当てた．そしてこれらをディスプレイ上でオペレータの見やすい状態に任意に配置できるようにした．

次にIOFの集中コンソール機能で表示するウィンドウの例を示す（図6～8）．

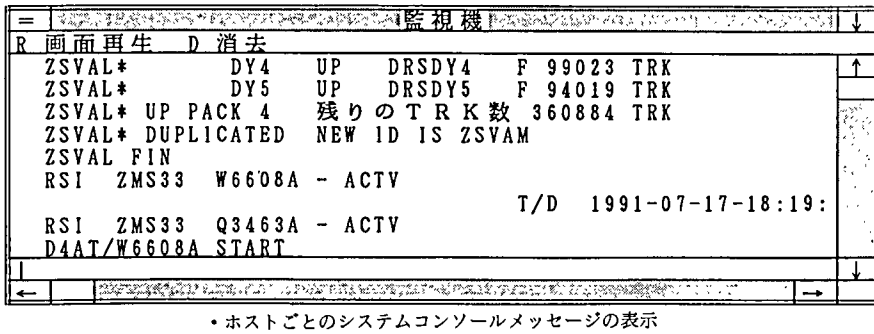
以上のようなウィンドウが図9のようにスクリーン上に配置され，各ウィンドウ間




・集中コンソールの各処理ウィンドウの切り替え
・応答型メッセージの確認，および応答型キーイン
（入力用のウィンドウがポップアップする）

**図6　集中コンソール（応答型キーイン）ウィンドウ**

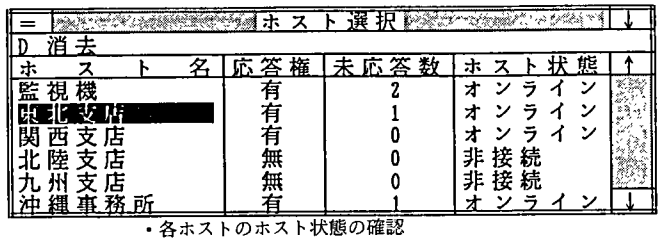
Fig.6　Interconnect console window(with responce key-in function)




・ホストごとのシステムコンソールメッセージの表示
・任意型キーイン

**図7　ホスト表示ウィンドウ**

Fig.7　Each host window




・各ホストのホスト状態の確認
・各ホストのコンソールメッセージの表示制御

**図8　ホスト選択ウィンドウ**

Fig.8　Host selection window

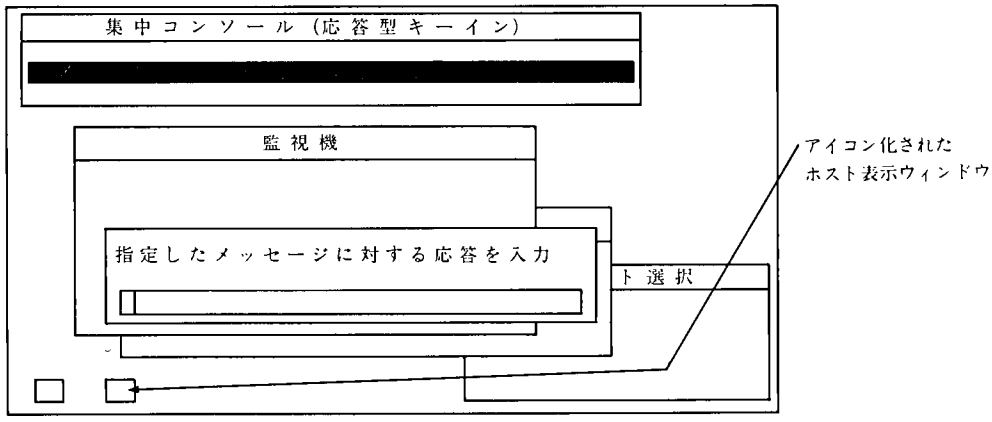



**図9　集中コンソールのウィンドウ配置の概念**

Fig.9　Abstraction of arrangement of windows

の移動は，マウス，およびメニューバー等を用いたキーボードからの指定によって行うことができる．

### 3.3.2　従来の画面および操作方法の踏襲

IOFの一機能としての集中コンソールではあるが，その機能は前に述べた通りCONSOLE1100をもとに設計された．このシステムの最初のユーザは従来のCONSOLE1100を利用したユーザであり，これをIOFに置き換える形で運用することになっていた．このような状況下で集中コンソールにGUIを適用した際の考慮点は次の三点であった．

第1に，UTS50端末のディスプレイで画面を切り替えながら行う操作方式をマルチウィンドウを用いてすべての画面を一覧しながら行う操作方式に置き換えながら，ウィンドウの種類やウィンドウ内のデザインはCONSOLE1100の画面に近いものを採用した．

第2に，操作方法，とくにキーボード操作においてCONSOLE1100の方式や本来のシステム・コンソールと比較し，ウィンドウ切り替え等のためのキータッチ回数をなるべく増やさないことを念頭において設計した．

第3に，CONSOLE1100の表示形態や操作方法は必ずしも使いやすいとは言えなかった面の改善も設計段階では考慮した．たとえば従来のものは，すべてのホストで発

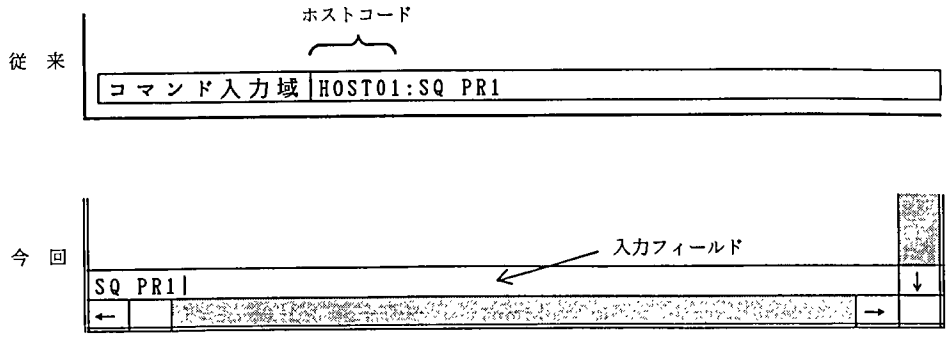



図10　キーイン方法の改善
Fig.10　Upgrade of key-in method

生したメッセージを表示したりコマンドを入力したりするための画面が一つであるため，任意型キーインを行おうとした場合，宛先ホストコードを付加した形でキー入力をしなければならなかった(応答型の場合はレスポンス番号があるため不要)．これをホストごとにウィンドウを割り当てるようにし，そのウィンドウそれぞれに入力フィールドを設け，オペレータは宛先ホストのウィンドウを選んでその中でキーインを行えば良いようにした．すなわち操作上でホストコードについてオペレータは意識する必要はなくなった．またこの操作は各ホストのシステムコンソールへ足を運ぶことのシミュレートにもなっており，より直感的な操作を提供することができたと考えている（図10)．

### 3.3.3　GUIのガイドラインやデフォルト設定に対する考慮

GUIをアプリケーションに適用する上で考慮しなければならないことがある．それは集中コンソールの移行のように，既存のキャラクタベース・アプリケーションに対してバージョンアップ等の機会にGUIを適用するような場合にとくに問題となるのである．

前章で述べたとおり，GUIプラットホームではGUIのための図形アイテムの意味付けやキーアサインについて，該プラットホーム上の操作方式を統一する目的でシステム側からその使用方法に一定のガイドラインが提示されている．一方，既存ユーザは従来方式の操作方法に慣れ親しんでいるため，とくにキーアサインが変更になることを好まない．

一例を挙げると，PM環境下では原則としてファンクションキーF1をヘルプ機能呼び出しに割り当てるようプログラミング・ガイドラインでは規定している．これに対してCONSOLE1100ではF1キーをメニュー画面への切り替えに割り当てていた．このような違いは，IOF/WSの集中コンソールではメニュー・ウィンドウ（ホスト選択ウィンドウ）への移動はマウスまたはPM側で提供しているアクティブ・ウィンドウ切り替えのキー操作（Altキーと Escキーの同時押下）を用いること，およびIOF/WSがヘルプ機能を未提供であることからF1キーにも何も機能を割り当てないこととした．

このようなGUIプラットホームが規定するプログラミング・ガイドラインはあくま

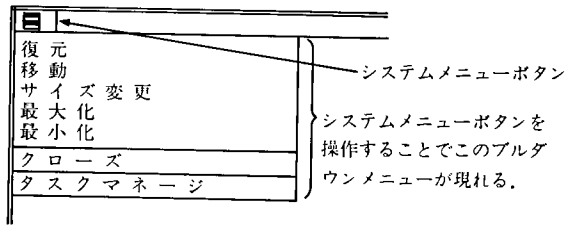



**図11 システムメニュー**
Fig.11 System menu

で原則であるため，厳密に従うか否かはプログラム開発者の判断に委ねられている．しかしGUI環境下においては，独自の操作方法を規定するよりも可能な限りこのガイドラインに従う方が好ましいと筆者は考えている．

また，PM上のウィンドウには図11のような「システムメニュー」と呼ばれる特別なメニューを組み込むことが可能であり（原則として組み込むようガイドラインにも示されている），この中の選択肢には同図に示したようなものがデフォルトで設定されている．しかし一定の手順によって，この内容はプログラム内で変更や追加・削除等が可能である．このようなデフォルトで用意されている選択肢を変更するか否かについても，ユーザが当該のGUI環境に慣れ親しんでくるに従いアプリケーションごとに異なっていると混乱を招くことになりやすいと予想される．やはりアプリケーションの差異を越えた操作性の統一を図る上でも変更は好ましくないと筆者は考えているのだが，システムメニューに限って述べると，どうしても必要であればデフォルトの選択肢群を削除するのではなく，選択肢としては残しつつ選択不可にする（薄色で表示され選択しようとするアラームが鳴るようにPMによって処理される），あるいは仕切線を用いて独自の選択肢をデフォルトの下に追加する程度の変更は許されるのかもしれないとも考えている．

## 4. GUI適用における将来性

現在のGUI環境は，環境としてようやく定着してきたといえる状況である．そして実用に耐えるアプリケーションはまだまだ数も少ない．しかし，ハードウェアメーカやGUIのベンダによるPRの結果，エンドユーザもGUI環境というものを見聞きする機会も多く，利用しようという意欲を持っていると感じられる．そして，ハードウェア/ソフトウェア両面の技術進歩によって，今後より使いやすく，より高機能で，よりレスポンスの高いGUIが登場してくることも十分考えられる．

このような状況の中でアプリケーション開発者としては，GUI適用を推進していくことは決して無駄にはならないであろう．

一方，いずれのGUI環境においてもアプリケーションの開発にはこれまでのプログラミング知識に加えて新しいプログラミング・テクニックを修得しなければならない．また，数百に及ぶファンクション・コールについてもバリエーション程度は最低限身につけなければならない．そのため現状ではGUIアプリケーション・プログラマの不足は否めない．しかし，この問題に対していくつかの解決策がすでに提示され始めて

いる．

まず第1にアプリケーション開発者向けのウィンドウ・ベースのアプリケーション開発ツールがある．これはツールが表示するウィンドウの中で，アプリケーションのウィンドウを設計し，その動作を規定していく．そして最終的にはC言語等のプログラムのソースコードを生成するもので，生成されたソースコードに，ウィンドウでの操作では規定しきれない細かい部分の修正を加えた上でコンパイル，リンクしてアプリケーションを開発するという手法である．この形態はUNIX系のGUI環境でよく用いられている手法で，OpenLook等が代表的である．

第2に，エンドユーザ自身がアプリケーションを生成していく手法で，ユーザに対してそれだけでも動作する各種のウィンドウの雛型を提供し，それにカスタマイズの余地を与えるというものである．この形態はパーソナル・コンピュータ系に多く，HyperCard, ToolBook等を挙げることができる．

このように，GUI環境ではとくにこのようなCASEツールと呼ばれるシステムの模索が盛んであり，将来は「誰でも利用できる」というレベルから「誰でもアプリケーションを作成できる」というレベルまで発展する可能性も秘めていると思われる．

## 5. おわりに

本稿ではGUIを適用したIOF/WSの開発を通して，GUI環境とはどういうものか，そしてGUI環境下のアプリケーション開発に関して考慮しなければならないと思われることについて述べてきた．

GUI環境では多くのアプリケーションで利用されるであろうと考えられるユーザ・インタフェース機能がシステムによって提供されている．このことによってアプリケーション開発者には，開発工数の軽減，短期間化等が期待できる．また，ユーザにとってはアプリケーションの操作方法修得期間の短縮等が期待できる．一方，他のアプリケーションとの差別化を図るためにはアプリケーション本来の機能に対する高性能化が必然的に要求されるが，これは開発者，ユーザの両者にとってむしろ好ましいことであろう．

GUIアプリケーションは，これまでとは異なるアプリケーション・パラダイムに則って作成する必要があり，その修得は必ずしも容易なことではない．しかし，本稿では深入りしなかったが，このプログラミング・パラダイムには最近注目されているオブジェクト指向プログラミングもその素地に考慮されている．したがって開発者はこれらの知識も有している方が有利であろう．

GUI環境はまだまだ新しい環境であり，アプリケーションの数は多くない．現段階では市場に発表されているアプリケーション等を見ても，従来のアプリケーションをGUI環境下へ移植する形態が多いようである．そしてこのような移植作業を行う場合，ユーザ・インタフェースを大幅に変更してしまうことは，従来のユーザに今後ともユーザであってもらうためには必ずしも好ましいことではない．だからと言って，それまでアプリケーション独自で規定していたユーザ・インタフェースを押し通してしまうこともGUIの基本的コンセプトに鑑みると好ましいことではない．このジレンマに対処することは，当面のアプリケーション開発者のテーマといえるであろう．

GUI環境がより良いユーザ・インタフェースの形態を追求することを目的として開発され，このためにいずれのGUIシステムも現在，頻繁にバージョンアップを繰り返している．そのような環境下でアプリケーションを開発するために，開発者にはこれまで以上にアイデアやセンスといったものが要求されるであろう．

---

**参考文献** ［1］ シリーズ2000・1100統合運用システム（IOF）集中監視制御解説書，日本ユニシス（株），0-1刷，1991 2月，pp.4-36～4-49.
［2］ シリーズ2200・1100統合オペレーション・システム（CONSOLE）解説書，日本ユニシス(株)，4-4刷，1990 7月，pp.1～30.
［3］ MS OS/2 1.1A プログラマーズ ガイド，日本電気(株)，Vol. 1, Vol.2, 1990.
［4］「特集 ユーザーインタフェースはグラフィカルであるべきか」，スーパー ASCII，(株)アスキー，Vol. 1 # 5, 1990 12月，pp.49～72.

**執筆者紹介** 佐 野 浩 之（Hiroyuki Sano）
1960年生．1984年東京理科大学理学部応用数学科卒業，1985年日本ユニシス(株)入社．”シリーズ1100コンピュータのアプリケーション開発に従事後，PW²アプリケーションである統合マンマシンインタフェースVenus（仮称）の企画・開発に従事，現在，システム企画開発部分散処理技術課に所属．

# PM環境下でのアプリケーション・プログラム開発事例

## A Case of Application Program Development under the OS/2 Presentation Manager

田 村 太 一

要 約 現在，UNIX*・WS（ワークステーション），PC（パーソナルコンピュータ）の世界では，ユーザ・インタフェースの改善やユーザ・フレンドリな操作環境の実現のために各種のGUI（Graphical User Interfase）が脚光を浴びている．

PCの世界では，MS OS/2**でのPM**(Presentation Manager)やMS-DOS**でのMS Windows**が提供されている．

本稿は，UNISYS PW²エクセレント・オフィス・システム（EXOS***）のコンポーネントとなるスクリーン・デザイナ（EXSD）の開発を通じて経験した事例をもとに，MS OS/2のPM環境下で稼働するアプリケーション・プログラムを開発する上で，開発者が考慮しなければならない点について報告する．

**Abstract** In the world of UNIX workstations and personal computers, various graphical user interfaces (GUI) are in the spotlight these days in an attempt to enhance user interfaces and build a more user-friendly operation environment. For the benefit of personal computers, both OS/2 Presentation Manager and MS-DOS Windows have already been released.

Based on the author's recent experience in developing the EXSD (screen designer) software product, a component of Unisys PW² EXOS (EXcellent Office System), this paper discusses what must be taken into consideration for the development of application programs which run under the OS/2 Presentation Manager.

### 1. はじめに

筆者は，1990年度にUNISYS PW²（MS OS/2）のもとで稼働する「統合オフィス・システム」エクセレント・オフィス・システム（EXOS）のコンポーネントとなるスクリーン・デザイナ（EXSD）の開発を担当した．

EXSDは，MS OS/2 Presentation Manager（以下PMと略記する）のもとで稼働するプログラムである．PMは，使いやすい統一操作環境を提供しており，複数アプリケーション・プログラムの同時画面処理も可能である．PM環境において使用者は，通常ポインティング・デバイス（マウス）を使用して操作を行うため，プログラムに対する指示は非同期に発生する．PM環境下で稼働するアプリケーション・プログラムの構造は，従来われわれが手掛けてきた逐次制御型のプログラムと異なり，イベント駆動型のプログラムである．

GUI（Graphical User Interfase）環境でのアプリケーション・プログラムでは，オブェクト指向，WYSIWYG（What You See Is What You Get）といった観点も重

* UNIX オペレーティングシステムはUNIX System Laboratories, Inc. が開発し，ライセンスしている．
** MS OS/2，PM，MS-DOS，Windowsは，米国マイクロソフト社の登録商標である．
*** EXOS：EXcellent Office System，ユニシス統合オフィスシステム．

要であるが，本稿ではEXSDのプログラム開発を通じて得た経験から，PM環境下で稼働するイベント駆動型のアプリケーション・プログラムを開発する上で，開発者が考慮しなければならない事項について報告する．

## 2. PM制御下でのアプリケーション開発

本章では，PMの概要と従来のプログラム開発との違いについて概説する．

PMは，グラフィックス指向のユーザ・インタフェース，複数アプリケーション・プログラムの同時画面処理といった機能を持ち，しかもハードウェアからの独立性という特徴を備えている．さらにMS OS/2とアプリケーション・プログラムの両者に対して，使いよい統一操作環境を提供している．

統一操作環境を実現するため，MS OS/2には，API(アプリケーション・プログラム・インタフェース）と呼ばれる洗練されたプログラミング・インタフェースが700（PM関連は459）種類以上用意されている．

PMは，アプリケーション・プログラムを開発するためのプラットフォームであり，PMとPMアプリケーション・プログラムの関連は，図1のようになっている．

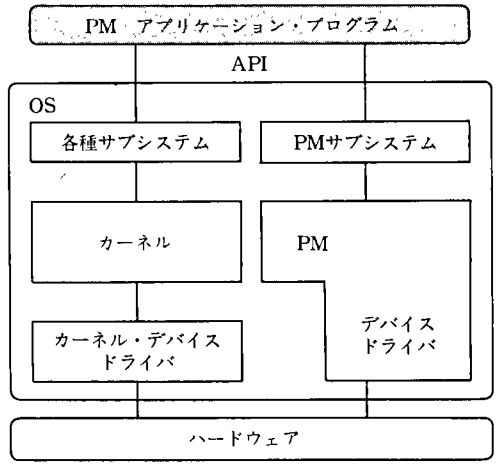



**図1 MS OS/2 PMの概要**

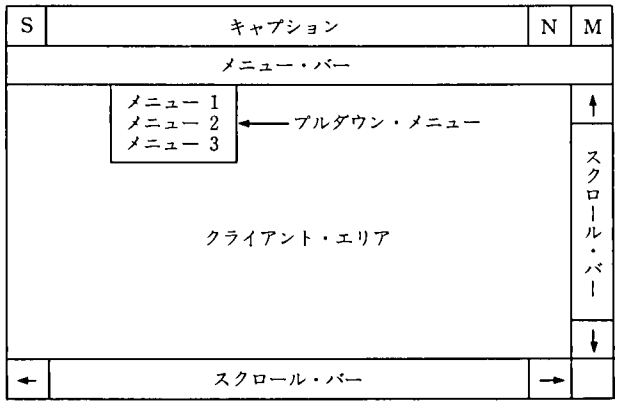
Fig. 1 Overview of MS OS/2 PM

PMが提供している代表的なウィンドウの形式は，図2に示す通りである．

ウィンドウの形式は，パラメータの設定によってシステム・アイコン，ミニマイズ/マキシマイズ・アイコン，スクロールバーの有無等を定義できるので，さまざまな形式を設定することができる．またこの他にもダイアログ・ボックス，リスト・ボックス，メッセージ・ボックスと呼ばれる形式のウィンドウを使用することが可能である．

ウィンドウの各部分の機能は，概略以下に示す通りである．

① メニューバーには，該当アプリケーションの機能ガイドが設定されており，マウスで指示を行うとプルダウンメニューが表示され，実行したい機能をマウスで指示する部分である．使用者は，クライアント・エリアの処理対象を指定した後メニューを選択する．




S：システム・アイコン
N：ミニマイズ・アイコン
M：マキシマイズ・アイコン

**図 2　ウィンドウの形式**

Fig. 2　The window style

② スクロールバーは，画面を上下，左右にスクロールさせる時にマウスで指示する部分である．

③ ミニマイズ/マキシマイズ・アイコンは，画面を最小化（アイコン）したり，ディスプレィ全体に拡大する場合にマウスで指示する部分である．システム・アイコンは，通常プログラムを終了する場合にマウスで指示する部分である．

MS-DOS の経験しか無い人が，MS-DOS ベースでこのような機能を実現しようとすればいかに大変かは，容易に想像ができる．開発を推進する立場の人にとっても，開発を担当する人にとっても PM でのアプリケーション・プログラムの開発は，意識改革を必要とする．

以下に，開発の各フェーズごとに開発者が注意しなければならない問題点と考慮点について記述する．

## 3. デザイン上の問題

本章では，プログラム構造の違うイベント駆動型のプログラムをデザインする上で留意しなければならない点について触れる．

MS-DOS におけるアプリケーション・プログラムは，図 3 に示すように逐次制御型のプログラムであり，MS OS/2 PM 制御下のアプリケーション・プログラムは，図 4 に示すようにイベント（メッセージ）駆動型のプログラムである．

逐次制御型プログラムの処理は，一般的に実現すべき機能と手順（処理手続き）を細かく分割し，その詳細を記述する方法でデザインを進めていく．

そして前述の方法によって記述されるドキュメントの集合がプログラム仕様書となり，このドキュメントによってプログラム開発が行われる．

一方，PM のもとで稼働するアプリケーション・プログラムはイベント駆動型のプログラムであるため，図 4 中の処理 1・・・処理 6 等は，プログラム仕様に記述される機能単位ではなく各イベントごとの処理を記述する必要がある．（イベントは，メッセ

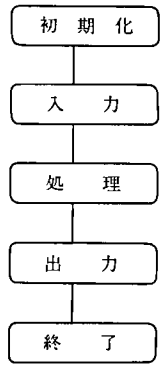



図 3 逐次制御型のプログラム
Fig. 3 Sequential process

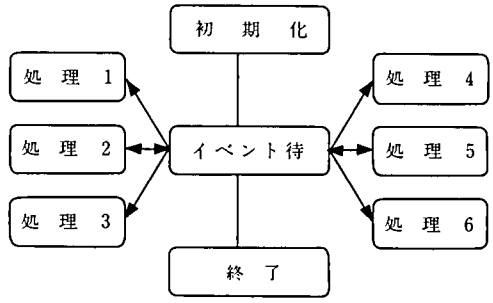



図 4 イベント駆動型のプログラム
Fig. 4 Event driven process

ージによって通知される.) たとえば，マウス・ポインタが移動した，マウスのボタンが押された，キーボードから入力があった，プルダウン・メニューが指示された，といった事象ごとに処理を記述する.

このことは，従来プログラム仕様書と呼ばれているドキュメントによって行われていたプログラム開発は限界があることを示唆するものであり，操作員インタフェースの観点から今まで以上に外部仕様に重点を置いて記述する必要がある．なぜならば，操作員の指示はマウスによって行われ，マウスの移動，マウス・ボタンの押下，といった事象が発生する都度メッセージによってプログラムに通知されるため，どのような操作が行われるのか，また異常な操作はどのようなものかを明確に定義する必要があるからである.

また，一つのイベントが異なる意味を持つ場合や，複数のイベントを処理上一つのイベントに統合する必要がある場合もあるため，イベントによる状態遷移を考慮して各関数をデザインする必要がある．このようなケースにおいては，イベントの類別の必要からユーザ・メッセージを独自に定義して処理を行う方法がとられる.

これらイベント単位の処理は分担する機能を明確にし，そのイベントに関する処理を完結させるようにしなければならない．分担する機能が曖昧なままコード作成を行うとエラーの要因となることを銘記する必要がある.

イベント駆動型のプログラムでは，実行する機能および実行順序は操作員の画面操作をトリガーにして決定される．従来このような形式のプログラムとして，通信回路の制御を司る通信ハンドラと呼ばれるプログラムがよく知られている．通信ハンドラの場合は，回線からのデータの到着や上位プロトコルからの要求をトリガーとして処理が実行される．

PM環境では，自アプリケーション・プログラムのウィンドウ間(通常複数のウィンドウを使用する)，他アプリケーション・プログラムのウィンドウと自アプリケーション・プログラムのウィンドウ間との関連についても考えておく必要がある．したがってデザイン段階で発生するイベント（メッセージ）と，イベント発生による状態遷移を検討する重要な資料として外部仕様書を，今まで以上に重点をおいて記述する必要がある．

初めてPMアプリケーションのデザインを担当する人は，開発キットに含まれているペッツォルド[3]のサンプルには，種々の例が含まれており調査する価値がある．また，発生するメッセージの種類やタイミングを調査する上で，6章に説明しているSPYコマンドは有効なツールである．

## 4. プログラム・デザイン上の問題

前章では，デザイン段階でイベント駆動型についての理解が必要であることを述べた．ここでは，プログラムの構造について例示するとともにウィンドウ・プロシージャを構成する上でのテクニックと，ウィンドウ・プロシージャの重要性について述べる．

プログラムを作成するためには，MS OS/2，PMの基本的事項，MSC(マイクロソフト社のC言語)やAPIを理解していることが前提となる．さらに，PMにおける特徴的な処理であるウィンドウ処理に関しては，種々の操作によって発生するメッセ

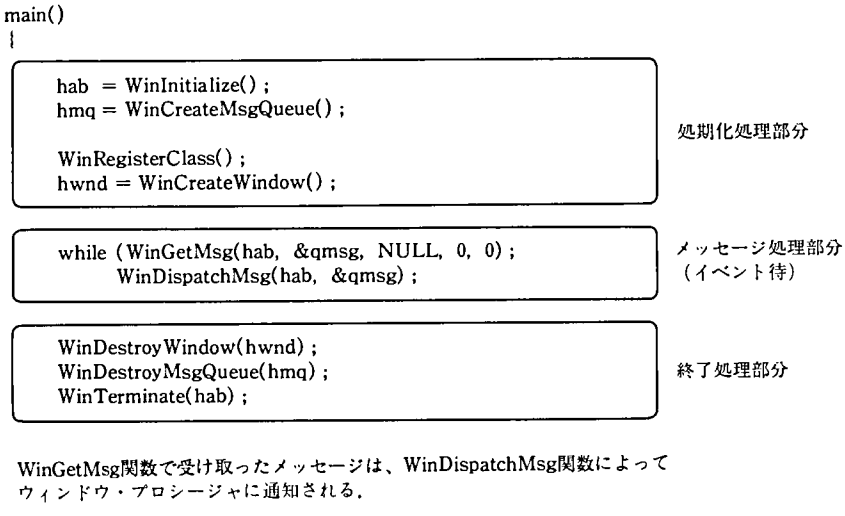



図5　メイン・プロシージャの例

Fig. 5　An example of main procedure

```
MRESULT EXPENTRY ClientWndProc(HWND hwnd, USHORT msg,
                               MPARAM mp1, MPARAM mp2)

    switch(msg) {
    case WM_CREATE:
        クライアント・ウィンドウの生成
        break;
    case WM_PAINT:
        ウィンドウの描画
        break;
    case WM_HSCROLL:
        水平スクロール処理
        break;
    case WM_VSCROLL:
        垂直スクロール処理
        break;
    case WM_CHAR:
        キーボード入力処理
        break;
    case WM_COMMAND:
        メニューバーのメニューが選択された時の処理
        break;
    case WM_BUTTON1DOWN:
        マウスのボタン1が押された時の処理
        break;
    case WM_BUTTON1UP:
        マウスのボタン1がリリースされた時の処理
        break;
    case WM_MOUSEMOVE:
        マウスが移動した時の処理
        break;
    case WM_DESTROY:
        ウィンドウを消去する場合の処理
        break;
    case WM_USER:
        ユーザ・メッセージに対する処理
        break;
    }
    return WinDefWindowProc(hwnd, msg, mp1, mp2);
        デフォルト処理へのリターン
}

WM_xxxxは，メッセージのIDを表す．
PMで規定している処理でよい場合は，WinDefWindowProc関数に
そのままメッセージを引き渡す．
```

**図 6　ウィンドウ・プロシージャの例**

Fig. 6　An example of window procedure

ージの種別，各メッセージに対応した処理についての十分な知識が必要となる．

図4のイベント待の部分で受け取ったメッセージは，図5に示すようにWinGetMsg関数で受け取り，WinDispatchMsg関数が該当ウィンドウのウィンドウ・プロシージャに通知する．ウィンドウ・プロシージャは，図6に示すような構造で，受け取ったメッセージの種類ごとにケース分けして処理を行う．

PMが定義しているメッセージは100種類以上あるが，図5，6の例は，代表的なメッセージを処理するメイン・プロシージャとウィンドウ・プロシージャの一般的なコーディング例である．

図6のルーチンは，メイン・プロシージャでWinDispatchMsg関数を実行すると，PMからコールバック*され各種メッセージを振り分けそのイベントに対応する処理を実行する部分である．

* PMがアプリケーションを呼び出すこと．

図 6 で示した処理には，図 3 の入力，出力といった処理がないことに注目する必要がある．このような処理は通常ユーザ・メッセージに対する処理で，コールバックされる処理として記述する．

プログラム設計上，考慮を必要とするのは次の 4 点である．

① ユーザの操作によって発生するメッセージの整理

② 対応を必要とするメッセージの決定

③ 機能実現のために必要なユーザ・メッセージの設定

④ 特定メッセージを処理する対応ウィンドウ・プロシージャの決定

メッセージは，マウス操作，キーボード操作やウィンドウ上でメニューやスクロールバーを操作した時に発生するとともに，ユーザのアプリケーション・プログラムがユーザの設定した独自のメッセージを発生させることもある．(たとえばメニューバーからプルタウンメニューを選択した場合)

また，そのプログラムが制御しているウィンドウが複数ある場合，ウィンドウごとに前述のようなウィンドウ・プロシージャを設定するが，ウィンドウ・プロシージャ一つだけで処理することも可能である．

図 7 は，複数のウィンドウを一つのウィンドウ・プロシージャで制御する場合の例である．

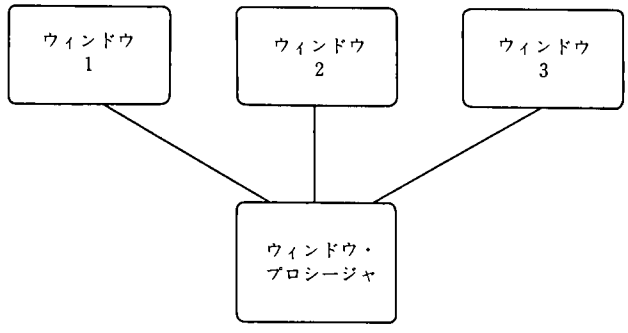



**図 7　複数のウィンドウを一つのウィンドウ・プロシージャで制御する例**

Fig. 7　An example of multi-window control procedure

PM が規定している処理では実現できない機能拡張，すなわちアプリケーション・プログラム独自のインタフェースを実現するためには，サブクラスと呼ばれる手法を使用する．

サブクラスは，PM のメッセージ処理 (ウィンドウ・プロシージャ) にメッセージが渡される前にメッセージをインターセプトして処理を行うようなプロシージャを言い，図 8 に示すような構造である．

EXSD では，業務実行時のキーボード・ハンドリングで独自の処理をするためにキーコードをインターセプトして処理を行っている．また，画面サイズを行桁の整数倍で制御するため，画面サイズ変更に関連するメッセージをインターセプトしてサイズ情報を行桁の整数倍に補正するために，この手法を使用している．

このようにウィンドウ・プロシージャは，PM アプリケーションを構成する上で基幹

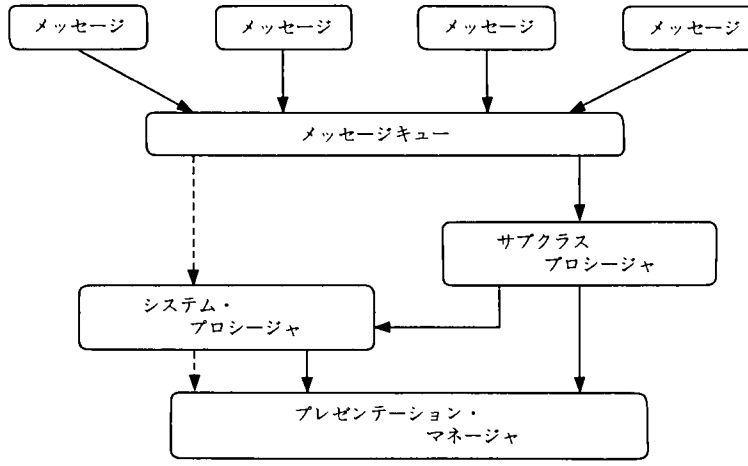



図 8　サブクラスの概念図

Fig. 8　Overview of subclass-procedure

となる部分であり，発生するイベントを処理するために種々のテクニックが使用される．

ここでは，プロダクトの機能を整理してウィンドウ・プロシージャ部分を早期に確定し，ウィンドウ・プロシージャを前提として機能部分を構成するという方法をとらなければ，PM のアプリケーション・プログラムを作成することができないと言うことを強調したい．したがって，複数のメンバーでアプリケーション・プログラムの開発を行う場合には，ウィンドウ・プロシージャの記述を担当する人が，コード開発全体をコーディネートをするのが望ましい．

開発初期に，ウィンドウ・プロシージャの検討を十分に実施せず個々にコーディング作業に入ると，結合テストを実施する段階で混乱状態になると同時に問題点の発見のために多大な時間を浪費することになる．

## 5. コーディング上の問題

本章では，コーディング上の考慮点についてコード例を示して説明し，開発の進め方について述べる．

MS OS/2 には，MSC で用意されている標準関数以外に表 1 に示すように合計 728 種の API と，さらに表 2 に示すような PM 独自の機能が用意されている．これらの関数や PM 独自の機能のすべてについて十分理解した上でコードを作成するのは当然のことであるが，コーディング上とくに次のような点に留意する必要がある．

① イベント（メッセージ）の発生順序や間隔等は，規定されていないのでイベントの発生順序や到着時間に依存したアルゴリズムは，エラーの原因となるため注意が必要である．非同期に発生するメッセージが，どのような順序で発生しても対処できるようにする必要がある．

② ユーザ・メッセージを使用する場合は，メッセージを該当ウィンドウ・プロシージャに通知する API として WinSendMsg 関数と WinPostMsg 関数の二つがあり，いずれもユーザ・メッセージをプロシージャに通知するが，通知さ

表1　API一覧
Table 1　The list of API function group

| 関数種類 | 関数数 | 機能 |
|---|---|---|
| DOS関数 | 158種 | OS関連 |
| DEV関数 | 7種 | デバイス関連 |
| KBD関数 | 18種 | キーボード関連 |
| MOU関数 | 23種 | マウス関連 |
| VIO関数 | 55種 | ディスプレイ関連 |
| WIN関数 | 238種 | ウィンドウ関連* |
| GPI関数 | 221種 | グラフィックス関連* |
| SPL関数 | 8種 | スプーラ関連 |
| 合計 | 728種 | |

*PM環境でのみ使用される関数

表2　PM独自の機能
Table 2　Summery of PM function

| 機能 | 説明 |
|---|---|
| リスト・ボックス | 一覧表形ウィンドウ |
| ダイアログ・ボックス | 対話形ウィンドウ |
| アイコン | ビットイメージによる図形表示 |
| ラジオボタン | 複数のボタンのうち，いずれかがONになるボタン |
| プッシュボタン | 複数のボタンのうち，複数がONになるボタン |
| メッセージ・ボックス | メッセージを表示するウィンドウ |
| ヘルプ・メッセージ | ヘルプ・メッセージを表示するウィンドウ |

れるタイミングが異なる．

・WinSendMsg 関数は，関数実行によって直ちにウィンドウ・プロシージャに通知される．

・WinPostMsg 関数は，関数実行によってメッセージ・キューにキューイングされ FIFO でウィンドウ・プロシージャに通知される．

このような差を認識して，ユーザ・メッセージを WinSendMsg で通知するのか WinPostMsg で通知するのかについて検討が必要である．

図 9 に示すコード・サンプルは，ユーザ・メッセージを使用している例である．

WinPostMsg 関数では，UWMX_COMPLETE というユーザ・メッセージを通知している．このメッセージは処理の終了を通知しており，現在キューにあるメッセージの次に処理をすれば良いため WinPostMsg 関数を使用している．

WinSendMsg 関数では，UWMX_MOVEFOCUS というユーザ・メッセージを通知している．このメッセージは，引き数 i-citem_で示されるウィンドウにフォーカスが移ったことを直ちに通知するために WinSendMsg 関数を使用している．

次に示す図 10 のコード・サンプルは，図 7 で説明した複数のウィンドウを一つのウィンドウ・プロシージャで制御している例である．あるタイミングではウィンドウ 1 の処理を行っているが，ウィンドウ 2 やウィンドウ 3 の処理も実行する．サンプルコ

```
/*------------------------------------------------------------------------*/
/*                    クライアントウィンドウ・プロシージャ                */
/*------------------------------------------------------------------------*/

MRESULT EXPENTRY ExeWndProc(HWND hwnd, USHORT msg, MPARAM mp1, MPARAM mp2)
{
                .
                .
    switch(msg) {
                .
                .

    case WM_BUTTON1UP :
        if( fInStep )
        {
            ButtonPush = FALSE;
            WinSetRectEmpty (hab_, &rclPushItem);

            if( crrstep_[citem_]->item->sel == CLASS_S
                && crrstep_[citem_]->usflag & ITEM_SELECT )
            {
                for (i=citem_, tempGrno = crrstep_[citem_]->item->numgroup;
                     i<citem_num_ && tempGrno==crrstep_[i]->item->numgroup ;
                     i++);
                    { ; }
                if( i >= citem_num_ )
                {
                    fInStep = FALSE ;
                    WinPostMsg (hwnd, UWMX_COMPLETE, 0L, 0L);
                }
                else
                    WinSendMsg (hwnd, UWMX_MOVEFOCUS, MPFROMSHORT(i-citem_),0L);
            }
        }
        break;
                .
                .
    default :
        break;

    }
    return WinDefWindowProc(hwnd ,msg ,mp1 ,mp2);
}
```

図 9　クライアントウィンドウ・プロシージャの例

Fig. 9　An example of client window procedure

ードは，WM_SETFOCUS メッセージを受け取った時の処理を示している．PCE → fState は，制御が移った時のカレント・ウィンドウにフォーカスがあるのか，ないのかを示すフラッグである．

この例では，ウィンドウごとに定義された構造体に情報を持っているが，フラッグを静的変数で定義した場合は，どのウィンドウの情報かが判別できないため動作が保証できない．この用法は，コードレビューにおいてもウィンドウとウィンドウ・プロシージャを一対一と想定してレビューするために発見が困難である．複数のウィンドウのメッセージを処理するウィンドウ・プロシージャをコーディングする場合は，静的変数を使用してはならない．静的変数を使用すると，テスト段階で問題点を発見するのに多大な時間を必要とする．

このような用法は，コード・レビューの段階で除去することが大切である．

図 11 および図 12 の例は，サブクラスを行っている例である．

SubclassCEF 関数では，WinSubClassWindow 関数で SubCefProc というサブクラス・プロシージャを登録している．この登録を行うことによって発生したイベント（メッセージ）が，システムのデフォルト・プロシージャに渡る前に SubCefProc 関数に通知される．

SubCefProc 関数では，マウス操作やキーボードからの指示に対して EXSD 独自の

```
/*--------------------------------------------------------------------------*/
/*                  カスタム・ウィンドウ・プロシージャ                      */
/*--------------------------------------------------------------------------*/

MRESULT EXPENTRY CustomEntryProc(HWND hwnd, USHORT msg, MPARAM mp1, MPARAM mp2)
{
    switch (msg)
    {
        case WM_CREATE:
             .
             .
             .
        case WM_SETFOCUS:
        {
            PCUSTOMENTRY pcs = WinQueryWindowPtr (hwnd, 0);
            if (SHORT1FROMMP (mp2))
            {
                pce->fState |= CEFF_FOCUS;
                DrawCursor (hwnd, pce);
                WinInvalidateRect (hwnd, NULL, FALSE);
            } else {
                pce->fState &= ~CEFF_FOCUS;
                WinShowCursor (hwnd, FALSE);
                WinDestroyCursor (hwnd);
                WinInvalidateRect (hwnd, NULL, FALSE);
            }
            break;
        }
             .
             .
             .
    }
    return (WinDefWindowProc (hwnd, msg, mp1, mp2));
}
```

図 10　カスタム・ウィンドウ・プロシージャの例

Fig. 10　An example of custom window procedure

```
/* 入力項目ウインドウのサブクラス  */
BOOL SubclassCEF (HWND hwndCEF)
{
             .
             .
   if ((CefProc = WinSubclassWindow (hwndCEF, SubCefProc)) == NULL)
       return (FALSE);
   else
       return (TRUE);
             .
             .
}
```

図 11　サブクラスの例(1)

Fig. 11　An example of subclass(1)

機能を実現するため WM_BUTTON1DOWN, WM_BUTTON1DBLCLK, WM_CHAR, WM_SETFOCUS といったメッセージをインターセプトしている．この例はサブクラスを行っている例であるが，キーボードからデータを得るためのコード例としても参考になる．case WM_CHAR：以降の部分では入力されたデータが ANK，数字，全角等あらかじめ設定されている条件に合致するかをチェックしており，エラーの場合はブザーを鳴動させている．

われわれは，開発キットに含まれるペッツォルド[3]のサンプルを検討しながら開発を進めた．さらに，ペッツォルドのサンプルを母体にして機能を追加し，プロトタイプを作成して評価しプロダクトに反映するといった方法も採用した．

関数の機能や動作がマニュアルの記述では良く理解できないとき，一般的な方法としてショート・プログラムを作成して確認を行うが，PM ベースのプログラムは，一つの関数の機能を調べるためには事前に実行しなければならない関数があるため，どう

```
/* サブクラス・プロシージャ  */
MRESULT EXPENTRY SubCefProc (HWND hwnd,USHORT msg, MPARAM mp1, MPARAM mp2)
{
   switch (msg)
   {
       case WM_BUTTON1DOWN:
           if (WinQueryWindowUShort (hwnd, QWS_ID) != citem_)
           {
              EntryCopy (ENTRY_GET, WinWindowFromID (hwndClient, citem_),
                                    crrstep_[citem_]->item->strp);
           }
           WinSendMsg (hwndClient_, UWMX_MOVEFOCUS,
                       MPFROMSHORT( WinQueryWindowUShort (hwnd, QWS_ID) - citem_),
                       0L);
           break;
       case WM_BUTTON1DBLCLK:
           if (LBCreate ())
              WinPostMsg (hwnd, UWMX_COMPLETE, 0L, 0L);
           break;
       case WM_CHAR:
           if ((SHORT1FROMMP (mp1) & KC_VIRTUALKEY))
               break;
           if (!(SHORT1FROMMP (mp1) & KC_CHAR))
               break;
           if (CharTypeCheck (SHORT1FROMMP (mp2)) == FALSE)
           {
               WinAlarm (HWND_DESKTOP, WA_ERROR);
               return (TRUE):
           }
           break;
       case WM_SETFOCUS:
       {
           HWND hwndLB;
           if (hwndLB = QueryLB ())
              if (SHORT1FROMMP (mp2))
                  WinPostMsg (hwndLB, WM_BUTTON1DOWN, 0L, 0L);
           break;
       }
   }
    return (*CefProc) (hwnd, msg, mp1, mp2);
}
```

図 12　サブクラスの例(2)

Fig. 12　An example of subclass(2)

してもロング・プログラムになってしまう．このような局面でペッツォルドのサンプルは，ショート・プログラムを作成する上で非常に有益な資料である．

イベント駆動型のプログラム開発においては，4章で述べたようにPMからコールバックされる処理として記述するため，それぞれのモジュールには独立性がある．最初は，ウィンドウ・プロシージャをコーディングする時に空関数として定義し各機能ブロックをビルトインしながら動作を確認していくプロトタイピング方式は，PMベースのアプリケーション・プログラム開発に最も適した方法と考える．したがって，初めてPMベースのアプリケーション・プログラムの開発に携わる人は，PMベースのアプリケーション・プログラムの一般形や，メッセージの制御方法について，開発キットに含まれるペッツォルドのサンプルを十分スタディした後にプロトタイピング方式で開発を行うのが効果的である．

つぎにコードレビューの時期と内容についてふれる．

コードレビューは，一般的にコード作成が完了した時点で実施されるが，プロトタイピング方式で開発を行う場合は，ある程度動作を確認した後で実施することになる．したがって，コードレビューのポイントは，コードの正確性よりも適切なモジュール分割がされているか，保守しやすい（読みやすい）コードになっているかといった点

である．なぜならば，外部仕様で記述されている操作員インタフェースについては，すでにかなりの確認が実施されているからである．

## 6. テスト上の問題

ここでは，テストで使用されるツールについての概要および使用目的とテストケースをリストアップする場合の考慮点について述べる．

逐次制御型プログラムの開発では，動作の確認のために printf 関数（COBOL の DISPLAY 命令に相当）を使用することがあるが，PM 環境では使用できない．これに変わる手段として非常に稚拙な方法であるが，DosBeep 関数を使用しテストポイントごとに音色をかえるという方法が使用できる．しかしながらこの方法では，多数発生するメッセージのために実質的に確認が行えない．

MS OS/2 では，CodeView[4]デバッガー（以下 CVP と略記する）というデバッグ・ツールが準備されている．

CVP は，プログラムの論理上の間違いを見つけ出すための，強力なウィンドウ指向のツールであり，プログラム実行中に分析が可能で，ソースコードの表示，実行しているソース行の表示，動的な変数値の表示等の機能を持っている．MS OS/2 でのプログラム開発は，CVP なくしてはできないといっても過言ではないだろう．

また，メッセージの発生状況を調査するツールとして SPY（特定のウィンドウに渡されるメッセージを表示するツール）があるが，このツールはメッセージの発生タイミング（トリガー）を調査したり，そのメッセージがどのウィンドウ・プロシージャに渡されるのかを知るために有効なツールであり，4 章で述べたようなクリティカルなエラーの究明に非常に役にたつ．この他にも，GPI 関数を使用して罫線等を描画した場合に，SNAP で画面の一部をビットイメージとして切取り PMCAP で拡大して画面上で位置を正確に確認する方法も利用できる．

これらのツールは，使用法を理解するために多少の時間を要するが，問題点の追求のためには強力なツールである．今後開発を担当する人は，開発キットとして標準に提供されるものや先人が作成したツールについて事前に調査し，その開発において有効なツールについては，使用法を理解しておくことが望ましい．

次にプログラムのテストについて述べる．テストは，テストケースをリストアップしテストケース・レビューを行った後に実施する．この手順は，PM 環境で稼働するアプリケーション・プログラムの場合も同様である．しかし PM 環境では，複数のアプリケーション・プログラムがウィンドウを表示しており，他アプリケーション・プログラムへの切り替え動作やテスト対象アプリケーション・プログラムに対する指示は，PM の基本動作すなわちマウスやキーボードを使用した指示によって行われる．テスト対象のアプリケーション・プログラム自身も複数のウィンドウを持つためウィンドウ切り替え動作の確認は，他アプリケーション・プログラムとの関連を含めて行うことが重要である．

テストケースをリストアップする過程で，このテストケースは，アプリケーション・プログラムの機能をチェックするテストケースなのか，PM の動作をチェックするためのものなのか，判断に迷うことがある（PM の機能チェックをしているような錯覚に

陥る)．しかし，このようなテストケースを省略すると，該当アプリケーション・プログラムが単独で稼働しているケースでは，発見できない問題点を見過ごすことになる．

前記の観点で，制御の遷移を考慮して網羅的にテストケースをリストアップしていくと膨大なケース数になることは容易に予想できる．

限られた時間を有効に活用するためには，テスト対象のアプリケーション・プログラム単体でのテストを済ませた後に，他のアプリケーション・プログラムの動作を並行してテストすることが望ましい．

## 7. おわりに

PM環境のプログラム開発に関して，入手できる資料には各関数の機能については記述されているが，ある機能を実現するために必要な関数の種類，言い替えればどういう手順で行えば良いのかということについての記述はほとんどない状況である．

そのため，われわれは開発キットに含まれているペッツォルドのサンプルを母体にプロトタイピング方式で機能ブロックを作成し，動作を理解しながらビルディングブロック方式で開発を行った．

はじめてPM環境で稼働するプログラムの開発を行う人は，同様の経験をすると考えられる．本稿で述べたことは，MS WindowsやUNIX・WSでのGUI環境で稼働するプログラムの開発においても共通する点がある．

本稿が，今後PMを含めGUI環境で稼働するアプリケーション・プログラムの開発を担当する人にとって少しでも役立つことを期待したい．

---

**参考文献**

［1］ M・ペロン，“OS/2の全貌：標準ユーザ・インタフェースとしてOS/2本体と統合”，日経バイト(1987/9)．

［2］ M. S Kogan, F. L. Rawson, “OS/2の設計思想”，日経エレクトロニクス(1989/1/23)．

［3］ C. Petzold, “Programming The OS/2 Presentation Manager”, MicroSoft press.

［4］ MicroSoft社，“MicroSoft C Optimizing Compiler Code View & ユーティリティ”，MicroSoft.

**執筆者紹介** 田 村 太 一 (Taichi Tamura)

1946年生．1969年甲南大学経営学部経営学科卒業．同年日本ユニシス(株)入社．製造関連のフィールドSEサービスに従事，1986年よりマイクロ系のソフトウェア開発を担当．現在，関西支社システム技術二部ワークステーション・システム課に所属．

# グラフィック環境下のターミナルシステム

## Terminal System on Graphic Environment

佐々木　茂

**概　要**　ワークステーション上ではハードウェアの高機能化と高性能化に伴って，GUI (Graphic User Interface)を備えたソフトウェア・プロダクトが整備されつつある．GUIは，ターミナルシステムにも同様に必要とされている．その主目的は，ワークステーション上のデータとホストシステム上のデータとの統合を可能にするためである．本稿は，MS OS/2* のプレゼンテーション・マネージャ（PM）上で走行するUNISYS 2200・1100シリーズ用のターミナル・システムであるインフォメーション・サービスシステムについて，その機能概要と設計のねらいについて記述する．

**Abstract**　With workstation hardware getting higher and greater in its functionality and performance, software products for workstations, which provide graphic user interfaces (GUI), are being lined up. GUI is required for terminal systems as well for the main purpose of making possible the integration of data both on the workstation and on the host system.

This paper describes the general functions and design targets of the information service system, a terminal software product for the Unisys 1100 Series, which operates under the MS OS/2 Presentation Manager (PM).

## 1. はじめに

UTS(Universal Terminal System)シリーズの汎用ターミナルは，1970年にU 100として開発されて以来，UTS 400，UTS 20/40/50，そして先のDS 7ワークステーション上のUTSターミナル・エミュレータへと発展してきた．

この間，マイクロプロセッサの発展に伴い，ハードウェア回路のみで構成されていたUTSの制御機能は，UTS 400，UTS 20/40/50ではマイクロプロセッサ8080**やZ 80**上のファームウェアに置き換えられた．次のDS 7システムでは，CPUにi 80186/i 80286**，オペレーティングシステムにパーソナル・コンピュータ用のCCP/M 86**（あるいはコンカレントDOS**）を搭載した．そしてターミナルシステムは，CCP/M 86オペレーティングシステム上の**一つのアプリケーション・プログラムの形態**で実現した．

このように，ターミナル機能の実現形態は変革を遂げている．それに伴って，図1に示すようにスクリーン・キーボード，OCR，イメージスキャナ等の周辺装置の拡張，グラフ，カナ漢字変換，外字，メニュー等の入力・表示機能，そしてDKT***，U-MicroForm***等のMML（Micro-Mainframe Link）機能を，操作性を含めたターミナルの互換性を保ちながら拡張してきた．

---

* MS OS/2，PMは米マイクロソフト社の登録商標である．

** マイクロプロセッサ8080，i 80186/i 80286は米インテル社の登録商標である．Z 80は米ザイログ社の登録商標である．CCP/M 86，コンカレントDOSは米デジタル・リサーチ社の登録商標である．

*** DKTはMAPPER操作に準じたファイル転送システムである．U-MicroFormは画面分散処理システム(画面様式情報をワークステーションに備え，データのみをホストシステムより取り出し，画面様式と合成し表示するシステム)である．

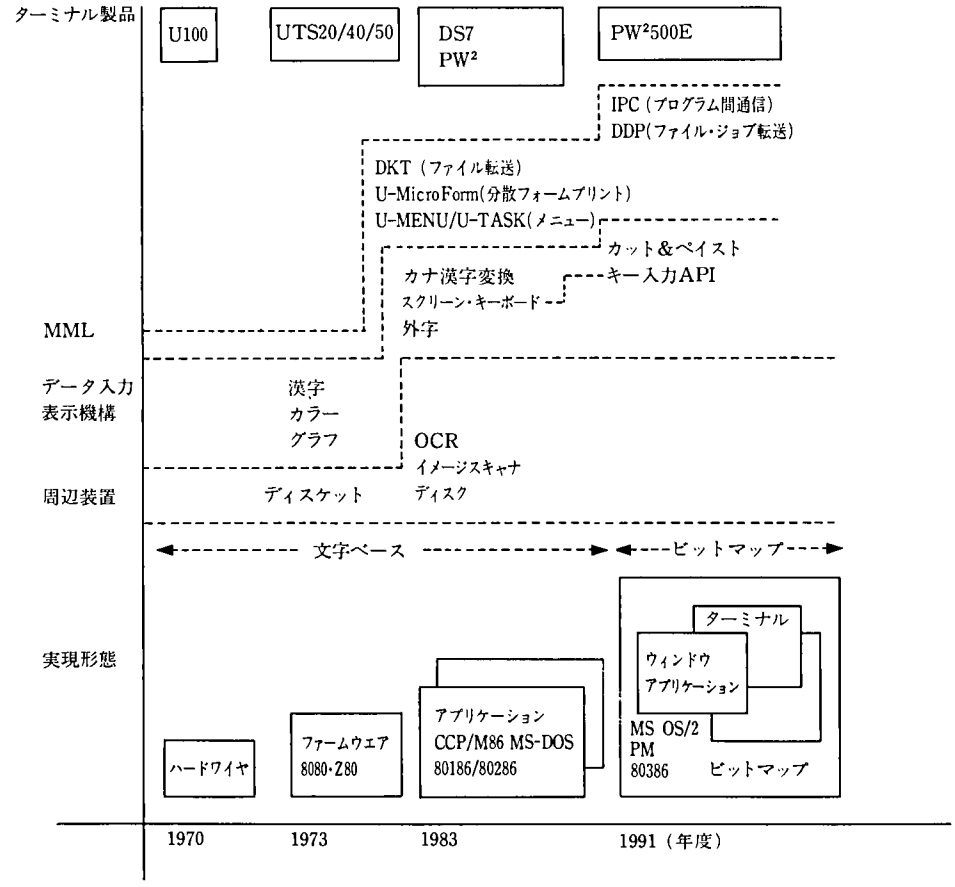



**図 1　ターミナルシステムの変遷と付加機能**

Fig. 1　Terminal system vicissitudes and its addapted features

さらに今日では，ワークステーションのハードウェアは，より高速な CPU(i 80386, 486*)，高解像度のビットマップ・スクリーン，そして大容量のメモリ(8 M から 16 M)を搭載している．また，MMI (Man Machine Interface) の改善を図る一つの方向として，ワークステーション・システム上のビットマップ・スクリーンにウィンドウ表示をする方式が定着しつつある．MS-DOS**および MS OS/2**上の MS-WINDOWS**や PM (Presentation Manager)**は，GUI (Graphic User Interface) を備えるシステムを構築しやすいプログラミング環境を提供している．

これらのビットマップ・スクリーン上への情報展開の利点は次に示す通りである．

1) 情報表現の多様性……サイズの小さい文字を利用し，限られた領域の中に同時に表示できる情報量を多くできる．また逆に大きな文字サイズを利用し重要箇所を強調することができる．

2) 情報の同時表示……マルチ・ウィンドウ表示機能による多角的な情報を同時に表示できる．たとえばデータとグラフの同時表示等である．

---

* i 80386/486 は米インテル社の登録商標である．

** MS-DOS, MS OS/2, MS-WINDOWS, PM は米マイクロソフト社の登録商標である．

3) ダイレクト操作……マウスによる操作は操作する対象（オブジェクト）を直接選択する方法が採用できる．

ワークステーションは既存システムとの互換性を保ちながら，これらのMMI環境の進展に対応することに加え，次のようなニーズに応えることが課題となっている．

通信層では専用線に接続する構成から，公衆網やローカル・ネットワークに接続する構成が増加している．ネットワークの利用の拡大とワークステーションの高性能化は，データ通信インタフェースの仮想化とホストシステム上の処理の一部をワークステーションに分担させる形態へと変化させている（分散処理）．

一方，OA化の進展は，基幹ホストシステムのデータを，ワープロ，表計算ソフトウェアに取り込む場合に，シームレス・オペレーションを容易に構築できる機能を必要としている（OAツールへの統合）．

本稿では，ターミナルシステムのグラフィック化の必要性，利点について述べ，MS OS/2のPM上に構成されたGUIを備えているUNISYS PW²のIS-PC/IS-CS (Information Service-Personal Computer/Information Service-Communication Server) の機能概要と具体的機能を示す．また，GUIをベースにしたターミナルシステムの開発に当たってのニーズ，技術的課題，およびその対策について述べる．

## 2. GUIをベースとするターミナルシステムの要件

ターミナルシステムのGUI化は，ターミナルシステムが利用される環境がビットマップ化されていることに直接的な理由があるが，ターミナルシステムの実現においては，使用形態からくるニーズに対応した機能を備えることが重要な点となる．今日のワークステーションにおけるターミナルシステムのニーズと機能は，図2に示すように整理することができる．

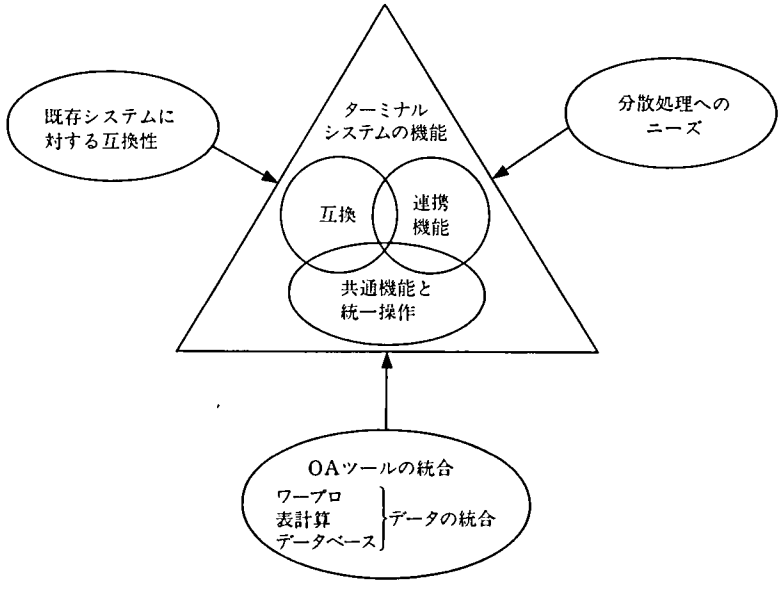



**図2　ワークステーション上のターミナルシステム機能とニーズ**

Fig. 2　Needs for terminal system on workstations and its features

## 2.1 既存システムとの互換性と課題

ターミナルシステムは，ホストシステムに構築された既存のアプリケーション・プログラムに対する互換性を絶対要件とする．また専用ターミナル装置として利用する場合には，プログラムに対する互換性に加え，すでに設置されている既存のターミナルシステムに対し運用面を含めた操作の互換性が求められる．

しかし，周囲の環境が統合化しているなか，既存ターミナルシステムとの同一操作を保持することは，相対的に操作性の低下となる．たとえば，ワープロ等を併用する際，ワークステーション上のシステムとしては，ターミナルシステムはそれとの操作の共通性をなくしてしまう．たとえば，入力フィールドの右桁合わせの文字入力等である．

図3に右桁合わせのフィールドを示す．カーソルが左側のスペースが入力されているフィールドにある場合入力できなかったり，カーソルがフィールドの先頭に位置づけられることである．これらについては，OA化の拡大とともに単に既存のターミナルとの互換性に執着するよりも統合操作に統一すべきと考えられる．

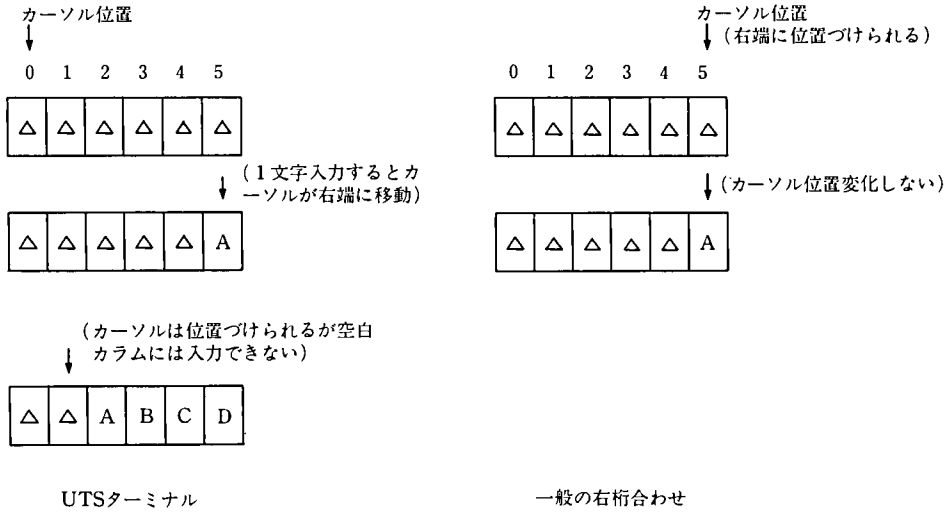



**図 3 右桁合わせのフィールド**

Fig. 3 Right-justified field

一方，ターミナルシステムの互換性からは利用者のニーズに合わせ，選択あるいは変更できることが必要となる．たとえば送信操作である．本来の機能では，CR は改行，XMIT はデータの送信を指示する．OA パッケージ・ソフトウェアにおいては，CR キーを入力の終わりを指示するキーとして利用している．そのため，OA ソフトウェアを主に利用している使用者がターミナルを利用すると，XMIT のかわりに CR キーを押す等の誤操作を行ってしまう．システムによっては，ターミナル機能を利用する際にも CR キーに送信機能を与える方が便利な場合がある．

## 2.2 分散処理へのニーズ（データ表示装置からデータ処理システムへ）

従来のターミナルとホストシステムの関係は，表示装置と処理システムの関係にあった．この関係は，データ処理システムとデータベース・サーバとの関係に移行する

傾向にある．しかし，それはワークステーションのソフトウェアとホストシステム上のソフトウェアのデータ連携の実現を前提とする．その際，ホストシステムをアクセスする意識を必要としないシームレス・オペレーションがワークステーションとホストシステムのアプリケーション間で構築できることが求められる．

具体的な例では，EXOS システム*がある．EXOSでは，IS-PC/IS-CS をベースにしてホスト上のオブジェクトもワークステーション上のオブジェクトと同様に操作できるデータ連携を提供している．図4に示すようにワークステーション上のワープロの使用者は，保管された場所を意識することなくオブジェクトの取り出しや保管ができる．

UTS ターミナルシステムでは，ホストシステムへの電文の送受信に関するプログラミング（ユーザ・オウン・コード）機能を提供してきたが，ネットワーク上のコンピュータシステム間を前提としたアプリケーション間通信機能，そしてプログラミングレスで容易にシステムが構築できる機能が必要となる．

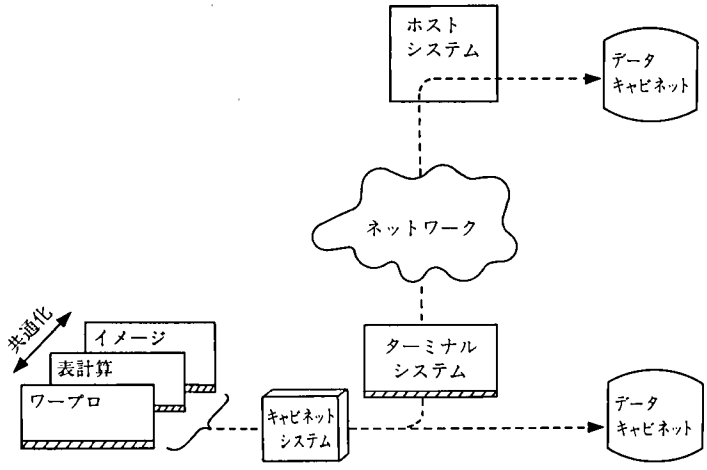



**図 4　ホストシステムとワークステーションの連携**

Fig. 4　Cooperation with workstation and host system

## 2.3　PM（グラフィック）環境と課題

PM の環境下でワードプロセッサ等のアプリケーション・パッケージは，オブジェクト指向を基本概念に採用し，操作性の改善を図っている．さらに，ワークステーション上のプログラム間では，たとえば保管，印刷，行への挿入等の同種の機能を統一した操作で実行できるように「操作の統一化」を図っている（図5）．

また，PM 上では図6に示すように，各アプリケーション・システムは一つのウィンドウとしてオペレータに表示される．PM 上では複数のプロセスが並行作動し，同時にスクリーン上にそれぞれの画面を表示できるマルチタスク/マルチウィンドウ環境を

* EXOS：EXcellent Office System，オフィスにおける「文書作成/保管」「データ処理」「コミュニケーション」「データベース検索」等の処理を支援するための統合オフィスシステム．

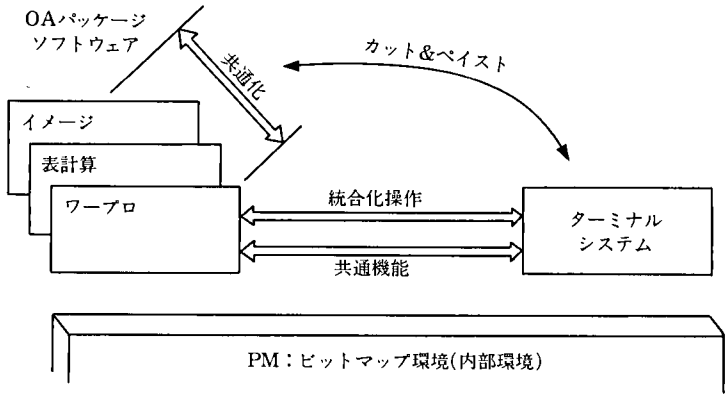



図 5　ワークステーションの操作環境

Fig. 5　Operational environment on workstation

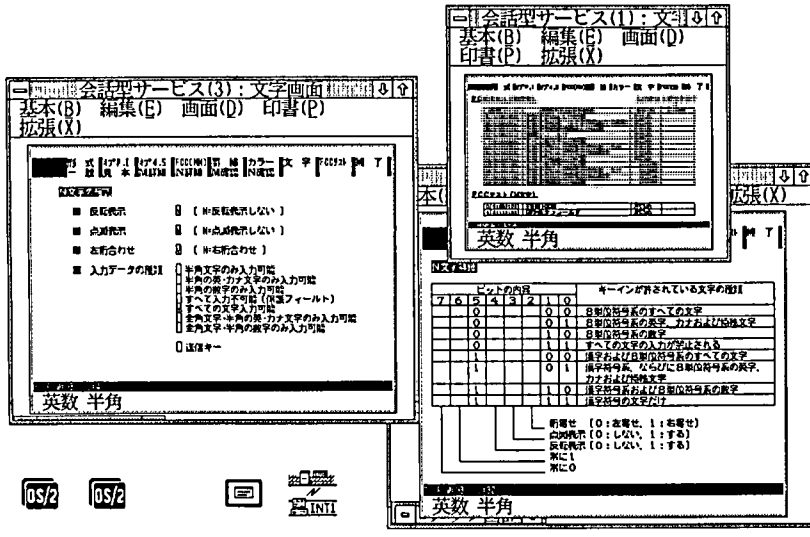



図 6　PM のオーバラップ方式によるマルチウィンドウ表示

Fig. 6　Overlapping multi-window display on PM

提供している．

ターミナルシステムをワークステーション・ソフトウェアと同一環境上に構成した場合，次の視点から共通操作と共通機能が求められる．

ウィンドウの一つに表示されたターミナルシステムは，外見上は他のワークステーション・ソフトウェアと差異がない．したがって，他のパーソナルシステムのソフトウェアが備える統一操作に加え統合化機能が求められる．たとえばカット&ペイスト機能である．共通操作は統一操作を修得する時間を削減し，結果的に正確な操作を促す．

反面，グラフィック環境では文字情報を表示するだけでも数十倍の CPU を要するため，応答性が低下する技術的問題が発生する．これについては 5 章で述べる．

これらのワークステーションの使用環境からワークステーション上のターミナルシ

ステムに求められる機能は，「操作性」，「連携」の二つに集約される．操作性と連携処理は互いに関連するが，操作性については次の2点である．

① ワークステーション上での共通操作・共通機能

② 既存ターミナルシステムとの互換機能と互換操作

また，連携については次の点である．

① ターミナルシステムとワークステーション・ソフトウェア間の連携

② ターミナルシステムを介した（または介しない）ホストアプリケーションとワークステーションの連動システムの構築を支援する機構

## 3. インフォメーション・システムの概要

前章のような機能要件に対応すべく開発したのがインフォメーション・サービス（IS-PC，IS-CS）である．本章では，その機能概要について述べる．

### 3.1 DCA ターミナル

IS-PC/IS-CS は，ユニシスの分散型通信アーキテクチャである DCA (Distributed Communication Architecture) に基づいたターミナルシステムである．図7にその構成概要を示す．通信階層は，リンク層を MS OS/2 のドライバで支援し，L3 と L4 の（トランスポート層）は，DLL (Dynamic Linking Library) で構成している．最上位

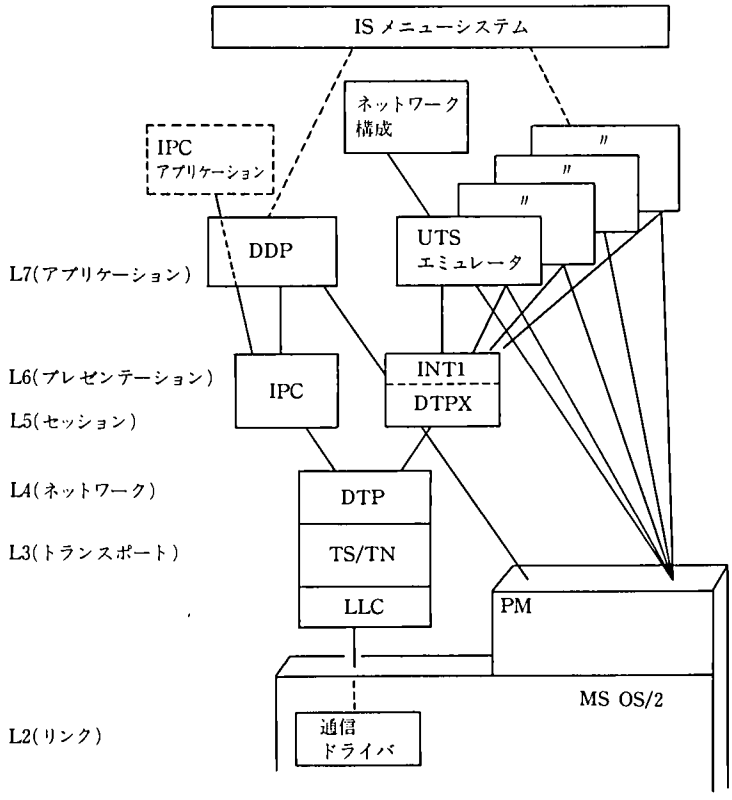



**図 7　IS-PC/IS-CS の構成**

Fig. 7　Configuration of IS-PC/IS-CS

層には，L6(プレゼンテーション層：INT 1 プロトコル) と L7 層 (UTS のスクリーンをエミュレーションするプログラム) からなる．

データ転送メディアとして，UDLC (Universal Data Link Control)，Ethernet* 802.3，パケット網 (80 年度版) そして TCP/IP (Transmission Control Protocol/Internet Protocol) から選択し利用できる．接続形態図を図 8 に示す．IS-PC の構成はスタンドアローン型であり，IS-CS は，ネットワーク層 (L 4) 以下の役割を 1 台の PW²に持たせ，他の PW²はセッション (L 5) 層以上を分担する，クライアント/サーバ構成のシステムである．これは物理回線の共有化，使用メモリの削減に効果がある．クライアントとサーバとなる PW²間は Netware**，あるいは LAN-MGR***のプログラム間通信機能を利用している．

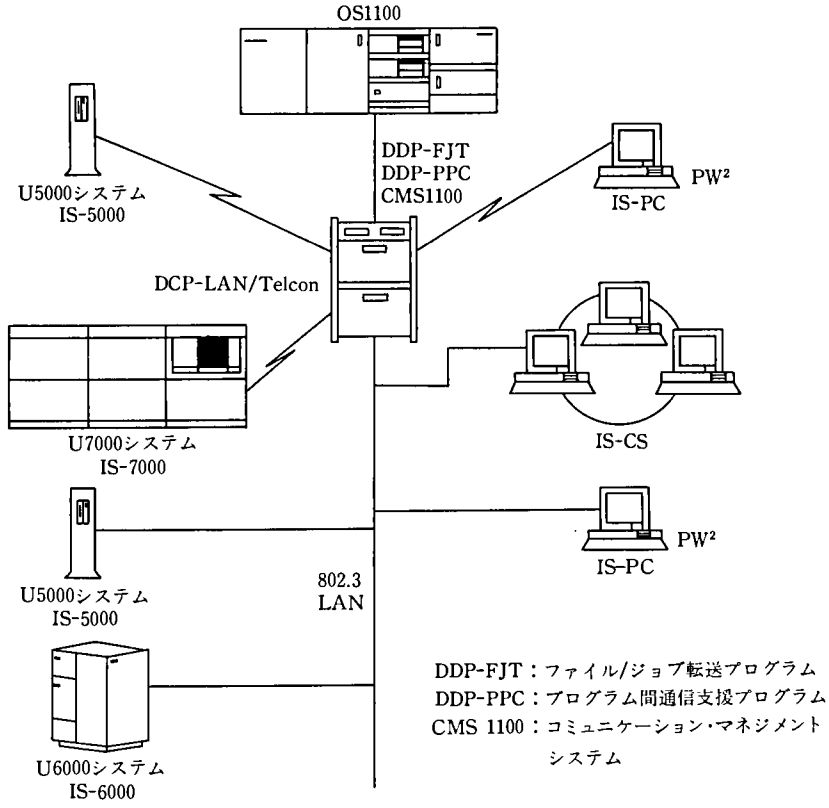



図 8　接続形態図

Fig. 8　Network connection

### 3.2　会話型サービス

UTS 50 ターミナル機能を INT 1 (Interactive) プロトコルの下で実現した機能である．ホストシステムの UNISYS 2200•1100 では，UTS から INT 1 あるいは逆のプロトコル変換を行い，従来の DS 7，UTS 50 を利用し構築されたホストシステム上のア

* Ethernet は米 Xerox 社の登録商標である．
** Netware は Novell 社の登録商標である．
*** LAN-MGR は米マイクロソフト社の登録商標である．

プリケーションと機能の完全互換性を保っている．データ転送は，透過モードの8ビット，そして従来のUTSターミナルと互換性のある7ビットで行うことができる．

### 3.3 プログラム間通信サービス

従来のホスト（主局）とターミナル（従局）でのプログラム間通信機能に加え，DCAのネットワークに接続されたコンピュータシステム間が互いに対等な（主局あるいは従局になれる）関係でデータ通信ができるIPC（Inter Process Communication）を使用できる．

### 3.4 ジョブおよびファイル転送サービス

3.3節で述べたIPC機能を利用した機能の一つである．UNISYS 2200・1100システムへのジョブ転送，IS 5000/6000システム（UNISYS U5000/6000シリーズに搭載されているUNIX*上のインフォメーション・サービスシステム），UNISYS 2200・1100システム，PW²のインフォメーションシステム間でのファイル転送を行うことができる．

## 4. GUIで実現されたターミナルシステム機能の利点

本章では，IS-PC/IS-CSが備えている特徴の一つであるGUIターミナルの機能とその利点について記述する．

### 4.1 既存システムの互換性と機能拡張

#### 4.1.1 プロポーショナル・ウィンドウ

PM上では，複数のサイズの文字を選択し利用することができる．小さい文字サイズの利用は，限られた領域に多くの情報を表示することを可能にする．

既存の多くのアプリケーションシステムは，ターミナルとの情報交換の窓口である表示領域（80字×24行）の全領域を使用している．したがって，ターミナルシステムに，ウィンドウ表示を採用する場合，常に（ウィンドウを縮小した場合でも）ウィンドウ内に80×24の情報が表示されていることが必須となる．

IS-PC/IS-CSでは，ウィンドウサイズに応じた最適文字サイズを自動選択し，常に80×25バイトの情報を表示する（プロポーショナル・ウィンドウ）．図9は四つのIS（IS-PC/IS-CSの会話型システム（UTS））をタイリング表示した例である．先の図6は，オーバラッピング方式による表示例である．

#### 4.1.2 マルチウィンドウによる関連データの表示

MAPPERグラフに代表されるホストからのグラフ表示が，マルチウィンドウ上で可能となる．グラフとそのグラフ作成のソースデータを同時に参照することがマルチウィンドウ表示を利用することにより可能となる（図10参照）．また，それぞれのウィンドウサイズを変えることにより表示レイアウトを使用状態に合わせ，移動・変更し，グラフを参照しながらのデータ変更，最適なグラフ表示選択ができる．また，先の図9に見られるように二つのデータを横に並べて比較できる等の利点がある．

（ワークステーション上のデータ処理とグラフ処理では，データを変更するに従って，グラフが自動的に変更される．また，グラフの種類も比較的容易に指定変更できる．ホストシステムで編集・加工している現在のホストシステムとターミナルシステ

* UNIXオペレーティングシステムはUNIX System Laboratories, Inc.が開発し，ライセンスしている．

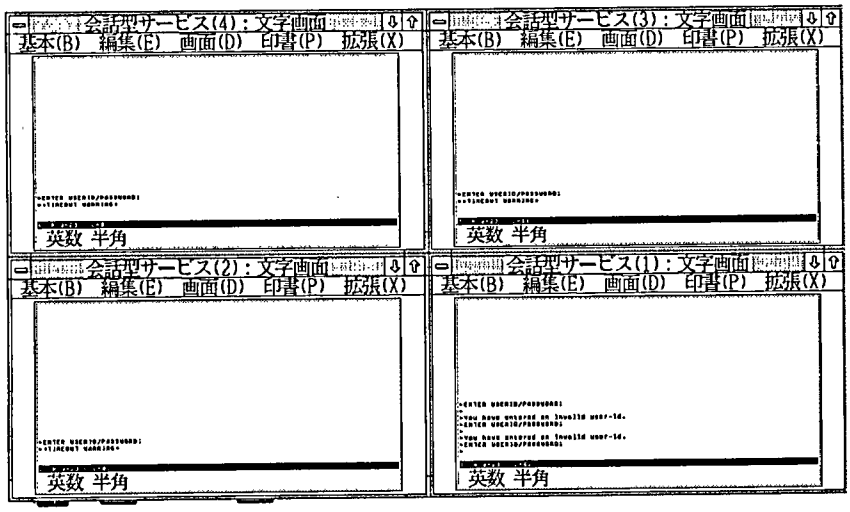



図 9 4セッションを使用中の会話型画面（UTS）

Fig. 9 Interractive screens (UTS) using 4 sessions

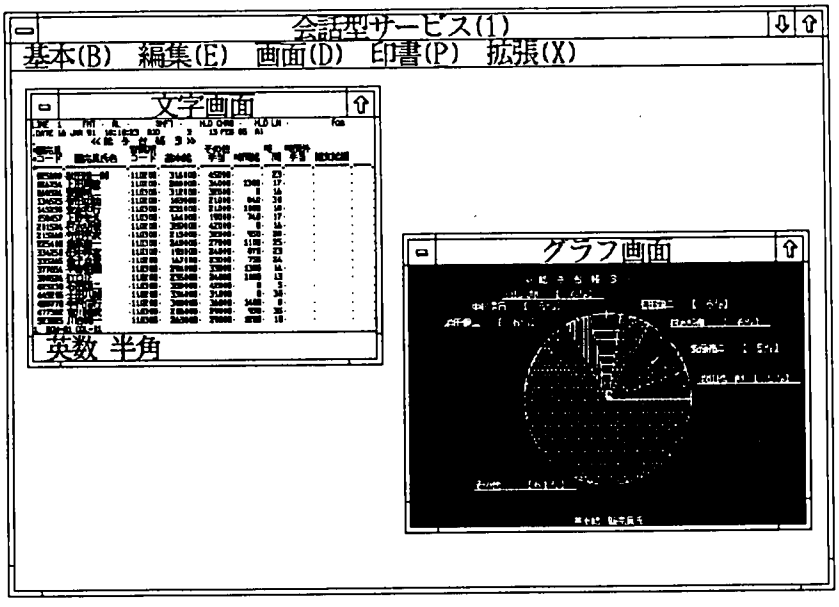



図 10 マルチウィンドウによるグラフとレポート表示

Fig. 10 Graph and its data displayed on multi-windows

ムの役割関係の上では，その連動に限界がある.）

### 4.1.3 既存操作との互換モード

MS OS/2 の PM 上にインプリメントされたインフォメーション・サービスは，二つの表示モードを備えている．一つは，先に示した PM 標準のタイトルバーを持つウィンドウ形式であり，他の一つが従来と同じに全表示領域にターミナル情報を表示するモードである．

図 11 に後者の表示例を示す．全画面表示モードでは PM で提供される機能を利用

```
LINE▸ 1     FMT▸   RL▸       SHFT▸     HLD CHRS▸    HLD LN▸   ▸       fcs       ▸
.DATE 14 JUN 91  18:21:26  RID      3     13 FEB 85  AI
.                 << 給 与 台 帳  3 >>
*販売員                    営業所                  その他          時    時間外
*コード    販売員氏名      コード   基本給          手当   時間給    間   手当   総支給額
*=======.==============.======.========.========.======.====.======.========.====.
 025688 秋田浩一郎        110200   316000   45000                23
 026754 上田周造          110200   288000   34000    1300        17
 068524 安藤裕二          110300   312000   32000       0        16
 134525 青田功助          110200   183000   21000     840        30
 145258 安本和行          110300   231000   21000    1000        18
 158457 上野幸久          110300   164000   19000     740        17
 201524 村木明俊          110200   359000   42000       0        16
 215268 守田平次          110300   215000   32000     950        28
 225400 塩原義一          110300   249000   27000    1100        25
 334250 田中淳次          110300   193000   24000     870        23
 335265 金子貞造          110200   167000   23000     750        24
 377654 平野史朗          110300   294000   33000    1300        14
 388524 江口正            110200   235000   24000    1000        13
 425230 石田諒二          110300   358000   42000       0         5
 445205 土田八郎          110200   334000   31000       0        30
 458770 中村信行          110200   308000   36000    1400         8
 477582 宮川清次          110300   206000   29000     930        35
 523625 川合純一          110300   263000   29000    1200        10
1  ROW=01 COL=07
英数 半角
```

**図 11　既存システムとの互換表示**

Fig. 11　Compatible display to existing system

できないが，既設置のターミナルシステムと完全に互換性を保ちたい場合，また専用システムとしてターミナルシステムを利用する場合に適する表示モードである．また使用が限られたシステムの場合，誤操作を防ぐ点でも有効である．

### 4.1.4　グラフィカル環境の統一操作

マウスによる基本操作（編集機能等）をターミナルシステムに採用することは部分的ではあるが，ワークステーション上のソフトウェアと同一操作が可能となる．代表的な操作がカット&ペイスト操作である．これらの操作はワークステーションのソフトウェアと同じく，基本的なマウス操作で実行することができるため，機能増による操作の修得の煩わしさはない．また，マウス利用によるテキスト編集操作等，UTS ターミナルシステムの編集操作の他に，共通操作の編集操作によって目的とする結果を得ることができる．

以下に，マウスによって使うことができる主な機能を列挙する．

1)　マウスによるカーソル移動……ターミナル操作では，アプリケーションによって設計された画面様式に従い，入力フィールドへのカーソル移動を行うことが多い．マウスカーソルは，縦横斜方向の手を動かす方向に移動させ，操作対象を選択するダイレクト操作ができるため，従来のスキャンキーやタブキーによるカーソル移動より直接的に操作が行える利点がある．

2)　マウスによる送信操作……メニュー選択方式，とくにカーソルポジションによって選択されたサービスを判断するように構成されたシステムでは，マウス操作のみで選択できるシステムを構成することが可能となる．オペレータが画面上のメニューを直接的に選択する操作が可能となる．

3)　マウスによる入力画面の作成……DPS 1100 (Display Processing System) を用いて入出力用の画面を作成する場合，入力フィールド等は罫線で囲む表示形式

が利用されている．マウスによる罫線作成は，作成する位置にマウスカーソルを位置づけ（ダイレクト操作），ドラッグ操作（マウスのボタンを押したままボタンを動かす）で作成することができる．従来のキーボードのスキャンキーによる作成より簡単に行うことができる．

4) SOE の入力……UTS の機能は，SOE(▷)からカーソル位置までのデータを送信する仕様であるため，レポートデータの変更を行う際等，MAPPER システムでは必須な機能となる．

5) 接続操作……図 12 は，ホストシステムとの接続画面がダイアログボックスで表示されている IS-PC/IS-CS 会話型機能の初期画面である．オペレータは，示された接続先からマウス操作によって選択することができる．

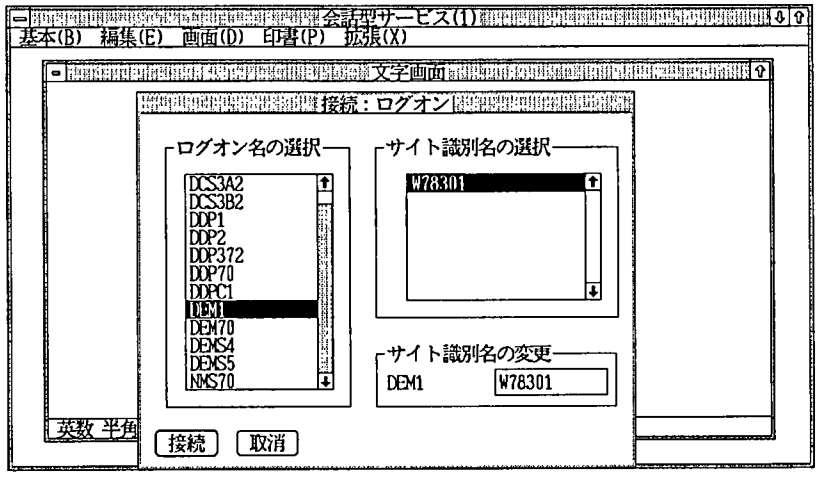



**図 12　接続（初期）画面**

Fig. 12　Connection (Initial) screen format

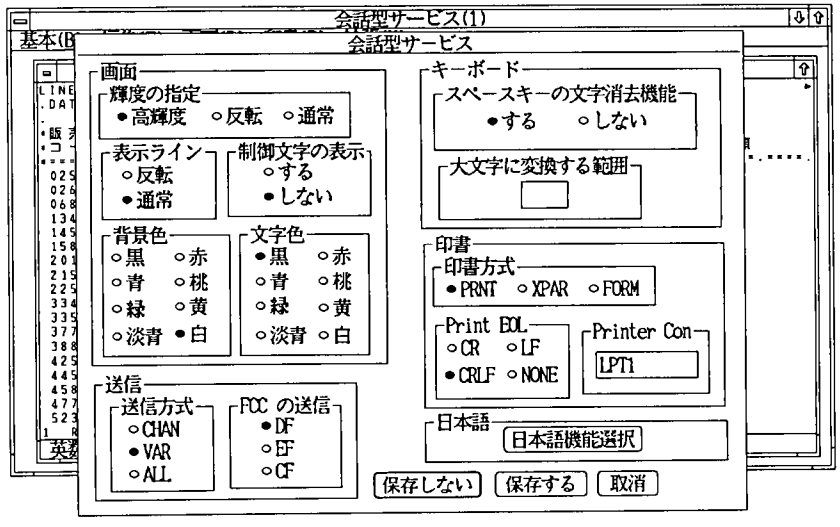



**図 13　コントロールページ**

Fig. 13　Control page

6)　コントロールページ……図13は，コントロールページの表示例である．タイトルバーから基本を選択しコントロールページを選択する．操作に必要な情報は，画面に表示できるため，コントロールページに割り当てられたキーを探す必要はなく，メニューから選択することによって目的とする機能を選択することができる．

7)　キーボードヘルプ……すべての編集操作は，キーボードとマウス操作の両者で行うことができる．タイトルバーを選択することにより，機能とそれに対応するキー操作を得ることができる．この機能は，キー割当の問い合わせ機能とマウスによるキー操作の二つを兼ねている（図14）．

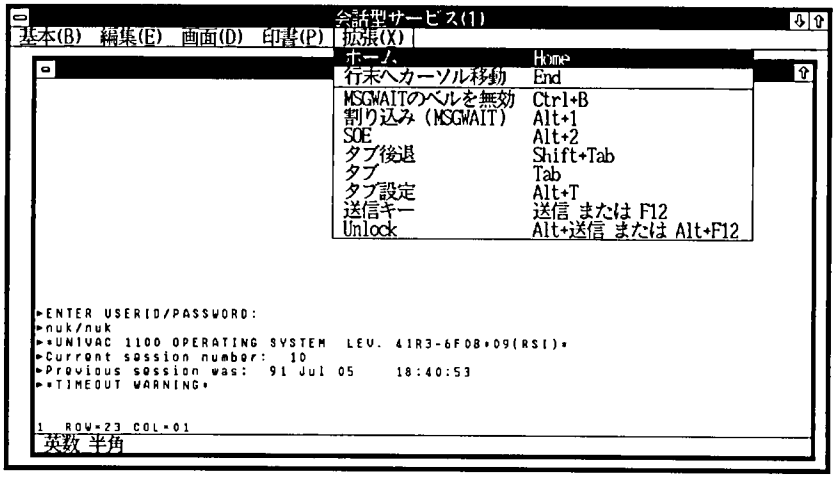



図 14　キーボードヘルプ
Fig. 14　Keyboard help

### 4.2　OA パッケージと連携機能

IS-PC(IS-CS)は，従来のターミナルシステム上にユーザプログラミング機能の他，プログラム間通信サービスを備えるが，さらに PM の環境では，データ入力においてもデータ入力の容易なシステムの構成を支援するユーザプログラミング機能を備えている．また，PM 上での共通機能であるカット&ペイストを備え，ワークステーション上のソフトウェア間のデータ交換ができる．

1)　カット&ペイスト機能……ターミナルシステムのウィンドウに表示された MAPPER レポートやグラフをワープロや表計算プログラムに取り込むこと，あるいは逆に MAPPER レポートに取り込みが可能である．図15にカット&ペイスト中の表示例を示す．

2)　簡易入力……従来のターミナルシステムでは，文字列入力の効率化を図るためハードウェア装置（スクリーン・キーボード）を採用していた．PM 環境では，ウィンドウ表示し，その中から採用する方式を選択できる（図16）．

## 5.　PM によるオーバーヘッドの対策

応答性は，ターミナルの操作性能を左右する一つの要素である．しかし，GUI を用

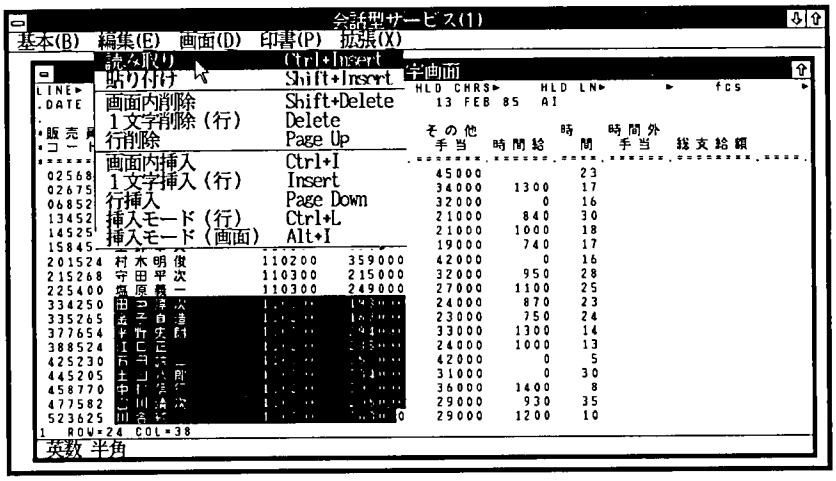
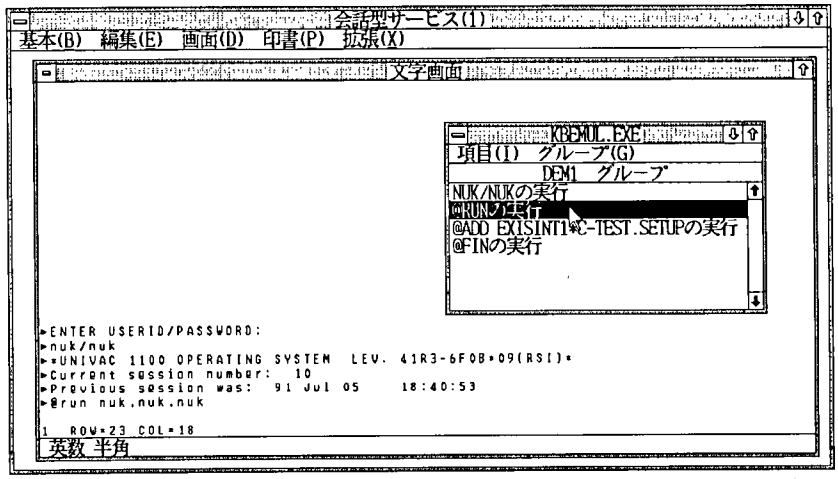



図 15 カット&ペイスト

Fig. 15 Cut & paste




子ウィンドウの中のメニューに対応した文字列がターミナルプログラムに送られる．この例では，@RUN の実行を選択すると@run nuk, nuk, nuk が送られる．メニューの定義，メニューに対応した文字列の定義はプログラミングが必要．

図 16 キーボード API の例

Fig. 16 An example of application using keyboard APIs

いることによって表示処理データ量は 2 桁増加する．また PM は，通信システムのように非同期に入力されるデータをハンドリングするリアルタイム処理には向いていない．そのために，応答性に関していくつかの技術的な問題が生じる．問題点の概要を示す．

「ターミナルシステムの応答性」

グラフィック表示は同一情報（1 文字）を扱うために数十倍の情報を扱う必要がある．たとえば 24×24 のフォント文字を表示する場合，72 倍の CPU を必要とする．近年，CPU の速度は飛躍的に向上しているが，1 文字を文字ベースの表示機構と同等の速度でグラフィック上に表示できる程には CPU 速度は向上していない．

表1　MAPPER レポートの表示時間と 100 ライン表示(デマンド)

Table 1　Display time of a MAPPER report and 100 lines on demmand mode

| | DS 7 | PW²(PM) |
|---|---|---|
| 時間（秒） | 3.0 | 2.8 |

MAPPER レポートの表示時間

| | DS 7 | PW²(PM) |
|---|---|---|
| 時間（秒） | 23 | 38 |

デマンドシステムでの比較
(100 ラインの連続表示時間)

ターミナルシステムは，基本的に回線からミリセカンド単位で非同期に入力される情報を1文字単位に表示するシステムである．そのため，文字表示機構は高速な表示を可能とするハードウェア機構によって実現されていた．

PM 上の IS-PC/IS-CS では，オーバヘッドを減らすため，表示を文字単位から文単位にすることにより，応答性を確保している．表1は MAPPER レポートの表示時間を DS 7 の UTS エミュレータと比較した表である．

文字単位から受信文単位に変更することにより表示開始が遅れる問題が発生するが，最近では通信回線の速度は高速化（9600 bps 以上）されており，表示に比べ伝送にかかる時間の割合が小さくなり，逆に文字単位の表示処理の必要性が少くなっている．

## 6. おわりに

GUI を備えるターミナルシステムの利点，すなわち統一された共通操作が行える点を述べた．これらの機能の中には，評価版を試用いただいたユーザの方々からのご指摘による改良が含まれている．この場をかりて謝意を述べたい．

最後に，今後の課題として次のことを指摘したい．前述のようにインフォメーション・システムにはすでにいくつかの（メニュー選択・カーソル移動等）共通操作が実現されている．しかし，このような改善は「外付け機能的」なものであり，根本的改善ではない．現在稼働中のホストシステムとターミナルシステムとの処理の関係，すなわちホストシステムがすべてを編集・加工し，その結果をターミナルシステムが表示する役割分担では，ターミナルシステムに GUI を備えてもシステム全体には大きな改善は得られない．たとえば，GUI によるメニュードリブン方式，ドラッグ等のオブジェクト指向操作，スクロール移動の方式，マルチフォント，画面表示方式，データ入力時点での妥当性検査，エラー表示方式等ホストシステムを変更しない限り利用できない処理があり，操作を統合化することができない．すなわち GUI 化は，ターミナルシステムの操作そのものに対しては操作性の向上はできるが，本来のユーザ業務の処理システムの改善は期待できない．これに対する解決策の一つがクライアント/サーバ方式であろう．ホストシステムとワークステーションの関係をデータサーバと処理加工システムに変えることが今後求められてくるであろう．

**執筆者紹介** 佐々木　茂 (Shigeru Sasaki)

昭和44年新居浜高等専門学校電気工業科卒業．45年日本ユニシス(株)入社．ハードウェアの保守に従事した後，ターミナルシステム，ワークステーションシステムのソフトウェア開発に従事．現在，ワークステーション・ソフトウェア1部開発2課課長．

# 垂直分散型ネットワークにおける統合データ交換環境の構築

# The Creation of an Integrated Data Exchange Environment for Vertical Distribution Networking

北 川 達 朗

要 約 近年，急激な情報量の増大，業務の多様化に伴い，より効率の良い複数コンピュータによる分散処理へとシステム体系が移行しつつある．

一方，コスト・パフォーマンスの優れた高性能ワークステーションの普及，それに伴うLAN等の中小規模ネットワーク技術の進歩により，柔軟性・拡張性の高い分散処理システムを構築することが可能となってきた．

このような状況のもとで，汎用機レベルの水平分散を実現していたシステム基盤ソフトウェアDDA（Distributed Data Administration System, 統合データ交換システム）に対してもワークステーションへの展開が求められた．この垂直展開に際し，対象とするコンピュータは，オープンシステム化の流れにそって，UNIX*, MS OS/2**搭載マシンを選定し，実装することとなった．

本稿では，このプラットフォームにまたがった統合データ交換環境の構築時の基本的考え方，留意事項，および現状の問題点を明らかにする．さらに，今後主流となっていくであろうインフォメーション・ネットワークにおいて必要とされる統合環境について言及する．

**Abstract** With the volume of information radically increasing and with applications surgingly diversifying these days, the structure of computer systems has been shifting to distributed data processing based on more than one host computer linked for higher efficiency. Meanwhile, the wide acceptance of high-performance workstations with excellent cost-efficiency and incidental technological development for small-to medium-size networking including a local area network have jointly made it possible to build a distributed data processing system providing high flexibility and expandability.

The background like this gave a rise to the new demand that the Distributed Data Administration (DDA), a systems infrastructure software product, which was already used for the horizontal distribution of general-purpose computers, be enhanced so it also could support workstations. To meet this vertical expansion requirement, the author's team's implementation effort has been made for selected Unix- and MS OS/2-based machines in line with the industry trends toward interconnect systems.

This paper is focused on basic concepts, some points worthy of attention and existing problems in connection with the creation of the environment for integrated data exchanging throughout different platforms. Also an additional mention is of the integrated environment required for information networks that will be a mainstream of future computer applications.

## 1. はじめに

近年，ユーザは自己の業務を確実にかつ効率よく処理するために，大型汎用機による集中処理から，汎用機レベルの分散処理へ処理状態を変えてきている．しかし，ユ

* UNIXオペレーティングシステムは，UNIX System Laboratories, Inc.が開発しライセンスしている．
** MS OS/2は米国マイクロソフト社の登録商標である．

ーザのかかえる問題は思ったほどには解決していない．逆に，

① 業務の多様化，システムのネットワーク化に伴う新規アプリケーション開発の必要性の増大，

② ①に伴う開発コスト，運用コストの増大，

等の点に関し，問題が顕在化していると言えよう．

一方，絶え間ない技術革新により，UNIX，MS OS/2 等のオープン性の高い OS を搭載した高性能ワークステーション，パソコン等は大幅にコスト・パフォーマンスが向上してきている．CPU 能力の絶対値も小型の汎用機を凌駕するまでになっている．

このような環境の変化に伴い，従来の情報処理システムを見直す動きが見られる．つまり，汎用機が得意とする大量データの処理等は従来のシステムで行うが，汎用機で行う必然性のない業務は，ワークステーション等で分散構築するといったアプリケーション分割型分散システムへの動きである．

このユーザ要求に応えるためにはベンダとして，

① データ/データベースのネットワーク内の配置をユーザ・アプリケーションに対して透過とする環境，

② データ/データベースをネットワーク内の任意のノードに容易に転送できる環境，

を提供していかねばならない．

本稿では，この命題に対し汎用機レベルでのデータ交換処理環境を提供するプロダクトである DDA（Distributed Data Administration system）をワークステーション対応することによって上記②を実現した経緯を述べるとともに，今後情報処理システムの主流をしめるであろうユニシス・アーキテクチャ(Unisys Architecture，以下 UA と記述）の規定するインフォメーション・ネットワーク（Information Network，以下 IN と記述）において実装する場合の望ましい形態について考察する．

## 2. DDA による垂直分散ネットワークへの対応

### 2.1 DDA 概要

分散配置されたコンピュータ機器（ホスト，ワークステーション，パソコン，…）間でデータ（ファイル，トランザクション，メッセージ，ドキュメント，…）の交換を行うためのシステム基盤ソフトウェアに対する要求は，ここ数年来とくに先進ユーザにおいて顕著になってきた．業務システムの複雑化，それを処理するコンピュータ・ノードの多様化，ノードを接続するネットワーキング技術の高度化に伴い，データ交換処理に対し統合的ユーザ・インタフェースを求めるのは，当然の成り行きと言える．ここで，統合的ユーザ・インタフェースとは，

① アプリケーション・インタフェースの統合，

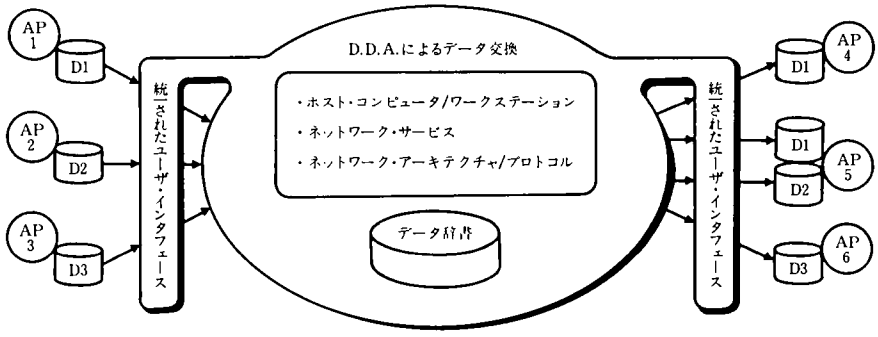
② オペレーション・インタフェースの統合，

③ 運用管理の統合，

を指す．

すなわち，その目的とするところは，すべてのコンピュータ・ノードに統合化されたユーザ・インタフェースを提供するシステム基盤ソフトウェアを導入することによ

り，アプリケーション開発，およびその運用における生産性の向上を図ることである．DDA は，この目的を達成するために開発されたものである（図 1）．




APn：業務処理プログラム
Dn　：データ

**図 1　DDA の概念**

Fig. 1　The concept of DDA

## 2.2　DDA の機能

DDA が規定している機能は，大きく以下に示す点である（図 2）．

1)　ネットワーク辞書生成・保守機能……DDA は，DDA ネットワークに加入するすべての構成要素を定義するコンフィグ情報としてネットワーク辞書を持ち，構成要素として下記を規定する．

①　ノード：DDA ネットワークに加入するすべてのプロセッサに対して 1：1 に設定する構成要素である．

②　ルート：各々のノードから他のノードへの情報の流れを規定する構成要素である．

③　データ：一つのノードからルートを介して他のノードへ転送される情報の論理単位に対して設定される構成要素である．通常，物理データ（ファイル，トランザクション・データ，…）と 1：1 に対応している．

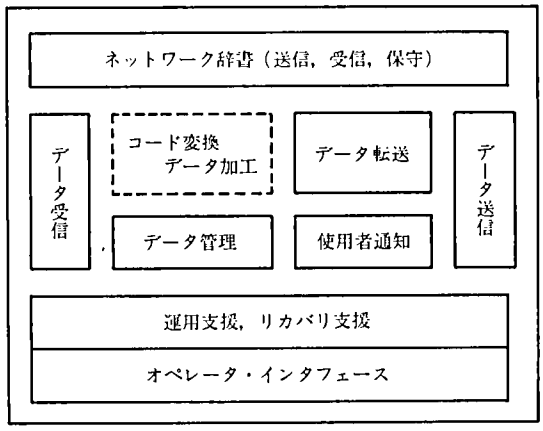



**図 2　DDA の機能構成**

Fig. 2　The function structure of DDA

④ アプリケーション：データの所有者，または利用者を示す構成要素である．これらの情報を一元管理することにより，ネットワーク全体の整合性を保つことができる．また，一元管理された辞書の他ノードへの配布機能，他ノードでの辞書受信機能を持つ．

2) データ送受信に関するユーザ・インタフェース機能

実際のデータ送受信に関する下記インタフェース機能を提供する．

① データの送信登録
② データの送信指示
③ データの受信指示
④ データの送信登録取消し
⑤ 受信側からのデータ受信要求
⑥ 受信側からのデータの受信要求取消し
⑦ データの送信完了時および受信時の使用者通知

3) 運用支援機能

システム運用に関し下記の支援機能を提供する．

① DDA システムの導入・更新，および初期化
② 自ノードおよび他ノードリカバリ
③ オペレータ・コンソール支援
④ システム稼働状況ログアクセス

これらの機能を実現することにより，DDA は分散されたコンピュータ・ノード間でのデータ交換のためのシステム基盤ソフトウェアとして位置づけることができる．

## 2.3 DDA の実装

### 2.3.1 UNISYS シリーズ 2200・1100 ホストへの実装

前述の基本概念および機能定義のもとに，1987 年に DDA はまず DDA 1100 として，異機種を含む汎用機レベルでの水平分散を実現した．当初汎用機レベルの実現に留まった理由は，

① 当然のコンピュータ・システムを取り巻く諸環境が汎用機中心に構築されていたとき，
② ワークステーション側の諸条件（ハードウェア性能，ソフトウェアの品揃え等）が非常に貧弱であった，

等があげられる．

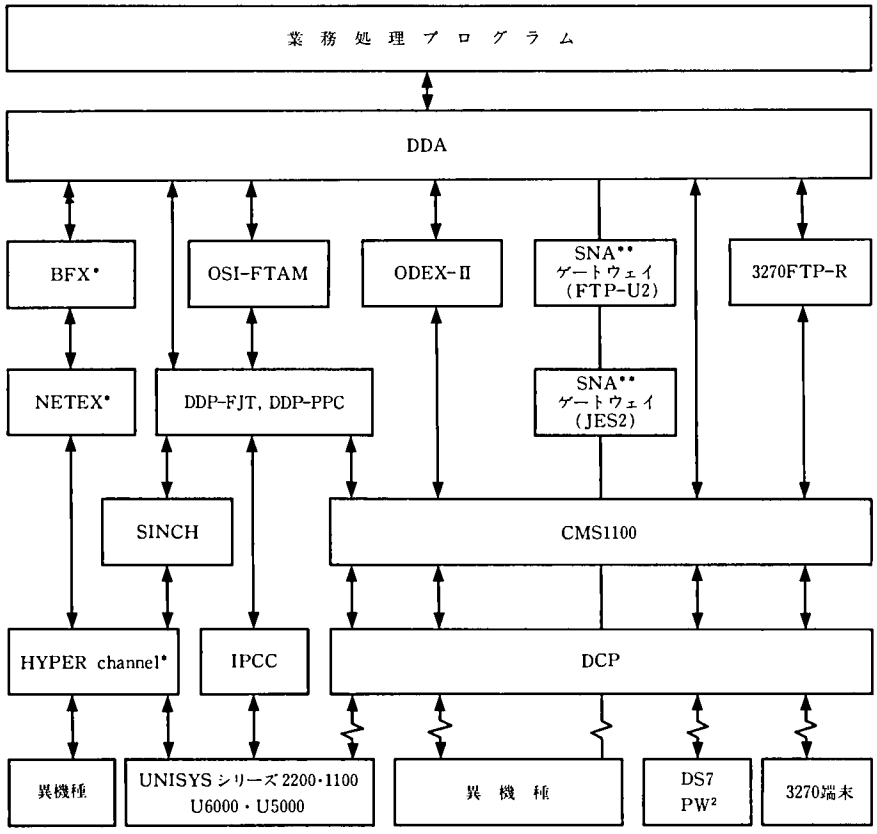
シリーズ 2200・1100 ホスト上でのソフトウェア構成を図 3 に示す．この図で示されている通り，複数のファイル転送プロダクトを統合し同一のユーザ・インタフェースを提供することにより，ユーザに対しネットワーク透過な環境を実現した．

### 2.3.2 ワークステーションへの実装

1989 年頃になって，当時活況を呈しはじめた UNIX，OS/2 等を搭載したマイクロ系コンピュータをコンピュータ・ネットワークへ組み込むという試みが注目を集め始めた（図 4）．

汎用機レベルでのデータ交換処理環境を実現していた DDA を，マイクロ・システムへ垂直展開したいという要望が，当社のユーザである A 社から出されたのも当然の成り行きであった．

マイクロ系コンピュータのプラットフォームは，該社のネットワーク構成との関係および下記の理由から，U 6000 シリーズ（UNIX マシン），PW²ファミリ（MS OS/




ODEX：Online Data Exchange
IPCC　：Inter Processor Channel Coupler
CMS　：Communication Management System
DCP　：Distributed Communication Processor
*BFX(Bulk File Transfer), NETEX(NETwork EXecutive program), HYPER channel は，Network System 社の登録商標である．
**SNA (System Network Architecture) は，IBM 社のネットワーク・アーキテクチャである．

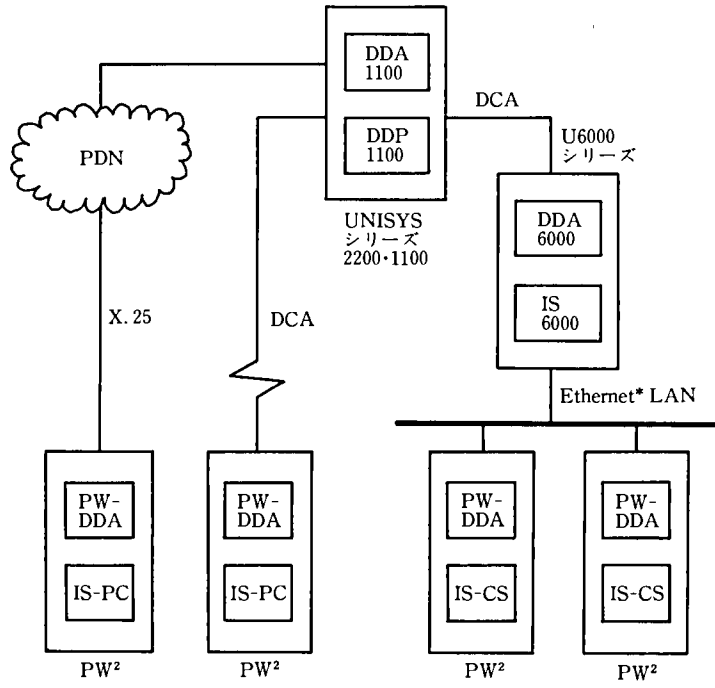
**図 3　DDA1100 のソフトウェア構成**
Fig. 3　The software structure of DDA1100

2 マシン）が選定された．

1)　U 6000 シリーズへの適用……当社の提供する UNIX マシンには，U 6000 シリーズおよび US ファミリがあるが，シリーズ 2200•1100 ホストとの接続性の観点から IS 6000 (Information Service 6000) により DDP (Distributed Data Processing) プロトコルを提供する U 6000 シリーズを採用した．

また接続プロトコルとして，DDA 1100 では DDP プロトコル以外に，いくつかのプロトコル（当社独自および対異機種用）を提供するが，それらをマイクロ・システム側に取り込むことは，開発マンパワーの問題，今後の展開上のリスクを考慮すると採用することはできなかった．

したがって，ハードウェアとして U 6000 シリーズ，ソフトウェアとして




*Ethernet：米国Xerox社の登録商標である．
PDN：Public Data Network
DCA：Distributed Communications Architecture

**図 4　DDA による汎用機ーワークステーション結合**

Fig. 4　The host-workstation combination by DDA

IS 6000，接続プロトコルとして DDP を適用することとした．

2)　MS OS/2 マシンへの適用……MS OS/2 が稼働するマシンは，現在 PW²ファミリがあり，該マシン上で DDP プロトコルを提供する IS-CS (Information Service-Communication Server)/IS-PC (Information Service-Personal Computer)が，稼働可能となった．このセットでの環境を前提とすると UNIX マシンと同環境となり，シリーズ 2200・1100 ホストを含めた相互接続に関し非常に高い親和性を確保できることになる．

3)　IS のデータ転送機能の適用……上記の理由により，DDP 1100 および IS 6000，IS-PC による DDP ネットワークを，DDA データ転送のプラットフォームとすることにした．DDA 1100 側から見ても，汎用機間の水平分散において DDP プロトコルを使用しているため，マイクロ・メインフレーム接続に関し修正を加える必要がないという利点があった．

DDA が使用する IS/DDP 1100 の基本転送機能は下記の二つである．

①　FJT (File Job Transfer) 機能のうちのファイル転送機能

②　プログラム間会話処理

ファイル転送機能は，実際の交換対象データがファイルの時のみ使用する．

プログラム間会話処理は，交換対象データの制御情報の授受および交換対象デ

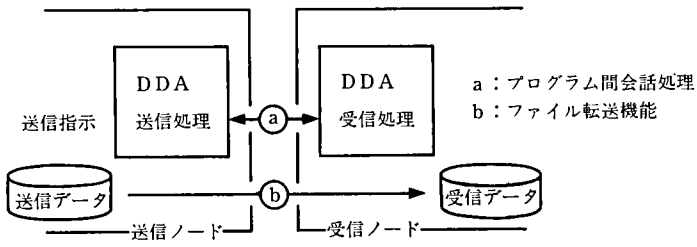



**図 5　IS のデータ転送機能と DDA 処理**

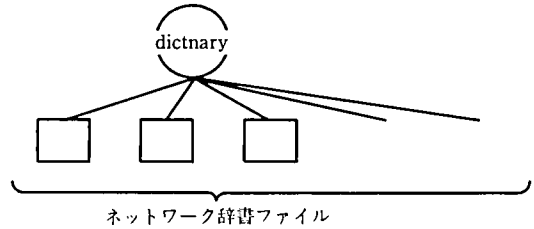
Fig. 5　The data fowarding function of IS & DDA processing

ータがトランザクションである場合に使用する（図 5）.

4)　DDA 機能の実現……以上のように，マイクロ系システムでのハードウェア，ソフトウェアのプラットフォームを決定した．ここでは，そのプラットフォーム上で，DDA 機能をいかに実現したかについて記述する．

①　ファイルシステム

DDA は，ネットワーク辞書，使用者データ，送受信を行うための各種制御情報を一元管理するため，dda ディレクトリ以下の階層的ファイルシステムを持つ．ディレクトリ・ツリーは図 6 のようになる．また，ディレクトリの詳細を表 1 に示す．




appldef：アプリケーション定義ファイル
ctlgdef：カタログ情報定義ファイル
datadef：データ定義ファイル
formdef：データ形式定義ファイル
lifedef：データ保管定義ファイル
nodedef：ノード間定義ファイル
optdef：ユーザ情報定義ファイル
owndef：自ノード定義ファイル
passdef：データ配布先定義ファイル
procdef：通知先情報定義ファイル
rutddef：ルート定義ファイル
sfixdef：データ定義補足ファイル
startrun：バッチラン定義ファイル
tpsname：TPS 定義ファイル
userdef：データ利用アプリケーション定義ファイル

**図 6　DDA ディレクトリ構成**

Fig. 6　The directory structure of DDA

そのうち，データ転送処理に必須のネットワーク辞書ファイルは図 7 のような構成となっている．

図 7 のようにネットワーク辞書の各構成要素ごとに別ファイルとすることにより，アクセス頻度の高い辞書ファイルの検索が容易かつ高速に行えるよう考慮している．

表 1 DDA ディレクトリの内容
Table 1 The contents of DDA directories

| ディレクトリ名 | 説　明 |
|---|---|
| dda | DDA が管理するすべてのファイルを持つディレクトリ |
| dictnary | ネットワーク辞書ディレクトリ |
| snd | データ送信処理ディレクトリ |
| rcv | データ受信処理ディレクトリ |
| odr | データ送信要求ディレクトリ |
| can | データ送信要求取消ディレクトリ |
| prg | DDA 実行プログラム管理ディレクトリ |
| msg | エラーメッセージ保管ディレクトリ |
| log | ログ情報管理ディレクトリ |
| reg | 辞書登録データを管理するディレクトリ |
| tmp | テンポラリ・データを管理するディレクトリ |
| ctl | データの授受を行う際の制御情報を，このディレクトリ下に使用者データごとにファイルとして生成する． |
| files | 実際に送信を行うファイルをこのディレクトリ下に複写して管理する．複写する際，辞書登録データの場合はデータ識別子を，テンポラリ・データの場合は物理ファイル名をそれぞれファイル名とする． |
| clt | データ受信要求を行った場合，このディレクトリ下に制御情報を使用者データごとに作成する． |
| svr | データ受信要求を受けた場合，このディレクトリ下に制御情報を作成する． |

② UNIX，MS OS/2 のプラットフォームごとの実装

DDA 機能を，UNIX，MS OS/2 搭載マシンに適用する上で，両者に機能的な差異を生じさせないことは勿論であるが，その実現方式も統一することを基本とした．しかし，プラットフォームの持つ機能・インタフェースの違いにより，どうしても各システム独自の方式を採らざるを得ない部分が発生する．最も大きな点は，ウィンドウ・システムおよび IS の機能性による相違である．

まず，ウィンドウ・システムであるが，U 6000 シリーズではメニュー開発システムを使用している．このため，見かけはマルチ・ウィンドウであるが機能性はかなり低いと言わざるを得ない．PW²ファミリでは当然 PM (Presentation Manager) ベースのウィンドウ・システムを構築している (図 8)．

また，IS の機能性が IS 6000 と IS-CS/IS-PC とでかなり異なっている．IS-CS/IS-PC が IS 6000 と比較して不足している点は下記の通りである．

- 会話 2 次局プログラムを，会話設定時に自動起動することができない．したがって，2 次局プログラムを常駐させなければならない．
- 会話 1 次局プログラムの複数稼働ができない．同様に，会話 2 次局プロ

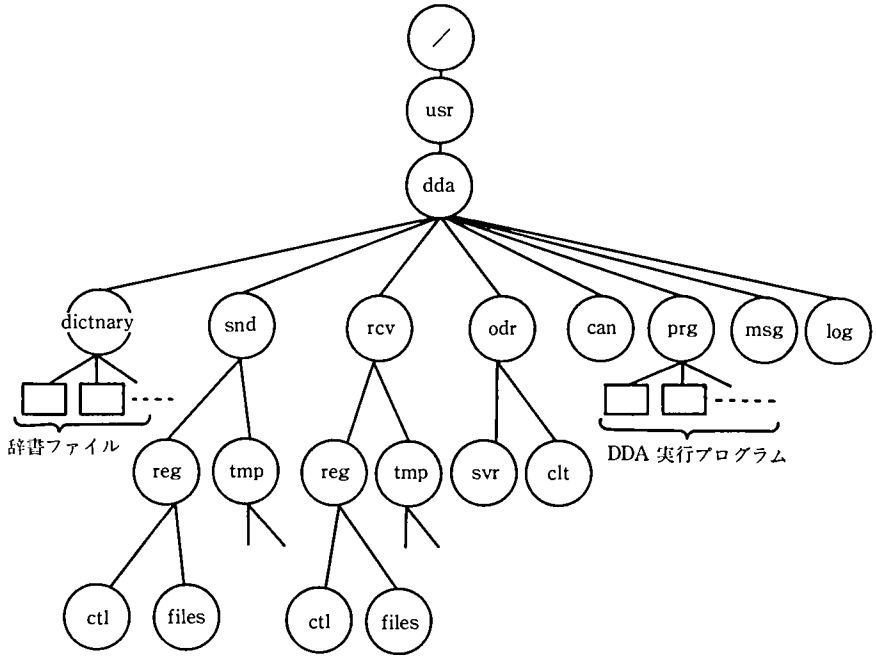



図 7　DDA ネットワーク辞書構成

Fig. 7　The network directory structure of DDA

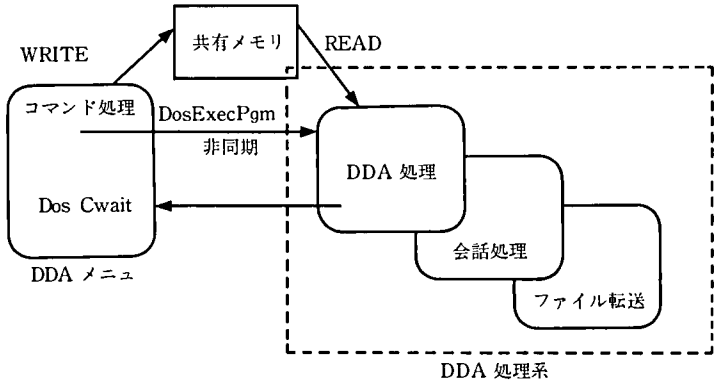



図 8　PW² (MS OS/2) における DDA メニュー・システム

Fig. 8　The DDA menu system on PW² under MS OS/2

グラムの複数稼働ができない．

したがって，DDA における IS からむ処理要求の並行処理および処理要求受信の並行処理は，DDA レベルで制限している．

これらは，ユーザ・システムに対して影響が大きいので，機能拡張していかねばならない．

## 2.4　DDA の今後

DDA をマイクロ系のシステムに実装したことにより，DDA 自体の可用性は高まったと言える．

しかし，さらにシステムの付加価値を高めるため対応すべき点は多い．以下にその項目をあげる．

1) ワークステーション・レベルでの水平分散対応……現状では，シリーズ 2200・1100 ホスト～UNIX マシン，シリーズ 2200・1100 ホスト～ワークステーションといった垂直分散に対応しただけである．これをさらに，ワークステーション～ワークステーションのデータ交換システムへと拡張する．通信プロトコルも，接続性を考慮して TCP/IP (Transmission Control Protocol/Internet Protocol)，FTP (File Transfer Protocol) を採用する．

2) ディレクトリ構成の拡張……現在，DDA は各ノードごとにディレクトリ・ツリーを保持しているが，必ずしもその必要性はない．NFS (Network File Server) 等のシステムを取り込むことにより資源の共有化を進めるべきである．

3) ユーザ・インタフェースの拡張……前にも述べた通り，UNIX の標準ウィンドウ・システムである X/Window* 対応がなされていない．ウィンドウ・システムの機能性，今後のネットワーキングを考慮すると対応は必須である．またヘルプ機能を実装することも考慮する必要がある．

### 2.5 環境ソフトウェアとしての DDA

DDA により，シリーズ 2200・1100 ホスト，UNIX サーバ，ワークステーション上に整合性のあるデータ交換のための環境が提供された．しかし，その環境は，

① 開発時期の違いにより，1100 ホスト主導のシステム化となっている，

② ホスト～UNIX 機，ホスト～ワークステーションといった垂直分散型ネットワークを主眼とした実装となっており，完全に「使用者に対してノードの透過性を確保している」とは言えない，

といった点があり「統合された環境」とは言いきれない．

また，プロダクトの内部構造という観点からみると

① 1100 主導のプロダクト化の影響が大きく，マイクロ系システムの特徴を活かしきれていない，

② X/Open** 対応，OSI 対応，オブジェクト指向といった標準/先端技術に対して柔軟に対応できる構造ではない，

と言う点が指摘できる．

また，UA が規定する IN (Information Network) に適合させることは，今後開発・提供されるすべての米国ユニシスおよび当社プロダクトの必要条件としなければならない．その意味でも，DDA は何らかの対応をせざるを得ないであろう．

## 3. 統合環境実現への道

各々のプラットフォームにおけるハードウェア性能の向上，ソフトウェアの充実化，それらを相互に接続するネットワーキング技術の進展により，使用者の業務システムは種々の実装形態が考えられるようになってきている．しかし，使用者の個々の業務システムが独自の実装を行っていたのでは，システム全体として整合性を保つことは

---

* X/Window：M. I. T.の登録商標である．

** X/Open：X/Open 社の登録商標である．

むずかしくなってくる．

このことは，業務システムの基盤となる環境（インフラ・ソフトウェア）の構築についても言えるであろう．環境の構築を個々の業務ごとに行うとすると，同様の環境を重複して抱えることにもなりかねない．また，逆にプラットフォームごとに各々に親和性のない環境を構築することになるかもしれない（図 9）．

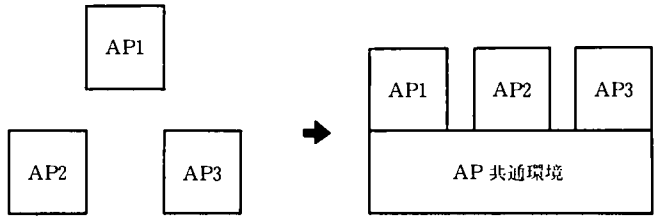



図 9　業務の統合化

Fig. 9　The integration of application systems

これらの危険性を排除した業務システム，プラットフォームに均一な環境を「統合環境」と呼ぶこととする（図 10）．

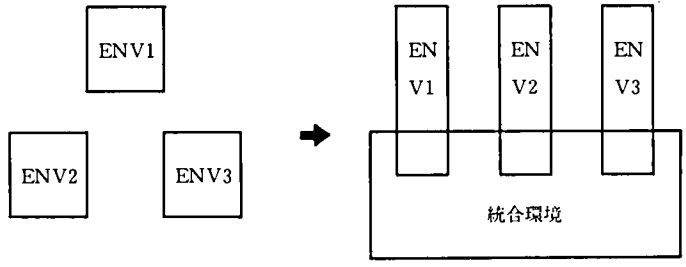



図 10　環境の統合化

Fig. 10　The integration of application environment

この統合環境を提供することにより，分散システムにおいて使用者が求めている，プラットフォームに亘る相互運用性を確保することができる．

以下，要求される統合環境についての考察を行う．

### 3.1　統合環境の目的

インフォメーション・ネットワーク（コンピューティング・パワーが分散されたネットワーク）上で統合的な業務処理環境を実現することを目的とする．

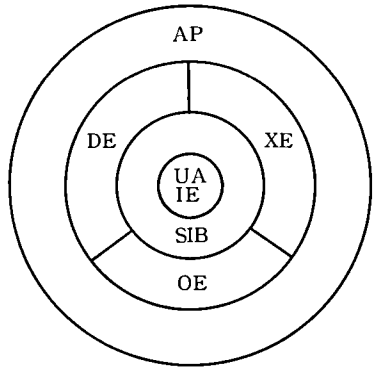
統合環境は，開発環境，実行環境，運用環境，の三つからなり，それらの関係は図 11 のようになる．

これらの三つの環境により，使用者システムは，

① IN 上の各アプリケーションの協調動作，

② 各プラットフォームの特性を活かした機能分担，

③ IN の相互運用性の確保，




UA：ユニシス・アーキテクチャ(Unisys architecture)
IE ：分散システム統合環境(Integrated Environment)
SIB：システム情報データベース(System Information Base)
XE ：実行環境(eXecuting Environment)
OE ：運用環境(Operating Environment)
DE ：開発環境(Development Environment)

**図 11　統合環境の構成要素**

Fig. 11　The composed elements of integrated environment

④ IN における一元的な運用管理

⑤ システムの拡張性の確保,

⑥ 業務処理プログラムの移植性の確保,

を容易に獲得できることになる.

### 3.2 統合環境の基本的考え方

統合環境を提供するにあたっての基本的考え方を次に示す.

① 国際/国内/事実上の標準に準拠する.

② 標準の定まっていない分野については独自の仕様を設定する. ただし, その仕様設定に関してはとくに移植性に考慮を払う.

③ 既存のソフトウェア資産の継承を可能とする.

④ 各プラットフォームへの実装時には各々の固有性を配慮する.

とくに,「標準」への対応に関するオープン性と,「既存のソフトウェア資産の継承」に対応する固有性の両立については十分に考慮する必要がある.

### 3.3 統合環境が前提とするモデル

1) ネットワーク……基本的には UA が規定する IN を前提とする. すなわち, ハブ/サーバ/ワークステーションが LAN/WAN で有機的に結合されたネットワークである. 当然, 他ベンダマシンもネットワーク・ノードの対象となる(図 12).

2) アプリケーション・システム……統合環境およびユーザ・インタフェースは, 各プラットフォームに対して透過であるが, その上で稼働するアプリケーションは, 各プラットフォームの固有性を意識せざるを得ない. 各々の特徴は下記の通りである.

① ハブ：情報の中枢を司るシステムであり, 巨大な複合サーバ機能を持つ. サービスする内容は, 大規模データベースへのアクセスが中心である.

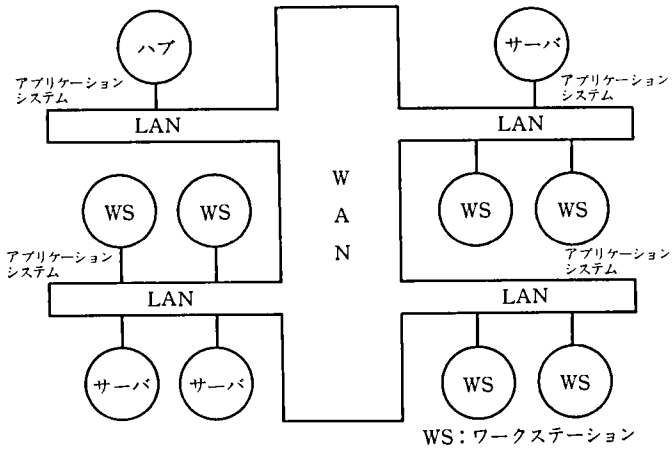



**図 12　統合環境におけるネットワーク・モデル**

Fig. 12　The network model in integrated environment

② サーバ：局所的な業務処理，あるいはハブのサーバ機能に対するクライアント機能およびワークステーションのクライアント機能に対するサーバ機能を有する．

③ ワークステーション：個人レベルの業務処理，あるいはヒューマン・インタフェースを提供し，かつハブ/サーバの持つサービス機能に対するクライアント機能を有する．

3) インタフェース……統合環境において定義するインタフェースには，アプリケーション・インタフェース（API），ヒューマン・インタフェース（HUI），システム間インタフェース，がある．各々の実装は図 13 の通りである．

### 3.4 統合環境のシステム構成

統合環境はシステム構成として次の 2 層からなる．

1) 統合環境応用層……使用者の業務システムと直接インタフェースをとるサブシステムが含まれる層である．この層の一つのサブシステムが統合環境の最小単位である．

この層のサブシステムには，

① OLTP（OnLine Transaction Processing）サブシステム，

② データ交換サブシステム，

③ プレゼンテーション・サブシステム，

④ ネットワーク管理サブシステム，

⑤ システム運用管理サブシステム，

等が考えられる．

2) 統合環境共通層……応用層のサブシステムが共通に使用するサブシステムの集合体の層である．

この層のサブシステムは，

① データ通信サブシステム，

② ネーミング・サービス・サブシステム，

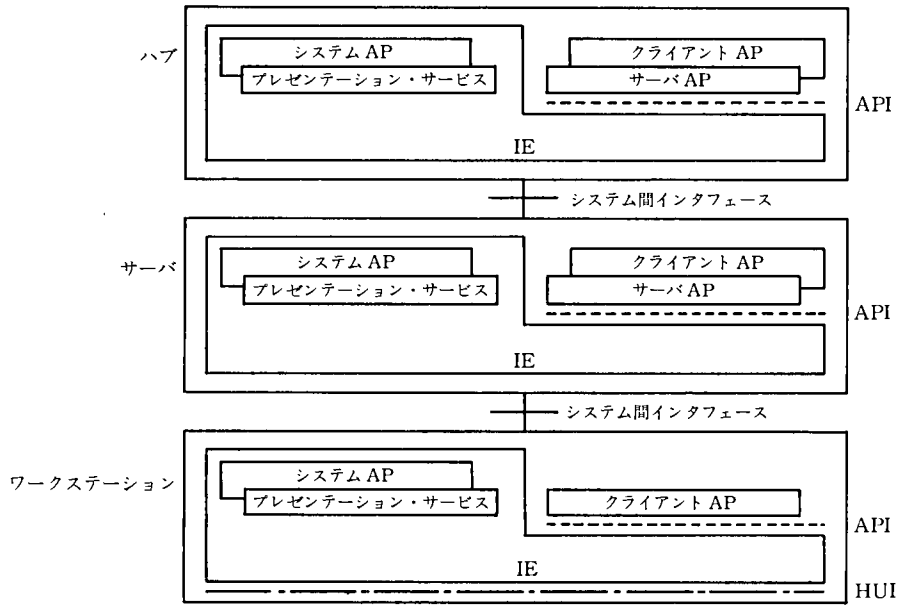



**図 13　統合環境におけるインタフェース**

Fig. 13　The interfaces implemented in integrated environment

③　セキュリティ・サブシステム，

等から成る．

上記のような環境のもとで，データ交換処理環境を構築することを考えてみる．

データ交換システムは，

①　メッセージ・ハンドリング・サブシステム，

②　リモート・プロセス間通信，サブシステム，

③　分散ネーミング・サブシステム，

に分かれると考えられる．

①はデータ交換システムそのものを特徴づけ機能定義する．上記1）の②に対応する．②，③は①を構築する上で必須の基本機能であり，2)の①，②に相当する．つまり，サブシステムの集合体として一つの環境が構築されることになる．このようにして構築された環境の上に，使用者の業務システムを作り込む構造とすることにより，システム全体の統合性，拡張性，再構築性を確保することができよう．

## 4. おわりに

本稿では，マイクロ系システム(UNIX サーバ，MS OS/2 ワークステーション)に DDA を実装した経緯，およびその内容について論じた．DDA は，マイクロ系システムにおけるシステム基盤ソフトウェアのあり方について一石を投じたと言う意味で価値があるものと考える．マイクロ系システムの性格の変化（汎用機化）によるシステム基盤ソフトウェアの必要性はさらに高まるのは必須である．

しかし，今後はシステム基盤ソフトウェアにとどまらず，システム共通基盤ソフトウェアの提供が求められるであろう．それが，後半で述べた「統合環境」である．近

い将来，業務システムの設計・開発者は，その業務のみに専念しシステム下位層については意識する必要がなくなるだろう．そういう透過性の高い環境の提供をめざして，プロダクトの開発を手がけていくつもりである．

---

**参考文献**　［1］ 日本ユニシス（株），シリーズ2200・1100　統合データ交換システム DDA 解説書，1991年1月，pp. 1-1～pp. 1-19.
［2］ 日本ユニシス（株），U 6000シリーズ　統合データ交換システム DDA 解説書，1991年3月，pp. 1-1～pp. 2-20.
［3］ サイエンス社編集部，分散コンピューティング環境 DCE の全貌，Computer Today，サイエンス社，1991年7月，No. 44，pp. 17～pp. 45.
［4］ 日経 BP 社編集部，UNIX 新時代，日経コンピュータ別冊，日経 BP 社，1990年7月.

**執筆者紹介**　北　川　達　朗（Tatsuro Kitagawa）
昭和28年生．52年埼玉大学理学部数学科卒業．同年日本ユニシス（社）入社．証券・金融機関でフィールドの SE サービスに従事した後，1100シリーズ系のオンライン・インフラ・プロダクトの企画・開発を経て，マイクロ・システム系のインフラ・プロダクトの企画・開発に従事．現在，システム技術本部システム企画開発部に所属．

# ワークステーションによる簡易入力支援機能

## A Support System Supporting Easy Data Entry by the Workstation

千葉啓善

**要約** CDOS*ワークステーションであるDS 7E上で稼働する，ローカルヘルプと名付けた簡易入力支援機能について報告する．

このローカルヘルプは大別して二つの機能を持っている．一つは，ポップアップキーの押下によりカーソルの位置から判断して対応する入力データの名称をウィンドウを開いて表示し，スクロールキーとリターンキーによって選択されたデータのコード部分を入力するコード入力支援機能である．もう一つは，カーソルが漢字を入力すべきフィールドにある時にキー入力が行われると自動的にかな漢字変換を起動し，フィールドを抜けると変換モードをオフにする漢字入力支援機能である．またこのローカルヘルプはUTS (Universal Terminal System: Unisysシリーズ2200•1100標準端末エミュレータ) のオウンコードとして作成し，UTS本来の機能を支援することもできるようになっている．

この機能によって実際にデータを入力するエンドユーザが，コードブックなしでコードを入力したり，漢字入力をあまり意識しないで漢字データを入力することができるシステムを実現している．

**Abstract** This paper reports on the easy data input support system named 'Local Help,' which operates on the CDOS-based DS 7E workstation. The Local Help software product provides two major functions as follows:

1) One is for code input support which serves to enter the code of selected data by the scroll key and the return key after the names of input data corresponding to the position of a cursor in the input field are automatically displayed, through window-opening, by depressing the pop-up key.
2) The other is for Kanji input support which serves to actuate an automatic Kana-Kanji conversion when there is a key input during the time the cursor stays in the field requiring Kanji input, and to turn off the conversion mode when the cursor goes out of the field.

The Local Help system, produced as UTS's own code, is also so created as to provide support for UTS's inherent capabilities. This support system has enabled end users to enter codes with no codebook referred to, and enter Kanji data without making operators too conscious of Kanji data input.

## 1. はじめに

ワークステーションの機能拡充による費用対能力比は年々向上している．処理速度の向上や接続可能媒体の増加は著しく操作性の向上に寄与していると考えるが，反面データの入力方法を考えた場合，ハードウェアの性能に比べ操作が簡便化されているとは言えない．本稿ではデータ入力簡便化を目的とした“ローカルヘルプ•システム”の概要を紹介する．

本システムは，花王(株)の戦略的情報システムの一つである“新ECHOシステム

* CDOS (Concurrent DOS)：米国デジタル・リサーチ社の登録商標である．

(Echo of Consumer's Helpful Opinions)”において初めて全面的に採用された入力支援システムであり，多くの支援機能を有し，汎用化を考慮した設計になっている．

データの入力を行う場合，入力すべき事項を操作の簡略化と情報量の圧縮のために操作者がコードに変換し，決められた体系のコードで入力している．操作者のコード変換の手助けとしてコードブックや操作マニュアルが準備されていることが多い．本システムはコードブックをワークステーション側のファイルに収納し，必要とするコード表を自由に参照できるようにしている．グラフィカル・ユーザ・インタフェース (GUI) として，マンマシン・インタフェース (MMI) 向上のためにウィンドウ分割を行っている．また参照データのウィンドウ表示定義と，参照コードの関係や処理記述部は，外部パラメータとして独立しており汎用化がなされている．

コードブックの配布に相当する作業はファイル転送機能で対応し，最新情報が送付されるように送達管理サブシステムにより更新情報の管理が行われている．

“新 ECHO システム”では，顧客（一般消費者）からの問い合わせ，苦情データを入力する際や，蓄積されたデータを検索・参照する際にもローカルヘルプが利用されている．

当システム作成に際しての基本方針は“操作性の向上”であり，対象者は業務知識の少ない人から熟練者までと考えており熟練度に応じた支援システムになっている．

## 2. ローカルヘルプの目的

ローカルヘルプは以下の目的で開発された．

1) コードを知らなくてもコードブックなしで入力できる．
2) 漢字属性のデータ入力操作を簡単に行うことができる．
3) 入力されたコードの正当性の検査ができる．
4) 操作を簡略化し，指1本でも操作を可能とする．
5) 入力画面，コード等の変更に対して容易に保守ができる．

これらの目的を実現するために，本システムでは入力されるデータを以下の9種に分類した．

① 年月日，年齢，数量等のようにキー操作で入力するもの
② 男女，有無等のようにコードの数が少く，入力画面のコメントで対応できるもの
③ 入力フィールドとコードが1：N で対応していて，コードに階層構造のないもの
④ 入力フィールドとコードが1：N で対応していて，コードに階層構造があるもの
⑤ 入力フィールドとコードがM：N で対応していて，コードに階層構造があり，複数入力が必要なもの
⑥ 入力フィールドとコードがM：N で対応していて，コードに階層構造があり，一括入力が必要なもの
⑦ 入力されるデータが漢字であるもの
⑧ 入力されるデータが半角カナであるもの

⑨ 姓・名のように，読みを入力したら，ふりがなと漢字の両方を入力したいもの

上記9種のデータの分類をもとに実現方法を検討した．

1) データ分類の①②については従来通りの入力とする．

2) データ分類③～⑥のコード入力の支援は，データ名称を一覧表として入力画面の一部にウィンドウを開いて表示し，その名称を選択するだけでコードの入力が行われ，しかもその名称も入力画面上に表示する．また，入力者がコードを知っている場合を想定して，入力フィールドでコードを直接入力することも可能とした．

3) データ分類⑦⑧のかな漢字入力フィールドでは自動的にかな漢字変換プログラムを起動し，フィールドを通過した時点でかな漢字変換プログラムを自動的に終了させることとした．半角カナはローマ字入力を採用した．

またデータ分類の⑨は，この機能に併せてふりがな項目への入力で漢字の姓・名候補をウィンドウに表示し，選択による入力を可能とすることとした．

4) データ分類の③～⑥のコードは，階層構造になっていないものはSAMファイルとして，階層構造になっているものはISAMファイルとしてワークステーション上に作成することとした．

また，目的④⑤の実現方法として以下のことを検討した．

5) 姓名と郵便番号・住所を変換するために専用の辞書を作成し，選択操作が容易にできるようにする．

6) 入力フィールドの位置，属性やウィンドウの位置，大きさ等はワークステーション側に入力画面ごとに定義ファイルを設け，ローカルヘルプのプログラムとは分離する．

7) ワークステーション上のSAMファイル，ISAMファイル，定義ファイルおよびワークステーションのプログラムは，ホストからファイル転送にて配信できることとし，一括管理を可能とする．

## 3. ローカルヘルプの機能

ローカルヘルプの機能を入力の支援という視点から整理すると図1のようになる．以降，その機能の詳細を述べる．

### 3.1 SAMヘルプ

SAMヘルプとは，入力データのコードブックにあたるワークステーション上のファイルがSAM形式であるものを言う．ファイルはコードとコード名称を一つのレコードとしており，レコードのならびはコード順でなくてもよい．ワークステーションはローカルヘルプ機能の立ち上げの際にSAMファイルをすべてメモリ上に読み込んでおき，検索時には先頭レコードから逐次検索を行う．これによってSAMファイルの読み込みにIO処理が不要となり，処理効率を上げている．しかしメモリに読み込むため，SAMファイルのレコード件数には一定の制約がある．

図2はSAMヘルプの実行例である．

ここでは「県」という入力フィールドに「14」を入力している．図の流れのように

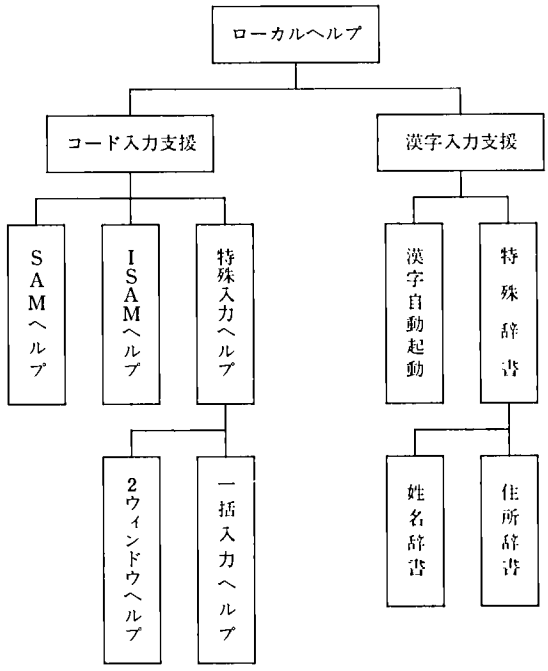



**図 1　ローカルヘルプの機能体系**

Fig. 1　Functional overview of "LOCAL help"

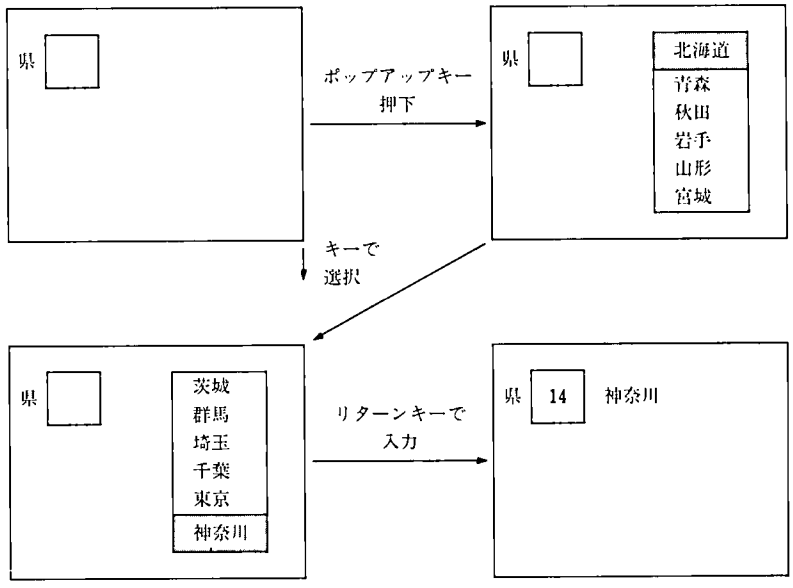



**図 2　SAM ヘルプの実行例**

Fig. 2　Excution example of "SAM help"

入力もできるし，テンキーにて「14」と入力することもできる．後者の場合はすぐに右下の画面となる．ポップアップキーの押下からウィンドウの表示までと，リターンキーの押下からウィンドウのクローズおよびデータの入力までの時間は，0.5秒以下であり，ウィンドウ内のスクロールも50件程度のデータ名称を全部スクロールしたとしても4秒程度でできるため，入力スピードが低下することはない．しかしウィンドウの行数は大きく取っても20行程度のため，100件を越えるようなコードについてはスクロールが速すぎて目が追いつかなかったり，目が疲れるといったことからSAMヘルプでの選択はむずかしいと思われる．

現在，新ECHOシステムのSAMファイルのレコード件数は平均約25件となっている．このため先に述べたレコード件数の制約には抵触していない．このレコード件数が多い時の対策として，キーを分割してウィンドウに表示する人数を絞り込むことは可能である．

### 3.2 ISAMヘルプ

ISAMヘルプとは，入力データのコードブックにあたるワークステーション上のファイルがISAM形式（索引付順編成）であるものを言う．これによって階層的に体系づけられたコードの入力を支援することができる．ISAM形式のファイルを扱うため，ISAMドライバを開発した．このISAMドライバは最大四つまでのISAMファイルを扱うことができるようになっており，もし五つ以上のISAMファイルを扱う場合には，ISAMドライバを二つメモリ上にロードすれば可能となるように設計されている．

現在の新ECHOシステムでは，ISAMヘルプドライバが扱っているISAMヘルプファイルは業務の関係で，多くて3個である．ISAMヘルプファイルを検索するためのキーは四つまで持つことができる．キーは画面上に表示されたフィールドであり，通常はSAMヘルプの対象項目である．キー項目に入力した後にポップアップキーを押下すると，キーによって絞り込まれたデータ名称がウィンドウに表示される．もしキーによって1件に絞り込まれた場合には，ウィンドウを開かずに（つまりリターンキー押下なしで）フィールドに入力される．ISAMファイルで扱うコードの件数は上位のキーで階層的に絞り込むように整理されていれば，1万件程度の件数には対応できる．

図3はISAMヘルプの実行例である．

この場合は氏名が，部，課のコードをキーとしてISAMファイルとなっている例である．また氏名コードがはじめからわかっている場合には，部，課は入力せずにに氏名を直接コード入力し，その入力された氏名コードから逆にキー項目である部，課のコードを自動的に入力することも可能である．

### 3.3 2ウィンドウヘルプ

2ウィンドウヘルプは，入力フィールドとコードがM：Nで対応していて，コードが階層構造であり，複数入力が必要な入力データのために開発された．たとえばある人の保有している資格を入力することを仮定する．入力項目は資格であるが，入力フィールドは複数必要となる．しかも資格には運転免許，情報処理，簿記，中小企業診断士等々複数のジャンルの違うものがある．そこで，第1ウィンドウに運転免許，情

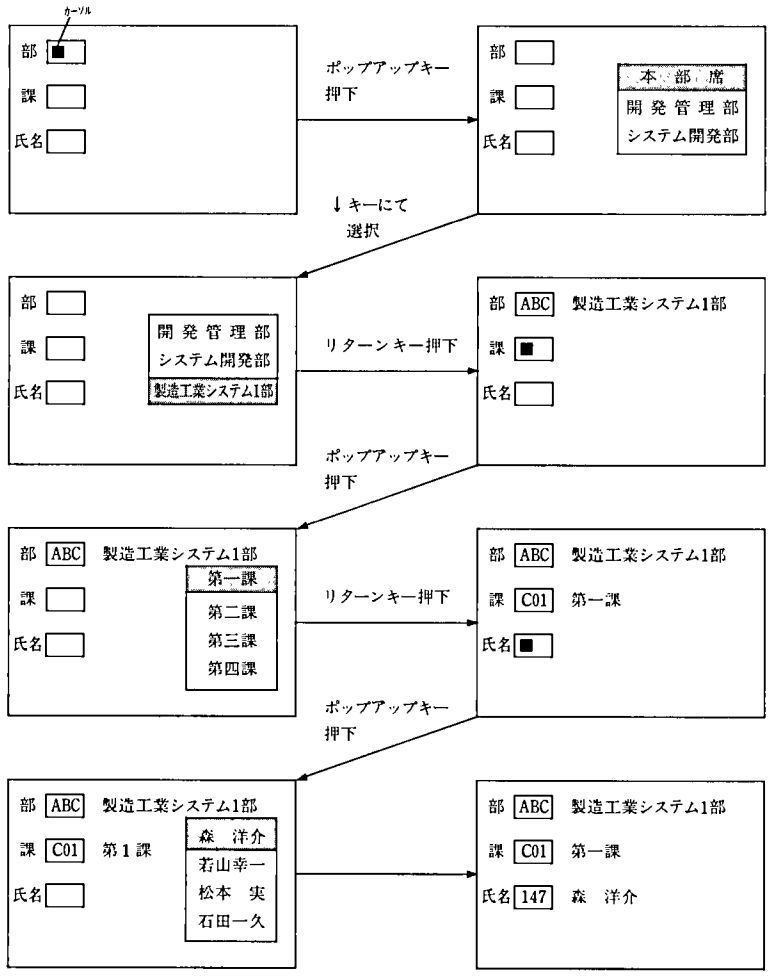



**図 3　ISAM ヘルプの実行例**

Fig. 3　Execution example of "ISAM help"

報処理のようなジャンルを表示し，リターンキーによってジャンルが選択されると，第2ウィンドウに資格名を表示することとした．この第2ウィンドウで資格名を選択しリターンキーを押下すると，入力フィールドに資格コードを入力する．またウィンドウは複数入力がやりやすいように，第1，第2ウィンドウともに選択しても開いたままとしている．このため別のジャンルの資格を入力したい場合には，取消キー（ブレークキー）を1回押下して第2ウィンドウを閉じて第1ウィンドウでジャンルを再び選択した後に第2ウィンドウで資格を選択し入力する．入力終了時には取消キーを2回押下し二つのウィンドウを閉じればよい．このような入力方式の場合，まちがえてリターンキーを2回続けて押下してしまうことが考えられるので，ダブリ入力はできないようにしている．図4は2ウィンドウヘルプの実行例である．ある人が情報処理試験の1種と日商簿記2級の資格を持っていた場合にこのように入力する．

### 3.4　一括入力ヘルプ

一括入力ヘルプは入力フィールドとコードがM：Nで対応していて，コードが階層

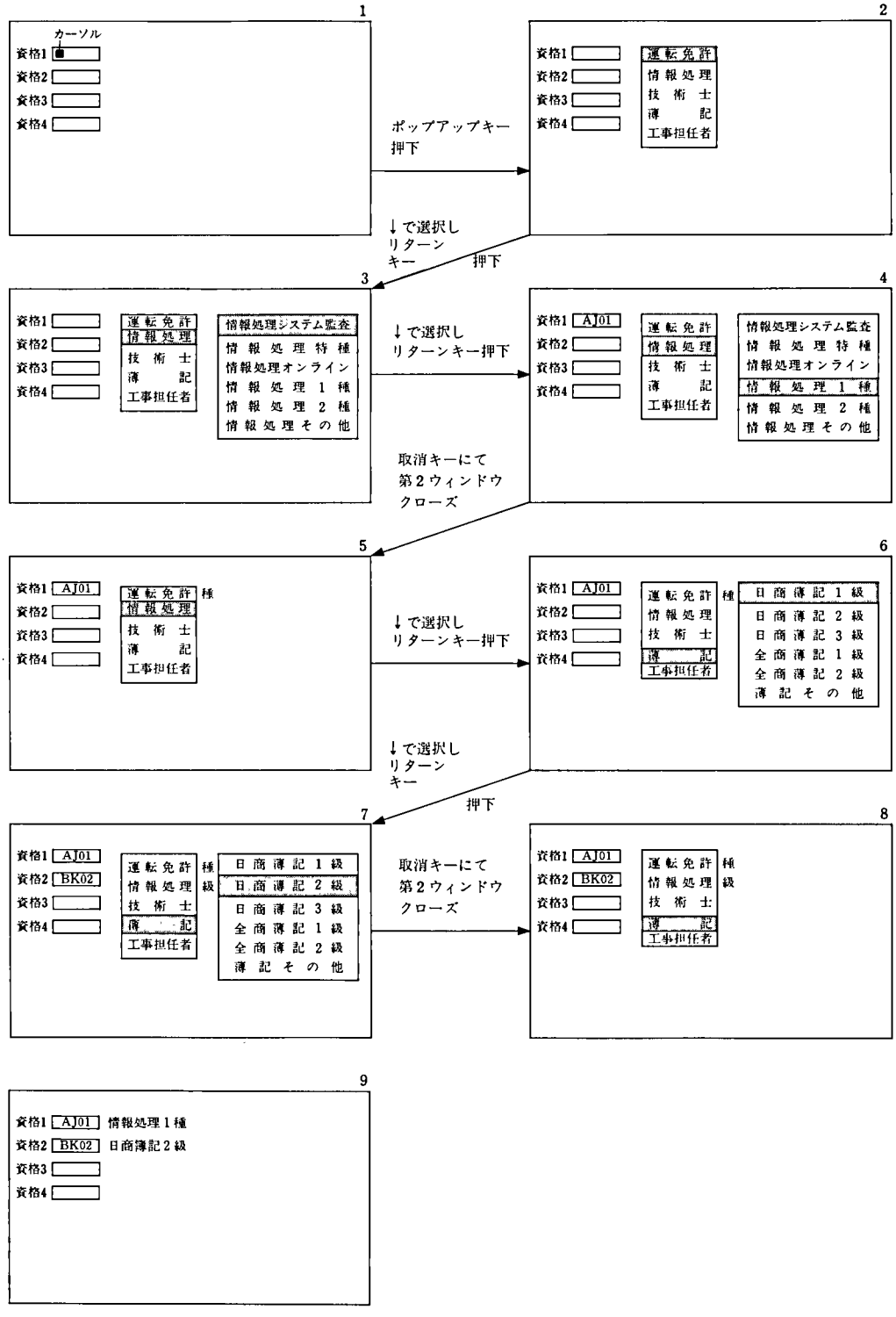



**図 4　2ウィンドウヘルプの実行例**

Fig. 4　Execution example of "2 WINDOW help"

構造であり，一括入力が必要な入力データのために開発された．これはいくつかの入力項目が他のいくつかの入力項目をキーにパターン化される時に効果がある．この一括入力ヘルプでは漢字の文字列も入力することができる．

### 3.5 漢字自動起動

この機能は，入力フィールドが漢字項目の場合に自動的にかな漢字変換を起動し，フィールドを通過するとかな漢字変換を自動的に終了させる．ヘルプ定義ファイルの入力フィールドの設定により，ローマ字入力漢字変換と，ローマ字入力半角カナ変換が選択できる．これによって，入力者は漢字変換キーと取消キーの二つのキータッチを省略できる．

漢字入力フィールドで漢字変換が自動的に起動されるため，入力者はデータの入力後，変換キー（タブキー）か英数キー（PF 1 キー）かカタカナキー（PF 3 キー）のいずれかを押すだけでよい．また，フィールドを抜けると自動的に漢字入力モードが終了するので取消キーを押す必要もない．ただし，漢字入力モードの起動と終了には若干の時間がかかってしまうので，もし漢字入力フィールドに何も入力しないでスクロールキーで通過しようとした場合に，カーソルの動きが少し止まってしまう．そこで漢字自動起動は，漢字入力フィールドで最初のデータ入力があった時とすることとした．

これによってスクロールキーで漢字入力フィールドを通過する時の動きに問題はなくなったものの，今度は最初のデータキーを入力後，漢字が起動されるまでの 1 秒弱の間に次のキーが入力されると，最初のキー入力が無効となってしまうことになった．つまり「か」と入力しようとして「KA」と続けて入力すると「あ」と入力されてしまうので，「K. A」と一呼吸おいて入力することが必要なのであり，現在これは使用上の制限事項としている．

### 3.6 特殊辞書

特殊辞書とは U-Micro JASTY（連文節のカナ漢字変換プログラム）のメイン辞書とユーザ辞書の他に，姓名変換用と郵便番号住所変換用の辞書を持ち，ウィンドウに表示し選択できるようにする機能である．これによって姓名だけのウィンドウ，住所だけのウィンドウの中から選択することができるため，選択操作が容易になっている．図 5 は姓を変換するヘルプの例である．この機能では，1 回の入力で漢字とふりがなを同時に入力することが可能となっている．

入力者はセイという入力フィールドにローマ字でデータを入力すると半角カタカナで入力される．もちろん漢字変換キーを押す必要はなく，直接ローマ字入力できる．そしてリターンキーを押下すると，相当する漢字がウィンドウに表示され，該当するものを選択しリターンキーを押下すると，姓という入力フィールドに漢字の姓が入力される．もし，該当するものがない場合は取消キーを押してウィンドウをクローズすると，カーソルが姓の入力フィールドへ行き，入力があると漢字変換モードとなるので，漢字の姓を入力する．名についても同様である．

図 6 は郵便番号から住所に変換するヘルプの例である．

このヘルプでは郵便番号を入力し，リターンキーを押下すると（郵便番号が 5 桁入力された場合はリターンキー不要）ウィンドウが開かれて，その郵便番号に対応する

市町村名が表示される。そしてリターンキーにより一つを選択すると住所1の入力フィールドに市町村名が入力され，入力された漢字データの直後にカーソルが位置づけられる．入力者は地番や部屋番号を入力する．住所1，2，3の入力フィールドは漢字自動起動になっている．

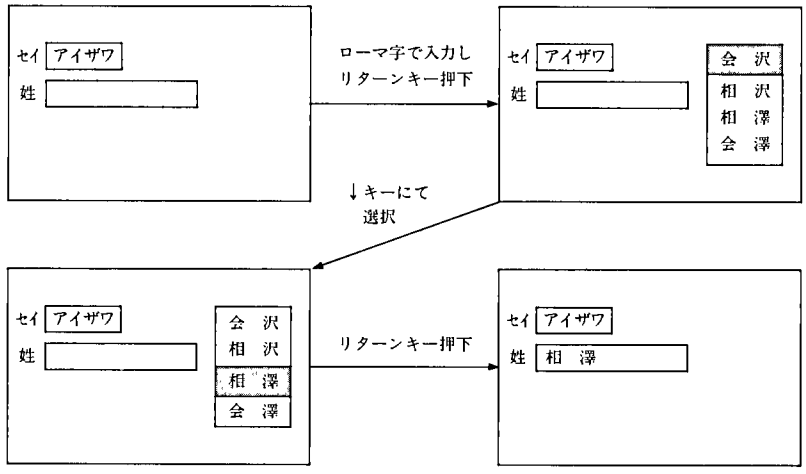



**図 5 姓変換ヘルプの実行例**

Fig. 5 Execution example of "Kanji name conversion help"

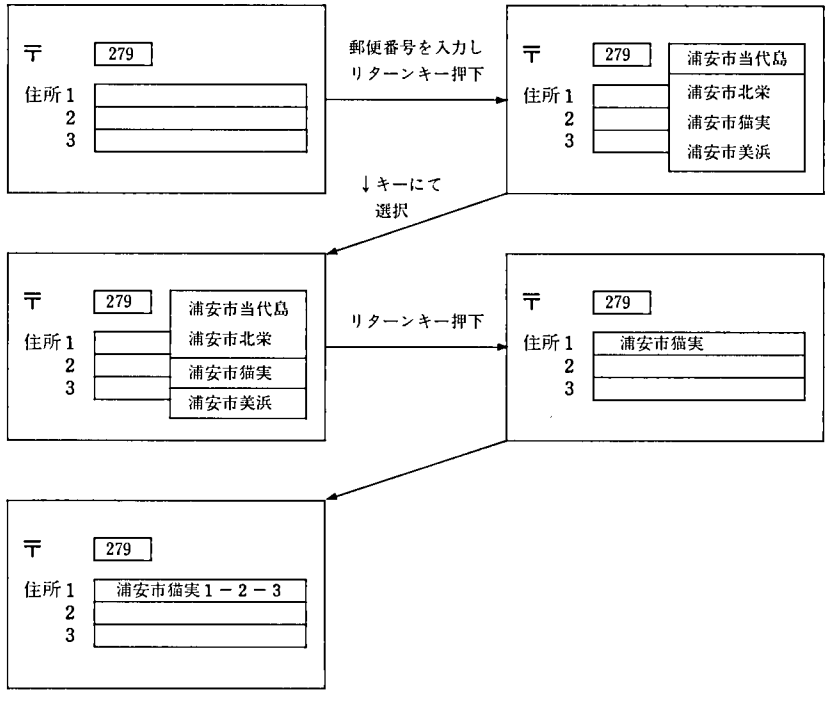



**図 6 郵便番号/住所変換ヘルプの実行例**

Fig. 6 Execution example of "zip code to address conversion help"

### 3.7 コードブックの更新

ローカルヘルプではワークステーション側にファイルを持ち，これがコードブックとして機能する．そこでコードブックの更新についての取り決めが必要となる．もしローカルヘルプの稼働するワークステーションの数が少なければ，コードブックファイルの配布を人手によって行うことも可能であるが，最新のコードブックが配布されている必要があり，コードブックの管理はホスト側で集中管理することとした．最新のコードブックはローカルヘルプの起動時に1日1回だけファイルの更新(ホスト側)の有無を自動的に判定した上でファイル転送を実行するような仕組みを開発した．またエラーを想定して，実際のファイル名とは異なるファイル名で受信し，受信完了後にファイルの保守を行うこととした．ISAM ファイルは，ワークステーション側で作成している．またファイル転送ではコードだけでなくヘルプ定義ファイルとワークステーションのプログラムも転送の対象としている．ファイル転送の実際の仕組みについては後述する．

### 3.8 ヘルプ定義

ローカルヘルプでは，入力画面の追加・変更時の保守を考慮してヘルプ定義ファイルを設け，プログラムとは分離した．ヘルプ定義ファイルとは各入力画面ごとにフィールドの位置，大きさ，属性，ヘルプウィンドウの位置，大きさ等を定義したレコードから構成されている．ヘルプ定義の詳細については5章にて述べる．

## 4. ローカルヘルプの適用事例

3章ではローカルヘルプがどのような機能を持っているかについて述べたが，本章ではローカルヘルプを全面的に採用した花王(株)の戦略情報システムの一つである新 ECHO システムの概略を述べる[1]．

花王(株)では花王生活科学研究所（以下生活研と略す）を昭和46年に発足させて，マーケットの声を商品に反映させるための窓口機能を持たせている．

新 ECHO システムは，消費者相談業務を支援するシステムである．生活研には年間約4万件の相談が寄せられ，商品に対する問い合わせや苦情等に対応している．相談の約70パーセントが電話で寄せられるため，相談窓口担当者は受話器を持ったまま何百種類もある商品に関する相談に対応している．相談窓口担当者は，消費者からの苦情や問い合わせに対して“正確・迅速・親切”に対応することによって消費者の信頼を得，また消費者の生の声を正しく企業活動に反映するためのセンサとしての機能をも担っている．このような相談窓口担当者を支援し，集められた消費者の声を企業内に正しく伝えることが新 ECHO システムの使命である（図7）．

### 4.1 新 ECHO システムの概要

本システムは，相談窓口システムと相談結果の解析システムの二つのメインシステムから構成される．相談窓口システムは，窓口担当者を支援するサブシステムと相談結果を入力するサブシステムから成っている．窓口担当者を支援するサブシステムには，商品情報検索サブシステムと氏名検索サブシステムがある．

商品情報検索サブシステムは，DS 7E ワークステーションとトスファイル*および

* Tosfile（光ディスクファイリングシステム）は(株)東芝の登録商標である．

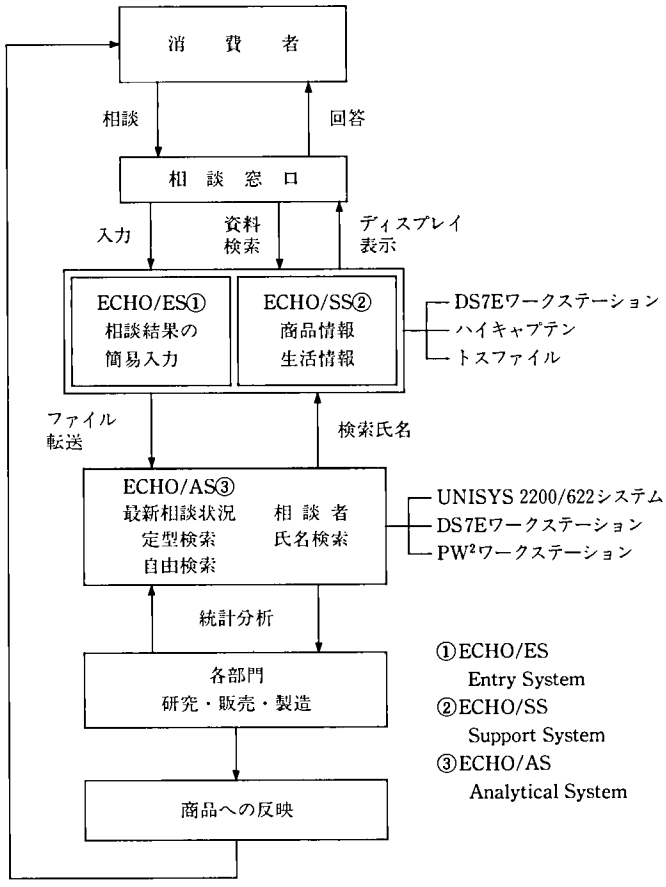



**図 7　新 ECHO システムの概要**

Fig. 7　Outline of "New ECHO system"

ハイキャプテン* というメーカもメディアも違う 3 種類のハードウェアを結合し構成している．商品に関する情報をモノクロ文字情報とカラー自然画情報に分類整理して登録しておき，DS 7 E ワークステーションからのローカルヘルプを使った商品コードの入力により，ハイキャプテンのディスプレイに商品の外観をカラーで表示する．DS 7 E ワークステーションはハイキャプテンに画像番号を送出すると共に，その商品に関してトスファイルに登録されている文字情報の目次をディスプレイに表示する．窓口担当者がトスファイルの文字情報を検索したい時には，目次をスクロールキーとリターンキーで選択すると，該当する書類番号がトスファイルに送出される．窓口担当者は，ハイキャプテンのディスプレイに表示されたカラー自然画情報と，トスファイルのディスプレイに表示されたモノクロ文字情報を参照しながら一般消費者からの相談に対応できるため，的確な対応が可能となっている．なおハイキャプテンにはタッチパネルを装備して検索を容易にしており，トスファイルの文字情報も DS 7 E ワークステーションのディスプレイ上で再度目次を選択したり頁めくりをすることができるようになっている．

* Hi-Captain（自然画ビデオテックスシステム）は日本電信電話(株)の登録商標である．

もう一つの氏名検索サブシステムは同一の相談者が複数回の相談をする場合を考慮し，過去の相談の内容を相談者の氏名の前方一致で検索するホストリアルシステムである．この応答時間は約2，3秒程度である．これらのサブシステムはDS 7 Eの一つのキーボードの操作で実行され，しかも電話対応中ということを考慮して指一本でも操作ができるようになっている．

相談結果を入力するサブシステムは，ローカルヘルプの機能を十分に使って簡易な入力システムを構築し，コードブックレス，マニュアルレスを実現した．入力されたデータはDS/LANを通してサーバよりホストへファイル転送される．また，とくにイメージデータを入力したい場合には，入力した相談結果を印書し定められたエリアにイメージ情報を書き込んだ上で，サーバに接続されたイメージスキャナから入力する．転送された入力データはホスト側の業務バッチで処理され，翌日にはデータベースに反映される．入力されたデータの修正は，当日分についてはサーバのデータを修正してホストへ再送し，前日以前のデータについてはデータベースよりひとまとめにしてワークステーションにファイル転送し，入力と同等のローカルヘルプを使って修正できるようにしている．

次に相談結果の解析システムであるが，これは検索キーの入力にローカルヘルプを採用し，解析者がより解析業務に集中できるようにしている．また，解析手法についても長年のノウハウを生かしたパターン化した検索方法と，いろいろな角度から自由に検索できる方法を併用し，FACILE 1100（FACILity for End users 1100；意思決定支援システム）を利用して多種類のグラフが表示できるようにしている．もちろん原データを見ることもできる．

このように，相談窓口担当者を支援することによって消費者の信頼・支持・満足を得，しかも消費者の生の声を敏感にキャッチして正しく入力し，その入力されたデータベースを簡単に自由にアクセスできるようにしているのが新ECHOシステムである．

### 4.2 ワークステーション側のプログラム

本節では，ワークステーション上で稼働しているプログラムについて簡単に述べる．

これらのプログラムのうち相談窓口での入力，商品情報検索，氏名検索の各プログラムとハードウェアの関連を図8，図9に示す．

1) UKEENTRYプログラム……相談結果の入力プログラムであり，SAMヘルプ，ISAMヘルプ，漢字自動起動，特殊辞書，2ウィンドウヘルプ，一括入力ヘルプを提供している．ホストとは関係なく稼働できるため，画面やカーソルの動き等，利用者がもっとも使いやすいように作られている．入力されたデータはサーバにLANで転送され，サーバ側のプログラムによりホストへファイル転送される．

2) UKEEDITプログラム……ホストのデータベース上の相談結果を修正するために使用するプログラムである．使い勝手をよくするため，ホストとの間はリアルで相談結果1件を転送し，UKEENTRYと同等の機能を提供している．修正中，データベースをロックしなければならないため，ワークステーションで10分のタイムアウトをチェックしている．

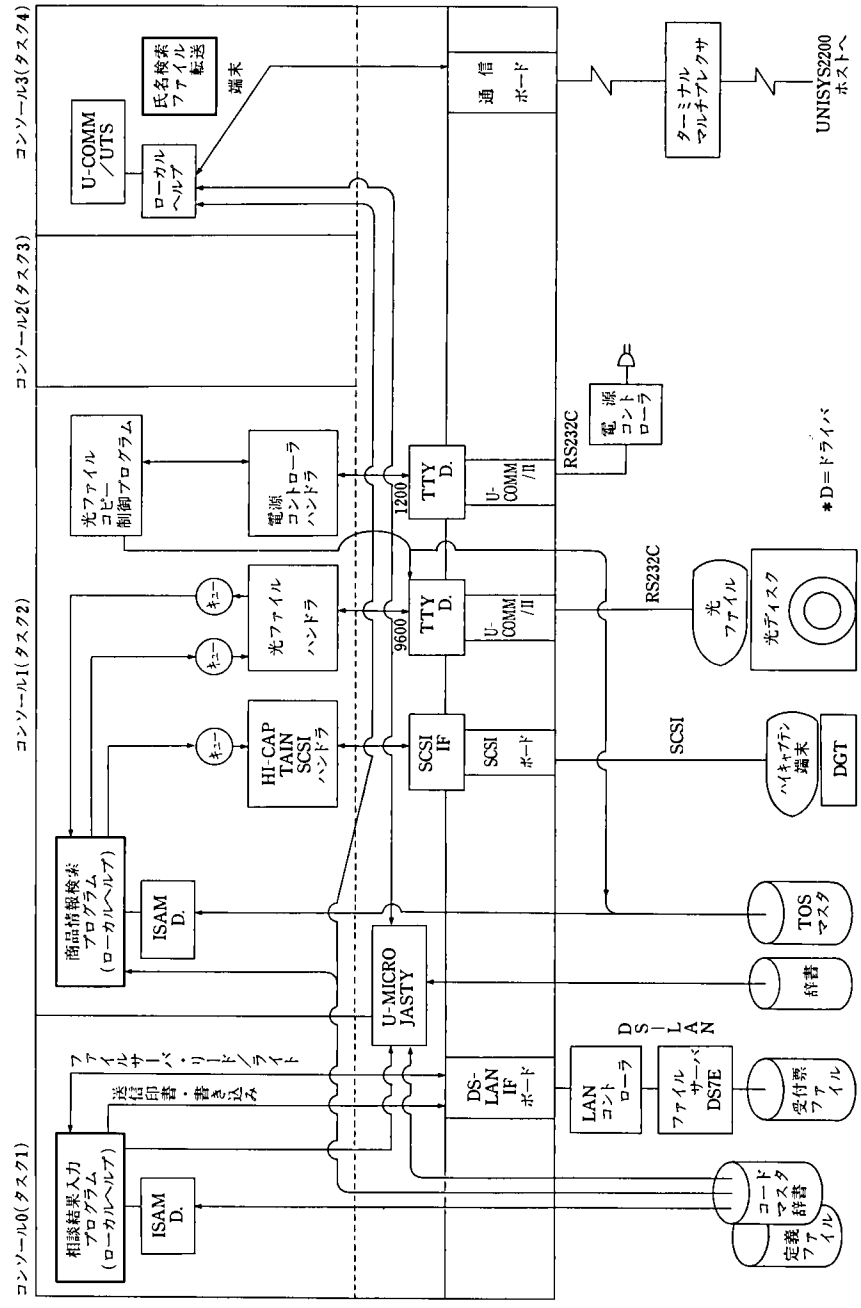



図 8 窓口システムの構成（窓口機）

Fig. 8 Configuration of NEW ECHO window system (window machine)

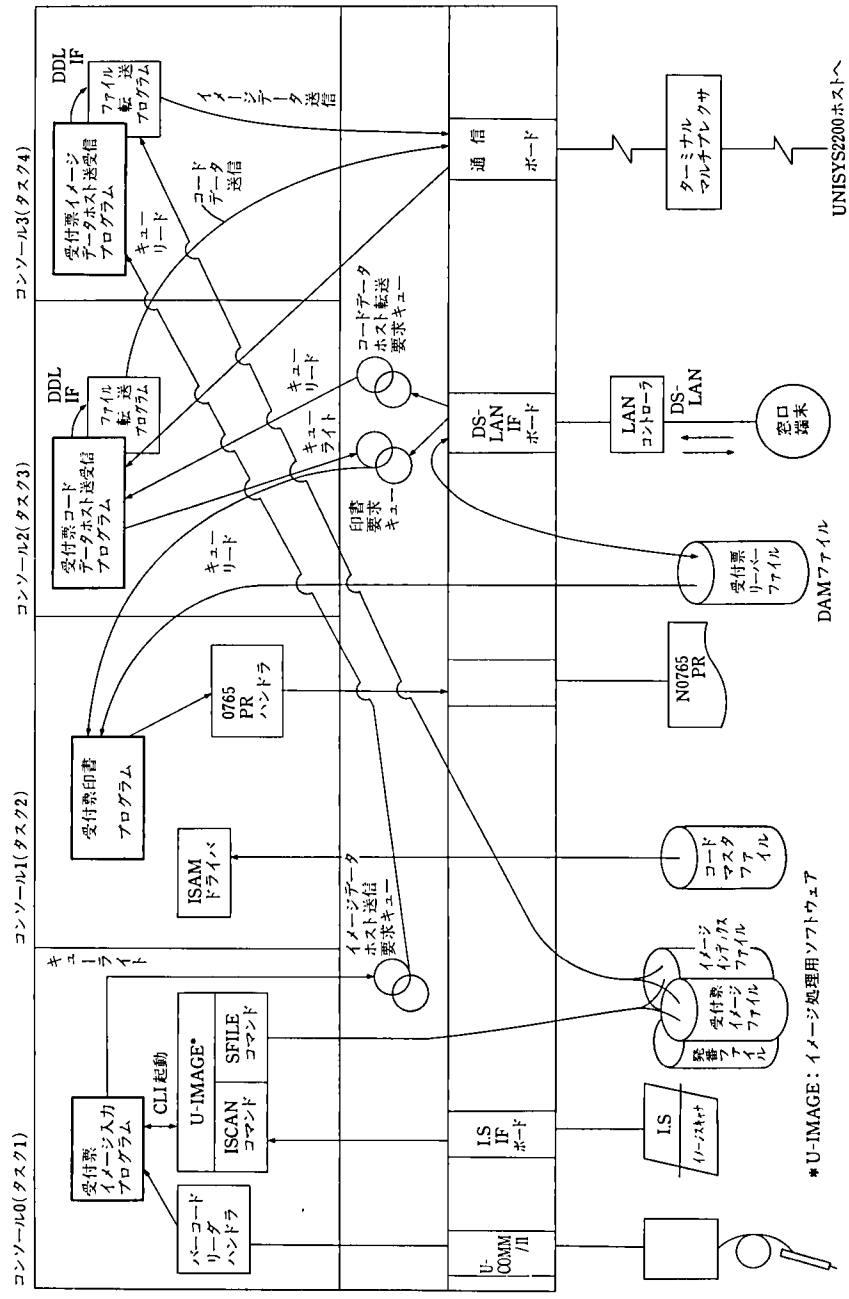



図 9　窓口システムの構成（サーバー機）

Fig. 9　Configuration of NEW ECHO window system (server)

3) ECHOBINF, TOSFILE, HICAPの三つのプログラムは商品情報の検索プログラムである.ECHOBINFはワークステーションに画面を表示し,商品コードの入力を受け付ける．このため，SAMヘルプやISAMヘルプを提供する．商品コードが入力されると商品外観の自然画の画像番号をHICAPプログラムに渡す．HICAPはSCSIインタフェースに画像番号を送出する．商品コードを入力してからハイキャプテンでの画像の表示開始までは約1秒以内である（表示終了までには6〜8秒程度かかる)．また商品コードが入力されると，トスファイルの索引ファイル（ワークステーション側に持っている）を検索し索引を表示する．表示した索引をスクロールキーで選択させ，リターンキーの押下により光ディスクの索引に対応した書類番号をTOSFILEプログラムがRS 232 Cインタフェースを経由してトスファイルへ送出する．送出後約4秒でトスファイルのディスプレイに画像が表示される．

4) LHSHIMEIプログラム……相談者氏名検索プログラムであり，U-COMM/UTSエミュレータ*のオウンコードである．氏名入力は自動ローマ字カナ変換であり，SAMヘルプも提供している．起動時にはファイル転送プログラムを起動し，最新のコードブック等の情報をホストから受信させている．

5) ECHOLHプログラム……U-COMM/UTSエミュレータのオウンコードであり，ECHO/ASの解析システムへの検索キーの入力をローカルヘルプにより提供するシステムである．SAMヘルプ，ISAMヘルプ，漢字自動起動，2ウィンドウヘルプ，特殊辞書を提供する．起動時にはファイル転送プログラムを起動している．

## 5. ローカルヘルプの実現方法

本章では，ローカルヘルプのソフトウェア構成，各種機能実現のためのヘルプ定義ファイルの詳細，およびマスタファイル更新のためのファイル転送の詳細について述べる[2]．

### 5.1 ソフトウェア構成

ローカルヘルプには,UKEENTRYのようにワークステーション側のプログラムが作成した画面上で入力を支援するものと,ECHOLHやUKEEDITのようにホストから送信された画面上で入力を支援するものの2種類があり，後者はU-COMM/UTSエミュレータのオウンコードとして作成されている.図10は後者のソフトウェア構成である.前者については図10からホストとU-COMM/UTSエミュレータを取り外したものと考えることができるので，ここでは後者のソフトウェア構成について述べることにする．

図10のHELPドライバはSAMヘルプを提供する．入力フィールドの設定はホスト側から送られてくる画面ごとにヘルプ定義ファイルで行う．画面の認識はホスト側とワークステーション側の取り決めによって行うが，画面上の定められた位置にユニークな文字を入れることによってできる．ワークステーション上のローカルヘルププ

* UCOMM/UTS: UTS (Universal Terminal System) 4000通信プロトコルに従って行われる通信制御，画面制御，利用者プログラムとのインタフェース，周辺装置の入出力制御の部分を遂行させるエミュレータである．

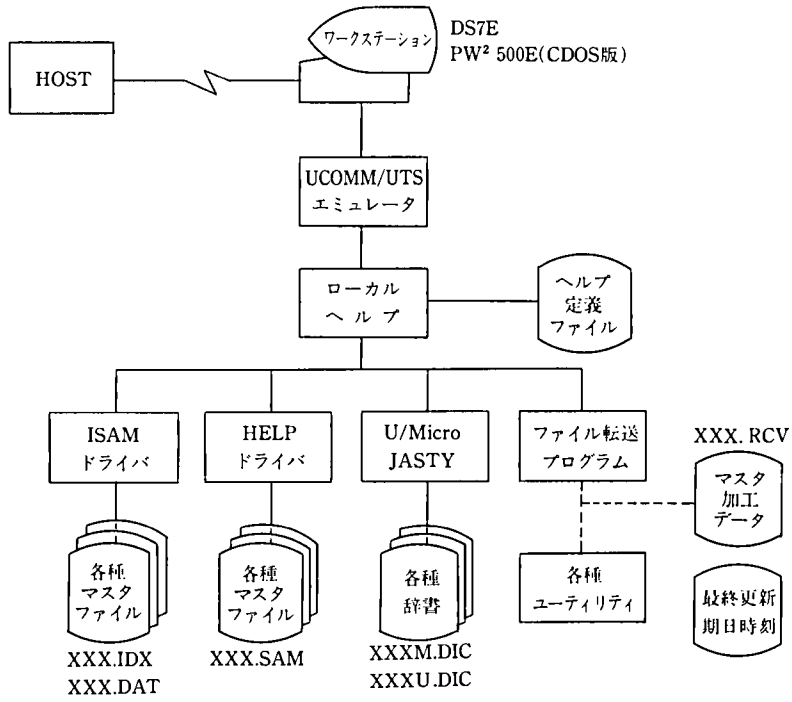



**図 10　ローカルヘルプのソフトウェア構成-1**

Fig. 10　Software configuration of "LOCAL help"-1

ログラムは，ホストから受信する画面データの中で定められた位置に存在するユニークな文字によって，ヘルプ定義ファイルの該当画面の定義を使用しローカルヘルプ機能を実行させる．ヘルプ定義ファイルについては，次節にて詳細を述べる．次にローカルヘルプとベーシックソフトウェア，ハードウェアとの関係を図 11 に示す．

図 11 の左上のキーボード入力は BDOS コールにより行われ，入力データの画面表示は PROCESS CHARACTER 機能によりスクリーンに表示する．ローカルヘルプの選択結果であるコードデータは，この機能によって UTS エミュレータに渡されるが，エコーバックにより表示されるデータ名称は BDOS 機能により表示されるだけなので，ホストに送信されることはない（図の左端のルート）．

ファンクションキーが入力された場合には，ポップアップキーを除いて FUNCTION KEY PROCESS 機能により，UTS エミュレータに渡される．コミュニケーションについては，ローカルヘルプは UTS エミュレータで使用中の回線パスワードを，GET PROCESS 機能により入手し，CSX 経由でデータを受信する．受信したテキストは，PROCESS STRING 機能により UTS エミュレータ経由で画面表示が行われる．

なお対応するインタフェースが存在しないため，UTS エミュレータは POLL を検出できないので，インディケータラインへの表示も行われない．そこでローカルヘルプがこの機能をシミュレートして，インディケータラインにローカルヘルプの ID とレベルを 6 桁で表示している．ローカルヘルプと UTS エミュレータの間では，電文のデータ部のみの受け渡しを支援するので BELL 受信の受け渡しもできない．

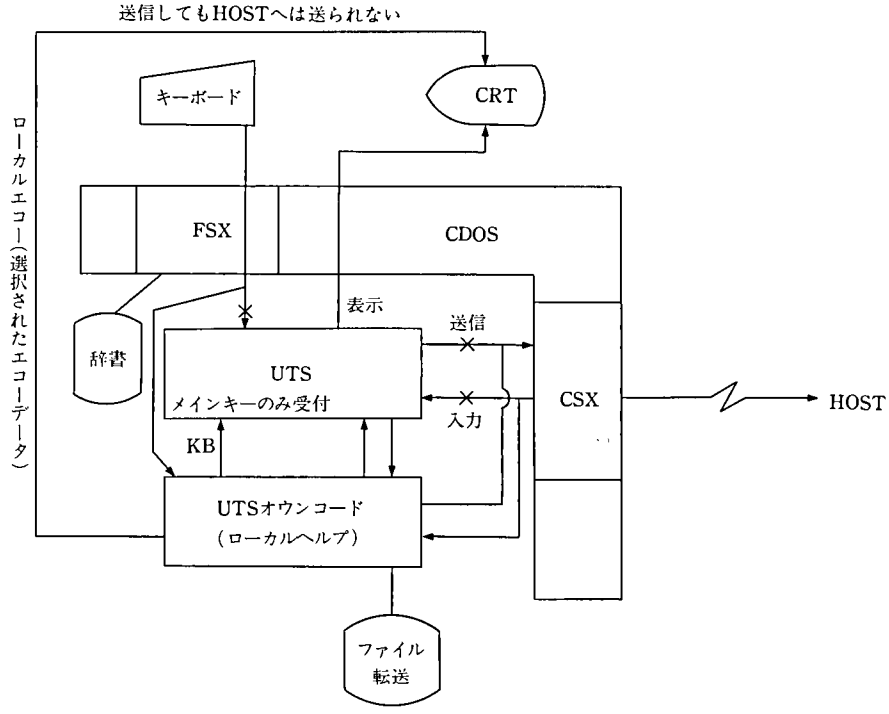



**図 11　ローカルヘルプのソフトウェア構成-2**

Fig. 11　Software configuration of "LOCAL help"-2

## 5.2　ヘルプ定義ファイル

ローカルヘルプは，ワークステーションでの入力簡易化のために各種機能を提供している．そのためには画面ごとにどのフィールドに，どのようなローカルヘルプを実行させるかという指定をしなければならない．そこでまず，ホストから画面を送信するたびにその画面のヘルプ機能の指定も送信することを検討してみた．しかしこの場合 DPS（Display Processing System）や FACILE 等の画面インタフェースを特別に機能追加しなければならず，現実的ではない．

次にワークステーション側でローカルヘルプ・プログラム自身に，画面ごとのローカルヘルプの指定を組み込むことを検討した．しかしこれもホスト側で画面の修正があったり，あるいは新しく画面を追加したりする時にローカルヘルプ・プログラムの改造が必要となり，柔軟性のあるシステムとは言えなくなる．そこで，ローカルヘルプの指定をローカルヘルプ・プログラムからヘルプ定義ファイルを設けることにより，分離するという方式を採用した．こうするとホスト側の画面送信に特殊な機能を追加することなく，また画面の修正や追加についても分離されたヘルプ定義ファイルを修正，追加することにより対応することができる．

ヘルプ定義ファイルとは，ローカルヘルプの各種機能を画面ごとに設定したファイルであり，一つの画面に一つのヘルプ定義ファイルが対応し，ワークステーション側に持つ．

このヘルプ定義ファイルはホスト側の画面設計等のアプリケーション・プログラム開発に制約を課すことはない．ただ一つワークステーション側がどの画面を受信した

のかを判断するための画面識別番号については，ホストとワークステーション間で取り決めた上で画面上の定位置に表示するように，ホスト側で画面作成することになっている．ホスト側で画面設計が終了すると，その画面の入力フィールドと出力フィールドおよびレイアウトを確認して，ヘルプ定義ファイルにローカルヘルプの定義をする．これはワークステーション側のエディタで作成し，ホストへファイル転送しホスト側配信ファイルへ書き込む．この新しいヘルプ定義ファイルは，翌日以降に各ワークステーションでローカルヘルプを起動する際にファイル転送される．なおこのファイル転送は，ホスト側での新画面の導入と同期をとらずに先行してリリースしてもよい（ただし画面修正の場合には同時にリリースすることが必要）．

以下に定義ファイルの仕様について述べる．

1) ヘルプ定義ファイルのファイル名……ヘルプ定義ファイルは，ローカルヘルプがホストから受信した画面ごとに自動的にアクセスするため，ファイル名にローカルヘルプの識別子（ID）と画面の識別番号を組み込み，拡張子でヘルプ定義ファイルであることを表示する．図12にファイル名の例を示す．

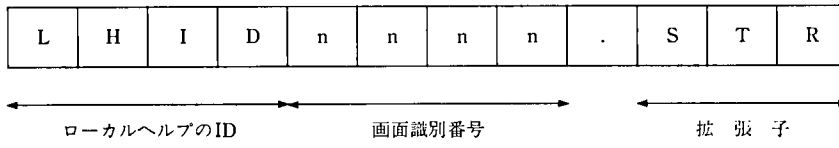



**図 12　ヘルプ定義ファイルのファイル名**

Fig. 12　File name of "LOCAL help" definition file

ヘルプ定義ファイルはCDOS標準テキスト形式で，1レコード80文字以内の可変長ファイルである．

ヘルプ定義ファイルのレコードフォーマットを図13に示す．なおレコード内の「；」文字以降はコメントとして扱う．

2) フィールド名称……ローカルヘルプが扱う画面内のフィールドの名称は1桁目は「F」固定で，残りの2桁は01～99の数字で指定する．フィールド名称は画面ごとにその中でユニークであればよい．フィールドは必ずしも入力フィールドであることはない（エコーバックフィールド）．

3) フィールド定義……フィールドの大きさを何行目の何桁目から何文字分という形式で指定し，ヘルプ属性をタイプ（I：ISAM，S：SAM，K：かな漢，空白：ヘルプなし）と番号で表示する．ヘルプ属性とは，どのコードブックを使うか，どのような漢字変換を提供するかを示す．

4) ウィンドウ定義……ヘルプでデータ名称を表示するウィンドウの大きさを何行目の何桁目を開始位置として何行分，何桁分というサイズで指定する．

5) キー定義……ローカルヘルプでコードブックを参照するためにキーが必要な場合（ISAMヘルプ等）に本定義で指定する．エコーとは，キーとして指定したフィールドに入力しないで直接このフィールドにコードが入力された時に，キーフィールドにも該当するコードを入力することを指し，'E' と入力があると実行す

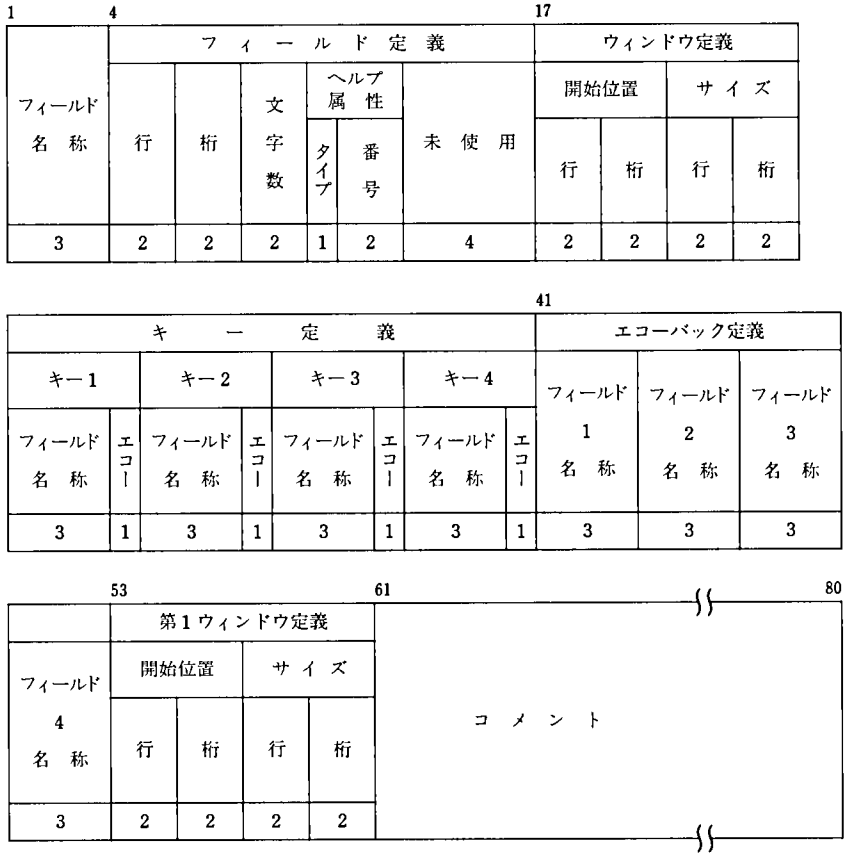



図 13 ヘルプ定義ファイルのレコードフォーマット

Fig. 13 Record format of "LOCAL help" definition file

る．たとえば図3で部・課に入力しないで氏名の入力フィールドに直接コードを入力した場合，部・課の入力フィールドにそれぞれの該当コードを入力することをいう．キーは4個まで指定できる．

6) エコーバック定義……このフィールドにコードが入力された場合に，そのデータ名称を表示するためのフィールドを指定する．このフィールドも4個まで指定できる．

7) 第1ウィンドウ定義……2ウィンドウヘルプの第1ウィンドウのウィンドウ定義をする．定義方法はウィンドウ定義と同様である．

ヘルプ定義ファイルのレコードは必要項目のみを指定し，コメントは必要に応じて付加する．たとえばエコーバックフィールドの定義は，フィールド名称とフィールド定義のみ指定すればよい．

このヘルプ定義ファイルがローカルヘルプの外に出ていることによって画面追加や変更等の際に，ローカルヘルプのプログラムの変更が不要になり，ヘルプ定義ファイルの追加，変更だけで対応できることが，ローカルヘルプの大きな特徴となっている．

## 5.3 ファイル転送

ファイル転送とは，ローカルヘルプがワークステーション側で使うコードブックとしてのSAMファイル，ISAMファイルとヘルプ定義ファイルおよびローカルヘルプ自身を含むプログラムファイルの更新や追加が発生した時に，ホストから各ワークステーションへ自動的に転送し，ワークステーション側で使用できるように変換する機能を言う．漢字辞書については，データ量が多いことと修正がほとんどないことからファイル転送の対象とはしていない．

図14はファイル転送の概念図である．

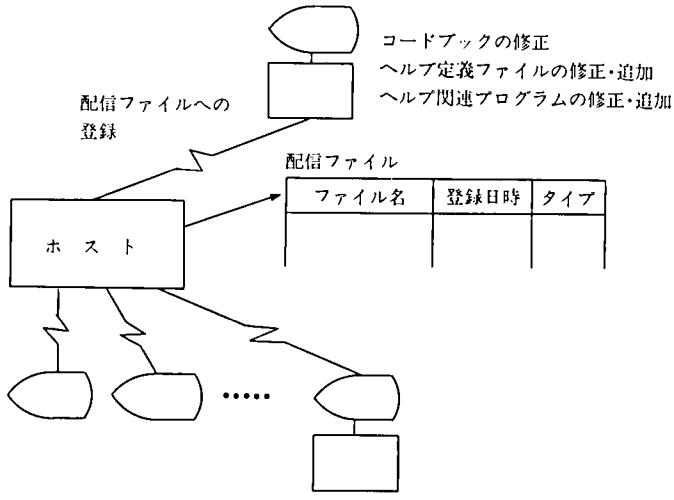



**図 14　ファイル転送の概要**

Fig. 14　Outline of file transmission for "LOCAL help"

コードブック，ヘルプ定義ファイルやヘルプ関連のワークステーションのプログラムに修正・追加があると，ホスト側配信ファイルにファイル名，登録日時，タイプ等が書かれる．タイプとはワークステーションが業務によっていくかのタイプに区分できる場合に，そのタイプごとに送信するファイル群を別にしたい時に使う．

たとえば検索専用のワークステーションには，データを修正するプログラムは送信しないようにすることができる．実際のファイル転送はワークステーション側がキックする．ローカルヘルプ・プログラム立ち上げの際にファイル転送プログラムを起動する．ファイル転送プログラムは，ファイル転送実行日時の入っているファイルと当日の日時を比較する．もし日付が同じなら当日にすでにファイル転送が実行されていると判断し，すぐにローカルヘルプ・プログラムを起動する．日付が違っていたり，実行日時の入っているファイルがない時はファイル転送実行要と判断し，最新登録日時の入っているファイルをアクセスする．そのファイルに入っている最新登録日時データをホストへ送信する．ホスト側は受信した最新登録日時データをホスト側配信ファイルの登録日時と比較し，登録日時の方が新しいファイルをリストアップする．そしてそのリストをワークステーションに送信する．

ワークステーションはリストに従って個々のファイルを受信し，受信が完了すると

ファイル変換プログラムを起動する．転送されたファイルは，転送中の障害を考慮してファイル名の拡張子が変えられている．ファイル変換プログラムは，これらのファイルをSAMヘルプファイルについては正規のSAMファイルにコピーし，またISAMヘルプファイルについてはこのプログラムで作成する．そしてファイル転送実行日時および，最新登録日時ファイルを更新し，ローカルヘルプ・プログラムを起動する．

このようにして1日に1回だけのファイル転送が自動的に行われ，コードブック，ヘルプ定義ファイル，ワークステーション・プログラムの一元管理を実現している．

## 6. おわりに

以上ローカルヘルプの機能を中心にして述べたが，最後にその長所と今後の課題について整理して，まとめとしたい．

1) 長　　所

① 入力を支援するための多種の機能を有している．

今回紹介した各種機能で完璧であるとは言えないが，現実に操作マニュアルとコードブックを必要としないシステムを実現した．(新ECHOシステムでは50画面以上でローカルヘルプを利用し効果を上げている．)

② ヘルプ定義ファイルによる汎用性

ヘルプ定義ファイルを採用したことにより，ホストシステムの変更や機能追加による画面の修正，追加にも柔軟に対応することができる．

2) 今後の課題

① コードブック指定のプログラムからの分離

現在は，コードブックにあたるSAMファイルやISAMファイルのファイル番号やファイル名称およびそのファイルのレコードフォーマット情報については，ローカルヘルプ・プログラムに組み込まれている．このため，ローカルヘルプを他のアプリケーションで別なコードブックで作成しようとしたり，現在稼働中のローカルヘルプに新しいコードブックを追加しようとすると，ローカルヘルプ・プログラムの修正が必要になる．今後は上記指定をヘルプ定義ファイルのようにローカルヘルプの外に出して指定できるように検討したい．

② MS-DOSへの展開

ローカルヘルプはDS 7 EのCDOSの下で開発した．そのためPW²ワークステーションのCDOS環境では動作するが，MS-DOS*やMS OS/2*の下では稼働しない．今後各種ワークステーションへの展開を考えた場合，まずMS-DOSでの開発を検討したい．

末筆ながら本稿執筆にあたり，花王(株)システム開発部，生活科学研究所の皆様に助言をいただいたことを記し謝辞としたい．

---

* MS-DOS, MS OS/2は米国マイクロソフト社の登録商標である．

**参考文献**　［1］ 岡崎静子,「フレンドリーなメディアミックスネットワークシステムの開発―新ECHO SYSTEM―」, ユニシス研究会 平成2年度春期全国大会入選論文.
［2］ 内藤隆志, ローカルヘルプ端末プログラム機能仕様書　1, 2版.

**執筆者紹介**　千　葉　啓　善 (Hiroyoshi Chiba)
昭和29年生. 48年東京工大附属工高電気通信科卒業. 同年日本ユニシス(株)入社. シリーズ90, シリーズ1100の保守を経て, 63年より製造業ユーザのシステム開発, サービスに従事. 現在 製造工業システム1部に所属.

# FINESSE-J の特質と操作性 ——カスタマイズの視点から

# The Characteristics and Operability of FINESSE-J ——From a Viewpoint of Customization——

天 野 剛

**要 約** FINESSE-J の保守モジュールであるカスタマイズ・モジュールは，FINESSE-J の環境変化に対する柔軟性を典型的に示すモジュールである．

このモジュールが示す柔軟性は，取引を各構成要素に分解し，各々をカスタム・ファイルとして再構築することによって，各取引ごとの形態的変化をプログラムロジックから分離させた結果可能になったものである．

しかし，それらのファイル群の連関構造が，トップダウン式のアクセス方法をとっていることから，操作上の利便性と視認性にやや課題を残す結果となっている．

本稿では，この課題をワークシートの作成とその利用によって解消しようと試みている．エンドユーザ的視点から，既存のソフトウェアパッケージをいかに使いやすいものにしていくかを志向した結果である．

**Abstract** The "customized module" which serves as the maintenance tool for the FINESSE-J financial solution package is typically indicative of the fact that the package is highly flexible to changes in the financial systems environment. The high flexibility provided by the module has only become possible as a result of the new adoption of the procedures to break up the attributes of each transaction and then to rebuild them into customized files; that is, by isolating non-normal types of transactions from the program logic. Yet, the relational structure for such files admits only downward access, resulting in some damaged convenience and visibility in workstation operation for transactions.

The author understands that it requires software approaches to get rid of such a shortcoming, but, as stated in this paper, he has tried to give a solution through the use of data sheets for the customized module as a challenge to make the existing software package easier to use from an end-user standpoint.

## 1. はじめに

FSA (Financial Systems Architecture) は，金融機関営業店の環境変化に柔軟に対応し，戦略的システムへの拡張性を持つ統合化された営業店システムの概念である．これを具現化しているのがワークステーションOSの"BTOS"，システムソフトウェアの"FSA-J"，アプリケーションソフトウェアの"FINESSE-J"である．

本稿では，B1000シリーズ金融機関勘定系パッケージ"ABOSII (Advanced Banking Online System II)"に準拠したFINESSE-Jの開発・導入時の経験をもとに，FINESSE-Jの持つソフトウェアパッケージとしての操作性を，その保守モジュールである「カスタマイズ・モジュール」をとりあげて評価する．さらに，評価の成果物として作成された「カスタマイズ用ワークシート」を事例として紹介する．

評価はエンドユーザの視点から，とくに操作上の利便性と視認性とに着目している．

ワークステーションソフトウェアの使いやすさを評価する上では，最も有効な視点と考えたからである．

ABOSII 版 FINESSE-J の開発・導入作業は，昭和 63 年 4 月から 11 月までの 8 か月間に及んだ．この時期に使用したハードウェアは，CPU が B26K であり，現在広く導入されている B39, B38/28-EXP シリーズとは，処理速度，メモリ容量，OS レベル，システムソフトウェアレベルの点で大きな差異が存在する．当然カスタマイズ・モジュールについても，当時のものと比較すると，プログラム的にその機能も改善されている．しかしながら，その変更は最小限にとどまっており，設計思想にも何ら影響を与えていない．したがって，B26K 版への評価はそのまま現行カスタマイズ・モジュールへの評価となり得る．

## 2. カスタマイズ・モジュールの位置付け[1]

### 2.1 FINESSE-J の構成

はじめに FINESSE-J におけるカスタマイズ・モジュールの占める位置について概観する．

FINESSE-J は，端末の機能をモジュールとして分化させており，①コントロール・モジュール，②テラー・モジュール，③カスタマイズ・モジュール，の 3 モジュールから構成されている*．

コントロール・モジュールは，マスタ・ワークステーション上で稼働し，クラスタリング機能**によって接続されるクラスタ・ワークステーションとのネットワーク管理（各クラスタ・ワークステーションとの論理経路管理），プリンタ・ファイル等の資源管理を行う複数のアプリケーション・プログラムの総称である．

テラー・モジュールは，クラスタ・ワークステーション上で稼働し，勘定系取引，非勘定系取引（照会取引等）の機能を実行するモジュールであり，金融機関営業店での第一線現金処理，第二線記帳処理，さらに両者を結合した OTM (Online Tellers Machine) 処理を行う入出力二つのアプリケーション・プログラムから構成されている．実際の営業店での窓口取引の大部分はこのモジュールが担当しており，一般に営業店端末の操作面での評価という意味では，このモジュールの問題となる．

カスタマイズ・モジュールは，上記の 2 モジュールとは異なり，開発・保守専用のモジュールである．このモジュールが保守対象とするのは，テラー・モジュールがパラメータとして参照する数種のカスタム・ファイルである．カスタム・ファイルにはさまざまな取引形態が定義されており，テラー・モジュールは端末操作に従ってこのファイルを参照し，取引制御を行う．

これら 3 モジュールの関連図を図 1 に示す．

### 2.2 FINESSE-J の柔軟性とカスタマイズ・モジュール

FINESSE-J の目指す環境変化に対応する柔軟性は，このような機能分化を目的としたモジュール構造によって実現されている．

---

* 次版の FINESSE-J (BTOSII 3.0 対応版) では 4 モジュールとなっている．追加されたのは「カスタマ・サービス・モジュール」で，金融機関からみた顧客，あるいは渉外担当者への情報提供を目的としている．1988 年の時点では，その概念のみが存在した．

** マスタ・ワークステーションと複数のクラスタ・ワークステーションとを BTOS 管理下で，RS422 インタフェースで相互交信する LAN の方法である．営業店ネットワークの基軸を構成する．

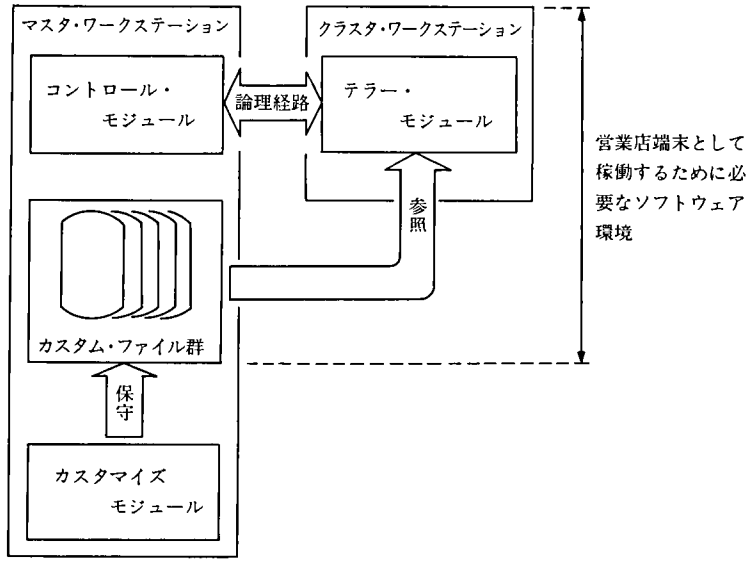



**図 1　FINESSE-J モジュール関連図**

Fig. 1　Modules of FINESSE-J

すなわち，コントロール・モジュールはネットワークおよび資源の管理機能を，テラー・モジュールは取引制御機能を，カスタマイズ・モジュールはシステム保守機能をそれぞれ分担することによって，各モジュール間の相互依存性を極力排している．とくに，テラー・モジュールとカスタマイズ・モジュールの機能分化には，カスタム・ファイルという“クッション”が置かれているのが構造的特色である．

この“クッション”は，取引形態の変化がテラー・モジュールに与える影響度合を最小限にしている．言い換えれば，すべての変化の要素をカスタム・ファイルとしてプログラムロジック（テラー・モジュール）から分離することによって，システム保守の容易性を高めているのである．

OTM 処理の変更，科目の追加，複雑な取引制御等を除けば，FINESSE-J の保守の大半は，テラー・モジュールの変更を必要とせず，カスタマイズ・モジュール単独で行うことができる．しかも，営業店端末における保守作業の大部分は，取引の追加や入力項目の追加・変更と考えられる．このような観点から見れば，FINESSE-J の柔軟性を具現化しているのはカスタマイズ・モジュールといってもいい．

## 3.　カスタマイズ・モジュールとカスタム・ファイルの構造[2][3]

次にカスタマイズ・モジュールとカスタム・ファイルの構造的特質について述べる．それらの設計思想は，これから行う評価の重要な視点を提供してくれている．

### 3.1　カスタマイズ・モジュールの構成

カスタマイズ・モジュールは，10 本のアプリケーション・プログラムと一つのユーティリティ・プログラムから構成されている(図 2)．

これらのプログラムは，その名称の通り，各々のカスタム・ファイルと対になっている．ここでも FINESSE-J の特徴である機能分化的構造が顕著に現われており，各

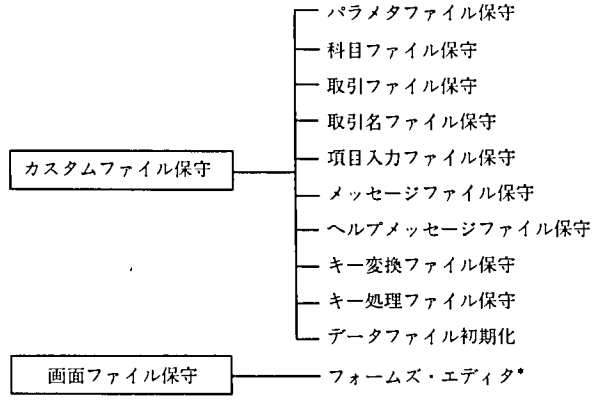
プログラムは該当ファイルの保守という単一の機能しか持っていない．しかも，各プログラム間相互のインタフェースもとっていない．

### 3.2 カスタム・ファイルの構造

カスタム・ファイルの構造的特徴は，科目コード・取引コードといった取引制御に必要なコード体系を，一度取引の構成要素として分解し，それらを一定の基準に従ってファイルとして再構成した点にある．一定の基準とは，取引を進めていく過程を段

表1　主なカスタム・ファイルと属性項目
Table 1　Custom files and its attributes

| カスタム・ファイル名 | 定義内容 | 主な属性項目 |
|---|---|---|
| 科目ファイル | 取引可能な科目の種類と，科目が許容する取引コードの範囲を定義する． | ・科目コード(キー)<br>・項目名<br>・取引順序番号 |
| 取引ファイル | 取引の処理パターンを定義する． | ・取引順序番号(キー)<br>・取引コード<br>・取引画面番号<br>・項目入力パターン番号<br>・取引名番号 |
| 取引名ファイル | 取引コードに対応する取引名を定義する． | ・取引名番号(キー)<br>・取引名 |
| 項目入力ファイル | 取引画面に対応するすべての入力項目とそのタイプを定義すると同時に，ホスト・システムへの伝送フォーマットを定義する． | ・項目入力パターン番号(キー)<br>・桁数<br>・文字タイプ番号<br>・編集タイプ番号<br>・入力値検査番号<br>・伝送上の位置 |
| 取引画面ファイル | フォームズ・エディタによって作成される．各取引に対応する画面ファイル | ・ファイル名そのものが RECORD 9999. FORM という形でパラメータ化されており，9999 に当たる部分が取引画面番号としてユニークになっている． |




* フォームズ・エディタは，取引画面を会話型で作成できる BTOS のユーティリティ・プログラムである．

図 2　カスタマイズ・モジュールのプログラム構成
Fig. 2　Programs of Customize Module

階的に眺める視点といってよい．少なくとも，ここには取引の流れに係わる“オペレータの視点”がなくてはならない．この基準が間接的に端末の操作性に影響を与えるからである．

表1は主なカスタム・ファイルの種類と属性項目の一覧である．これらのカスタム・ファイルが取引操作上，どのようにテラー・モジュールによって参照されていくかを示したのが図3である．この二つの図表から，各カスタム・ファイル相互の関連性は一方のカスタム・ファイルのキー項目が他方の属性項目として定義されることによって保たれていることがわかる．その関連性は取引操作の流れにそって，科目ファイル→取引ファイル→取引名ファイルというように方向性を持っている．

カスタム・ファイル間の方向性を図4に示す．

この方向性は，そのままカスタム・ファイルの設計の視点と置き替えることができる．この視点はさらに，テラー・モジュールのカスタム・ファイルの参照の視点をも反映している．このことから，カスタム・ファイルとして再構成される場合の基準とは，テラー・モジュールの取引制御の処理の流れに従っているといえる．

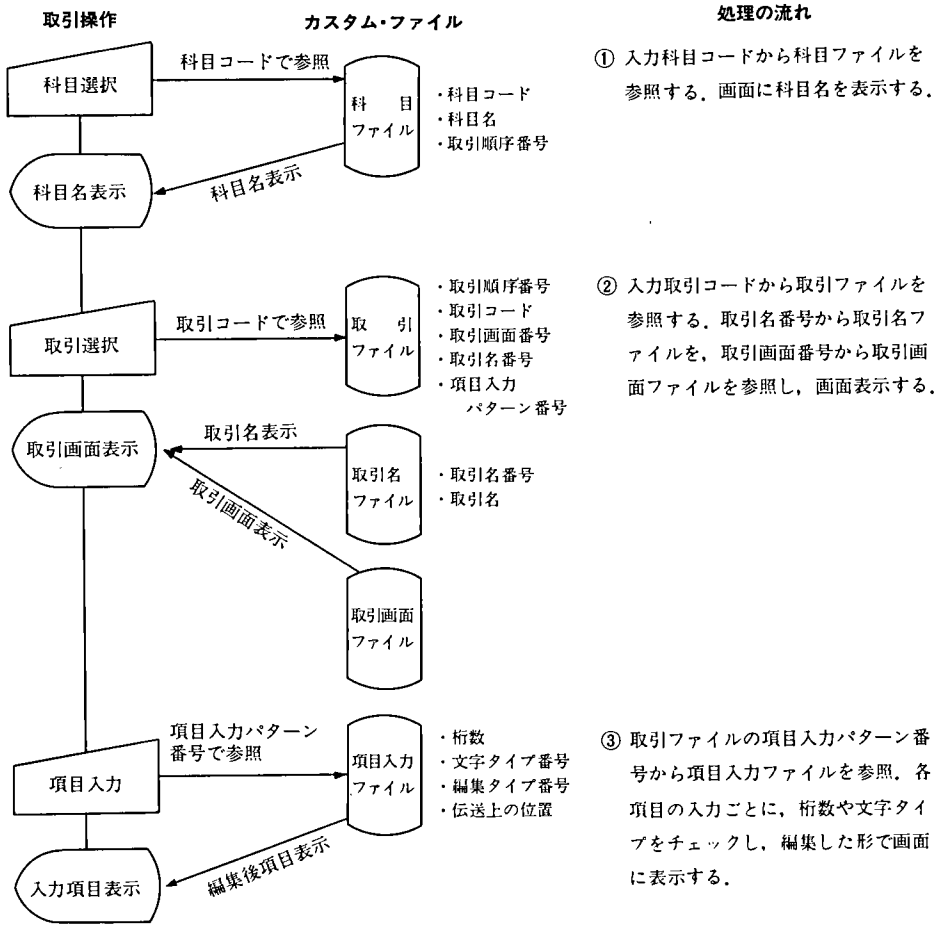



**図3　取引操作とカスタム・ファイル**

Fig. 3　Transaction operation and Custom Files

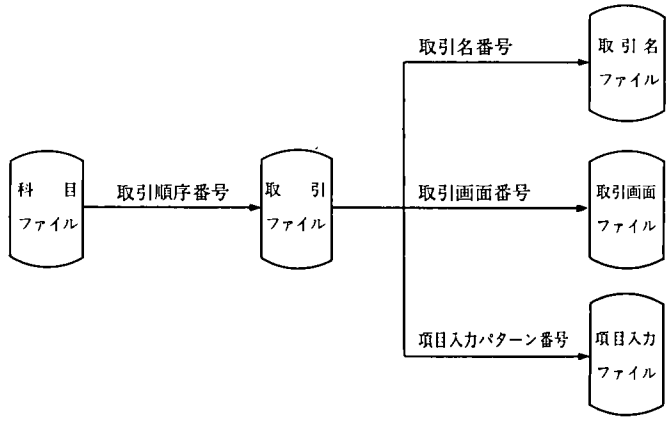



図 4　カスタム・ファイルの結合の方向性
Fig. 4　Combination dimension of Custom Files

さて，テラー・モジュールが参照しやすいカスタム・ファイルの構造は，この保守モジュールであるカスタマイズ・モジュールにとっても参照しやすい構造になっているだろうか．

前述したように，カスタマイズ・モジュールの各プログラムは，該当カスタム・ファイルのみを処理対象としている．また，図3と図4からも解るように，カスタム・ファイル自体も，参照できる方向性が特定化されている．ここから，カスタマイズ・モジュールの操作性に何らかの影響を与えていることがうかがえる．

この影響がどのような局面で現われ，どのような結果を生み出しているかを次章で明らかにする．

## 4. カスタマイズ・モジュールに対する評価

さて，カスタマイズ・モジュールの操作上の利便性と視認性について評価していくわけであるが，その前にまず評価基準について触れておく．

### 4.1 評 価 基 準

テラー・モジュールが操作上の利便性，視認性の点から評価されるのは，当該モジュールが FINESSE-J の中核となるモジュールであるところから当然であるが，同一評価基準がカスタマイズ・モジュールの評価に適用できるかという問題がある．

カスタマイズ・モジュールは，テラー・モジュールのように FINESSE-J の実稼働面から見れば，確かに評価対象とはならない．しかし，システム保守の面から見れば，それ自体が独立したソフトウェア体系と考えることができ，しかもユーザに解放されているという点から見ても，カスタマイズ・モジュールはテラー・モジュールと同一基準での評価が可能である．

### 4.2 カスタマイズ・モジュールの課題

カスタマイズ・モジュールが孕む操作上の課題は，3章で明らかにしたカスタム・ファイルの構造的特質から派生している．以下にエンドユーザがカスタム・ファイルの保守を行う場合を具体的に想定し，操作上の課題がどのような局面で現われるかを見

てみることにする．

たとえば，普通預金の「入金」取引に，あるコードの入力項目を追加したいと仮定する．修正者は，カスタム・ファイルの中で取引画面ファイルと項目入力ファイルだけを修正すれば望む結果が得られることは理解している．しかし，該当する取引画面番号と項目入力パターン番号は，図4（カスタム・ファイルの結合の方向性）にも示したように，科目ファイルから取引ファイルへと辿っていかなければ判明しない．

次に，修正者はフォームズ・エディタを使用して，「入金取引画面」（図5）内に「コード入力欄」を追加する．さらに，項目入力ファイルを修正するために，図6に示す項目入力ファイル保守画面を表示させる．この画面上のテーブルの一行は，取引画面の一つの入力項目に対応しているが，今回追加した項目が取引画面上何番目の項目であったかが解らなければ，この保守作業はできない．図6の画面には項目の順序番号表示がないため，修正者は，取引画面ファイルを修正する段階で記録をとらない限り，両ファイルの項目レベルでの相関関係を理解するのは困難である．

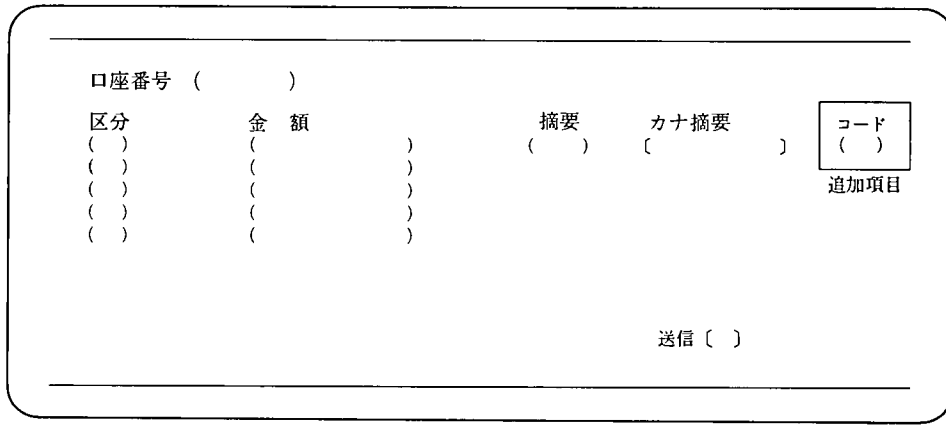



**図5 入金取引画面**

Fig. 5 Transaction screen

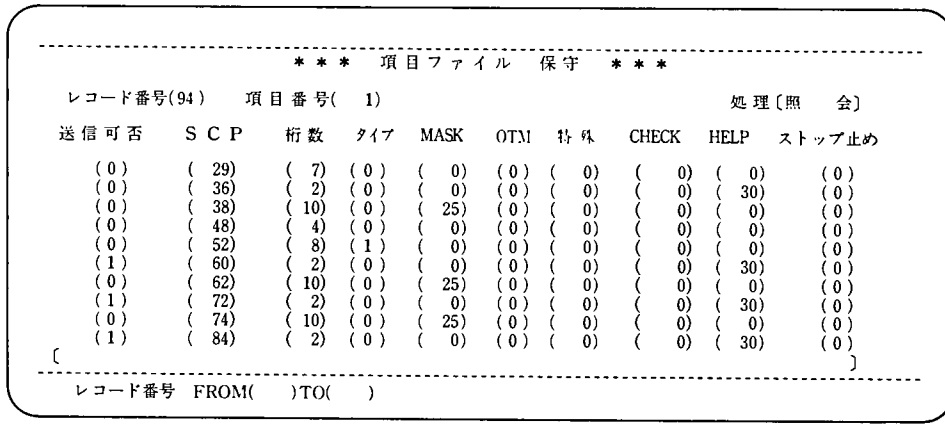



**図6 項目入力ファイル保守画面**

Fig. 6 Maintenance screen of "Koumoku File"

この例では，追加した「コード」は取引画面上6番目の項目であるから，項目入力ファイルのテーブルの6番目に属性定義項目を挿入すればよいが，修正はこのような単純な項目追加だけとは限らず，既存項目の変更や削除が同時にある場合等は，経験者といえども戸惑うことが多い．

このように，エンドユーザから見た場合のカスタマイズ・モジュールの課題は，次のように集約できる．

1) 各カスタム・ファイル間の関係は，トップダウン式にしか見ることができない．
2) 各カスタム・ファイルの項目レベルでの相関関係は保守画面からは判断できない．

この二つの課題は，ABOSII 版 FINESSE-J の顧客システム開発時にすぐさま直面した課題であり，解決しなければ作業過程に直接影響を及ぼすものであった．しかし，プログラムの改善を行うという根本的対処は実施しなかった．それは，工数増大や要員補充等の多くの問題を発生させるとともに，FINESSE-J そのものの仕様変更を伴うことが確実であり，パッケージソフトウェアを利用する利点を失う危険性を意味していたからである．

## 5. 評価の成果物

このようなカスタマイズ・モジュールの課題をいかに解決したかの方法論は，そのまま評価の成果物としての意味がある．本稿では，ワークシートの利用を回避策としてとったが，それにはあらかじめヒントがあったのである．

### 5.1 LINC からのヒント

カスタマイズ・モジュールの課題解決の最も簡易な方法は，各カスタム・ファイルの項目レベルでの相関関係を机上で参照できることである．この単純な回答へのヒントは，LINC (Logic & Information Network Compiler) 用のワークシートから得られた (図7)．

当社の 4GL の一つである LINC は，内部で使用するデータの名称や属性をデータディクショナリに登録し一元管理している．しかし，画面の設計過程では，画面上の項目とデータディクショナリ上に登録したデータ名を関連付ける作業は，やはり理解しづらく，カスタマイズ・モジュールと同質の課題を孕んでいた．そこで LINC においては，画面設計や項目定義用のワークシートを標準化して，デザインシートとして使用している．

### 5.2 カスタマイズ用ワークシート

LINC のワークシートをモデルとして，カスタマイズ用ワークシートを作成するにあたって留意した点は次の四点である．

① 記述の容易性
② 各カスタム・ファイルの連関性の明示
③ 各項目間の連関性の明示
④ 設計書として利用できること

これらの意図にそって作成したのが図8に示す「FSA 画面仕様書」である．上段に各カスタム・ファイルのキー項目となるデータ値を記述し，中段に画面レイアウトを，

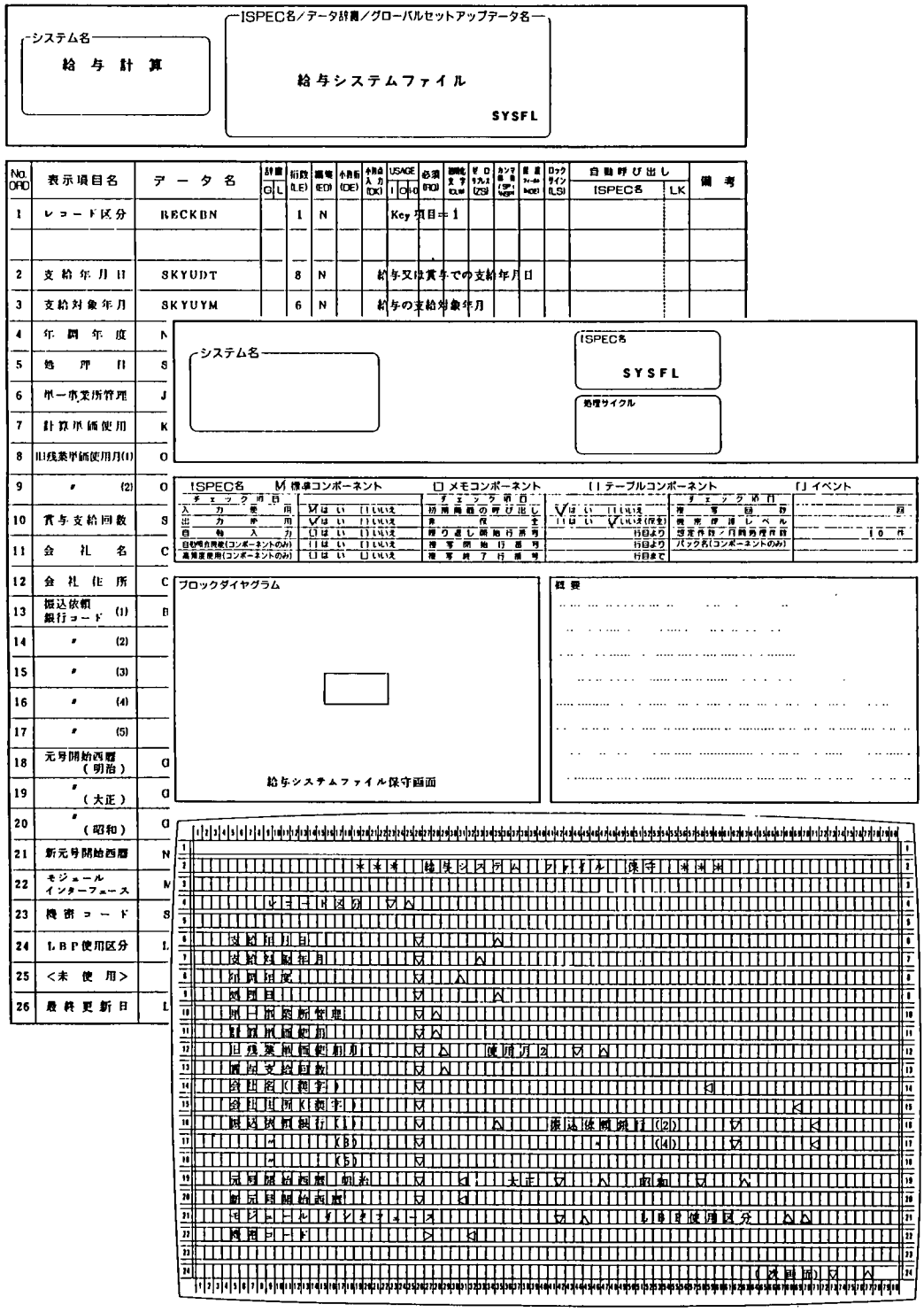


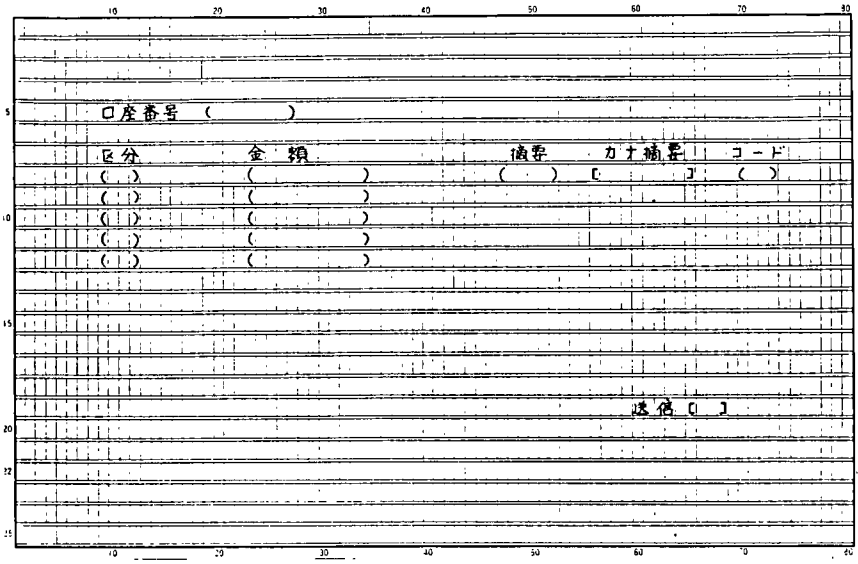

図 7 LINC 用ワークシート使用例

Fig. 7 Sample of LINC specifications

# FSA 画面仕様書

PAGE 1

| 画面番号 | 科目名 | 科目コード | 取引名 | 取引通番 | 取引コード | 項目入力パターン | 取引名No. | JRNL精査 | OTM汎用 |
|---|---|---|---|---|---|---|---|---|---|
| 0200 | 普通預金 | 02 | 入金 | 14 | 00200 | 94 | 10 | 1 | 0 |




| SEQ NO. | 表示項目名 | データ名 | 桁数 | 送信可否 | SCP | 送信桁数 | データタイプ | 編集(MASK) | OTM | SPECIAL ROUTINE | CHECK NO. | HELP NO. | SKIP止 | 備考 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 口座番号 | N1 | 7 | 0 | 29 | 7 | | | | | | | | |
| 2 | 区分1 | N2 | 2 | 0 | 36 | 2 | | | | | | 30 | | |
| 3 | 金額1 | N3 | 12 | 0 | 38 | 10 | | 25 | | | | | | |
| 4 | 摘要 | N4 | 4 | 0 | 48 | 4 | | | | | | | | |
| 5 | カナ摘要 | N5 | 8 | 0 | 52 | 8 | 1 | | | | | | | |
| 6 | コード | N6 | 2 | 0 | 60 | 2 | | | | | | | | 新規追加 |
| 7 | 区分2 | N7 | 2 | 1 | 62 | 2 | | | | | | 30 | | |
| 8 | 金額2 | N8 | 12 | 0 | 64 | 10 | | 25 | | | | | | |
| 9 | 区分3 | N9 | 2 | 1 | 74 | 2 | | | | | | 30 | | |
| 10 | 金額3 | N10 | 12 | 0 | 76 | 10 | | 25 | | | | | | |
| 11 | 区分4 | N11 | 2 | 1 | 86 | 2 | | | | | | 30 | | |
| 12 | 金額4 | N12 | 12 | 0 | 88 | 10 | | 25 | | | | | | |
| 13 | 区分5 | N13 | 2 | 1 | 98 | 2 | | | | | | 30 | | |
| 14 | 金額5 | N14 | 12 | 0 | 100 | 10 | | 25 | | | | | | |
| 15 | 送信 | Soushin | 2 | 1 | 110 | 0 | | | | 255 | | | | |
| 16 | | | | | | | | | | | | | | |
| 17 | | | | | | | | | | | | | | |
| 18 | | | | | | | | | | | | | | |
| 19 | | | | | | | | | | | | | | |
| 20 | | | | | | | | | | | | | | |

**図 8　FSA 画面仕様書使用例**

Fig. 8　Sample of FSA screen specification

下段に項目入力ファイルの各属性項目を配している．これによって各カスタム・ファイル間の連関と，画面項目と項目入力ファイルの各テーブルとの連関とが一目で認知できる．

図 8 に示したのは，前述した普通預金の入金取引の修正例である．実際のワークステーションにおける修正操作の前に，このようにワークシートを記入しておけば無駄

な間違いも回避できるし，修正履歴も残ることとなる．

他のワークシートの一部を図9に示す．

### 5.3 ワークシートの効果

ABOSII版FINESSE-Jの顧客システム開発にあたって使用したカスタマイズ用ワークシートの効果は，作業の正確化，分業化，および文書化という形で現われただけでなく，作業量は増加したものの，作業時間の短縮に貢献する結果となり，わずかではあるが全体の作業工数は減少した．

また，本来意図した保守の局面においても，カスタマイズ・モジュールの操作上の利便性と視認性を十分補うだけでなく，FINESSE-Jの仕様書の一部としての活用も可能にしている．

しかしながら，これらのカスタマイズ用ワークシートが生み出した最大の効果は，カスタマイズ・モジュールの欠陥を補うという消極的な側面にあるのではなく，FINESSE-Jの柔軟性という特質をより強調したという積極的な側面にあると考える．確かに，ソフトウェアの欠陥への対応は，ソフトウェアの改善にしか求められない．しかし，いかに形骸的に見えようとも，このような「対症療法」は，既存のソフトウェアを単に否定するのではなく，そのソフトウェアの持つ特質を引き出して，いかに有効に活用していくかという問に対する一つの回答である．

## 6. FINESSE-Jの将来性とカスタマイズ・モジュール

近年の金融機関を取り巻く環境は，日々その変化の激しさを増し，新商品の開発や顧客への情報サービスの強化等，営業店のより戦略的な営業形態が求められつつある．

このような変化に応じてホストシステムはもとより，端末システムに対しても迅速かつ多様な対応が迫られる状況となってきている．

FSAによる営業店システムは，B39，B38/B28-EXPシリーズのハードウェアとBTOSをはじめとするモジュール構造を持つソフトウェア群の機能により，このような急速な変化に対応できる許容力と発展性，拡張性を備えており，営業店の多様な情報化の要請にも十分耐え得るであろう．

しかし，システム保守の面からこの状況を眺めると，克服困難とも思える矛盾点が透視できる．

変化の度合にかかわらず，変化自体がシステムへ及ぼす影響力は非常に大きい．ユーザ自身がFSAの保守を行うとの前提に立てば，端末システムの機能が複雑化すればするほど，保守の容易性が求められるという二律背反性が出てくるのである．

したがって，端末システムのシステム保守における，メーカおよびユーザの投資コストを最小限にする意味でも，FINESSE-Jというパッケージソフトウェアにとってのカスタマイズ・モジュールが持つ柔軟性と利便性との意味合いは，ますます重要になっていくと言わなければならない．

## 7. おわりに

FINESSE-Jを評価する上で，カスタマイズ・モジュール以外のより重要な視点がなかったわけではないが，それはまた別の機会に譲りたい．

## FSA 科目ファイル 仕様

PAGE 1

| 科目コード (2) | 科目名 (5) | 科目キー (2) | 取引ファイル検索FROM (4) | 取引ファイル検索TO (4) | 取引ファイルエリア番号 (1) | PROG NO. (1) | 取引HELP画面番号 (2) | 補助HELP画面番号 (2) |
|---|---|---|---|---|---|---|---|---|
| 01 | 当座預金 | 1 | 0001 | 0100 | 1 | 0 | 234 | 00 |
| 02 | 普通預金 | 2 | 0101 | 0200 | 1 | 0 | 235 | 00 |
| 03 | 通知預金 | 0 | 0201 | 0300 | 1 | 0 | 236 | 00 |
| 04 | 別段預金 | 0 | 1401 | 1500 | 3 | 0 | | 00 |
| 05 | 納税準備 | 0 | 0301 | 0400 | 1 | 0 | 237 | 00 |
| 06 | 定期預金 | 3 | 0401 | 0500 | 1 | 0 | 238 | 00 |
| 07 | 積立定期 | 0 | 0501 | 0600 | 2 | 0 | 239 | 00 |
| 08 | 定期預金 | 4 | 0601 | 0700 | 2 | 0 | 240 | 00 |
| 20 | 為替 | 0 | 0701 | 0800 | 2 | 0 | 241 | 00 |
| 29 | 職員預り金 | 0 | | | | | | |
| 51 | 預金CIF | 5 | | | | | | |
| 52 | 債務CIF | 6 | | | | | | |
| 60 | 割引手形 | 7 | | | | | | |
| 61 | 手形貸付 | 8 | | | | | | |
| 62 | 証書貸付 | 9 | | | | | | |
| 99 | システム | 0 | | | | | | |

## FSA 取引ファイル 仕様

PAGE 1

| 科目名 | 普通預金 | 科目コード | 02 |
|---|---|---|---|

| 項番 | 取引名 | SeqNo (INDEX) (4) | 取引コード (5) | 画面インデクス (4) | 項目入力パターン (3) | 取引名番号 (3) | ジャーナル・補金書込有無 (1) | OTM・汎用モード設定 (1) |
|---|---|---|---|---|---|---|---|---|
| 01 | 有帳入金 | 0101 | 00000 | 0001 | 001 | 001 | 0 ① | ⓪ 1 2 |
| 02 | 有帳入金取消 | 0102 | 10000 | 0004 | 002 | 001 | 0 ① | ⓪ 1 2 |
| 03 | 無帳入金 | 0103 | 00100 | 0102 | 023 | 002 | 0 ① | ⓪ 1 2 |
| 04 | 無帳入金取消 | 0104 | 10100 | 0107 | 024 | 002 | 0 ① | ⓪ 1 2 |
| 05 | 新規 | 0105 | 〃 | | | | | |
| 06 | 新規取消 | 0106 | 〃 | | | | | |
| 07 | 解約 | 0107 | 〃 | | | | | |
| 08 | 解約取消 | 0108 | 〃 | | | | | |
| 09 | 不渡 | 0109 | 〃 | | | | | |
| 10 | 不渡取消 | 0110 | 〃 | | | | | |
| 11 | 付込 | 0111 | 〃 | | | | | |
| 12 | 証書通帳再発行 | 0112 | 〃 | | | | | |
| 13 | 停止警告設定 | 0113 | 〃 | | | | | |
| 14 | 〃 解除 | 0114 | 〃 | | | | | |
| 15 | 変更 | 0115 | 〃 | | | | | |
| 16 | カード出金 | 0116 | 〃 | | | | | |
| 17 | 照会 | 0117 | 〃 | | | | | |
| 18 | 最終精査取消 | 0118 | 〃 | | | | | |
| 19 | 資金確定 | 0119 | 〃 | | | | | |
| 20 | | | | | | | | |
| 21 | | | | | | | | |
| 22 | | | | | | | | |
| 23 | | | | | | | | |
| 24 | | | | | | | | |
| 25 | | | | | | | | |

科目コード範囲 : 00
科 目 名 : 漢字
科 目 キ ー : 1
検索 FROM : 取引
検索 TO : 取引

<注意> 取引コード = I

## FSA 取引名ファイル 仕様

PAGE 1

| 項番 | 取引コード | 取引名番号 (3) | 取引名 (漢字12文字) | 項番 | 取引コード | 取引名番号 (3) | 取引名 (漢字12文字) |
|---|---|---|---|---|---|---|---|
| 01 | 00 | 001 | 有通帳 | 26 | 10 | 026 | 停止・警告設定 |
| 02 | 01 | 002 | 無通帳 | 27 | 10 | 027 | 書替 |
| 03 | 01 | 003 | 入金 | 28 | 11 | 028 | 停止・警告解除 |
| 04 | 01 | 004 | 条件変更 | 29 | 12 | 029 | 変更 |
| 05 | 02 | 005 | 新規 | 30 | 19 | 030 | カード出金 |
| 06 | 02 | 006 | 実行 | 31 | 20 | 031 | 照会 |
| 07 | 03 | 007 | 解約 | 32 | 20 | 032 | ダウン後照会 |
| 08 | 03 | 008 | 買戻 | 33 | 22 | 033 | 担保設定 |
| 09 | 03 | 009 | 内入 | 34 | 23 | 034 | 担保解除 |
| 10 | 03 | 010 | 回収 | 35 | 30 | 035 | 予約登録 |
| 11 | 04 | 011 | 不渡 | 36 | 31 | 036 | 明細予約 |
| 12 | 04 | 012 | 集中入金 | 37 | 32 | 037 | 予約照会 |
| 13 | 04 | 013 | 回収 | 38 | 40 | 038 | 通帳繰越(総合口座) |
| 14 | 04 | 014 | 完済 | 39 | 41 | 039 | 入金カード発行 |
| 15 | 05 | 015 | 付込 | 40 | 50 | 040 | 開局 |
| 16 | 05 | 016 | 不渡再起 | 41 | 51 | 041 | 再開局 |
| 17 | 06 | 017 | 付込 | 42 | 52 | 042 | 閉局 |
| 18 | 06 | 018 | 付込 (総合口座) | 43 | 93 | 043 | 書替解約 |
| 19 | 06 | 019 | 不渡回収 | 44 | 08 | 044 | 引落し |
| 20 | 06 | 020 | 完済 | 45 | 30 | 045 | 資金確定 |
| 21 | 07 | 021 | 証書・通帳再発行 | 46 | 03 | 046 | 解約(現金利払) |
| 22 | 07 | 022 | 利入 | 47 | 20 | 047 | 新型期日指定定期 |
| 23 | 08 | 023 | 内入 | 48 | 20 | 048 | 外貨預金 |
| 24 | 08 | 024 | 利息免除 | 49 | 20 | 049 | 毎日均等返済式月返済額 |
| 25 | 09 | 025 | 特別回収 | 50 | 20 | 050 | K回返済後残存元本 |

図 9　カスタマイズ用ワークシート使用例

Fig. 9　Sample of customize data sheets

最後に，ABOSII 版 FINESSE-J の顧客システム開発当時から現在に至るまで，FSA 導入において多大な援助や重要な助言を与えてくれた FSA システム部に謝意を表したい．

---

**参考文献** ［1］ 岡井功雄・横田正信，「金融機関における営業店統合システム—FSA」，UNISYS 技報 通巻 21 Vol.9 No.1，日本ユニシス，1989 年．
［2］ 「FSA FINESSE-J 概説書」ユニシス・マニュアル 6552417283．
［3］ 「FSA FINESSE-J 使用ガイド カストマイザ編」ユニシス・マニュアル 6552407202．

**執筆者紹介** 天　野　　剛（Takeshi Amano）
昭和 32 年生．57 年独協大学法学部法律学科卒業．60 年日本ユニシス（株）入社．金融機関のシステム開発に従事．現在東北支店システム部に所属．

# S 8400 シリーズ・エンジニアリング・ワークステーションの開発

## The Development of the S8400 Series Engineering Workstation

吉田欣司, 唐下　勉

要　約　近年，ユーザのニーズが原動力となりオープンシステムが，コンピュータ市場での大きな流れになっている．それに応えるものの一つとして，国際標準とされているUNIX*を搭載した高性能エンジニアリング・ワークステーション（EWS）に対する期待が高まってきている．

そこで，米国ユニシスと日本ユニシスの共同開発チームが，モトローラ**社のM 88000**ファミリのRISCプロセッサを採用し，日本市場に合わせた先進的なテクノロジを備える高性能EWSを開発した．

本稿では，今回開発したEWSで，高性能化を実現するためのハードウェア面での工夫および考慮点について述べる．

**Abstract**　The recent upsurge of user needs has driven the industry's more active approaches to open systems into the mainstream in the computer market. As one of the solutions, user demand has intensely focused on the necessity for an international standard Unix-based, higher-performance engineering workstation (EWS).

In response to such trends, a joint development team formed together by both Unisys Corporation in the U.S.A. and Nihon Unisys, Ltd. has succeeded in developing a high-performance engineering workstation with state-of-the-art technology fully incorporated for the Japanese market through the use of Motorola's M88000 Family RISC processor.

This paper describes the hardware features newly devised and some special considerations adopted so the new EWS can provide the required levels of high performance.

## 1. はじめに

近年，マイクロプロダクトの急進展が，ホストコンピュータのダウンサイジング(小型機需要の急増）の動きに拍車をかけている．これにより，これまでの垂直型分散システム形態に対し，システムを柔軟に組み合わせられる水平型分散システムを構築するためのオープンシステムの強化が必要とされている．加えて，ワークステーション（WS）およびパーソナルコンピュータ（パソコン）の世界においても，WSの低価格化とパソコンの機能アップにより両製品の境界線がはっきりしなくなってきた状況から，オープンシステムにおけるプラットフォームとしての中核的位置を確固たるものとする製品が必要とされている．

そのための施策の一つとして，システム面では，国際標準とされているUNIXオープン・システムを前面に押し出し確固たるものにすることであり，ハードウェア面では，ますますの高性能化を実現することが重要なテーマになっている．

これらの情勢に対応するため，米ユニシスと当社が共同開発プロジェクトを起こし，

* UNIXオペレーティングシステムは，UNIX System Laboratories, Inc.が開発し，ライセンスしている．

** モトローラは，米国MOTOROLA社の登録商標であり，M88000は同社の商標である．

日本市場に合わせた先進的なテクノロジを備えた高性能EWS（エンジニアリング・ワークステーション）の開発にあたることとした．このプロジェクトには，当社から10数名が参加し，米国サンノゼにてハードウェアおよびソフトウェアのベーシック・デザインから評価の工程に至るまでの作業を行い，1991年6月に新S 8400シリーズ・ワークステーションとして初号機を出荷する運びとなった．

今回，開発したS 8400シリーズの特徴は，上述のテーマを具現化するために，モトローラ社製のM 88000ファミリのRISCプロセッサ（縮小命令セットコンピュータ）を採用し，システム全体の性能強化を図り，高速・高性能を実現したものである．

本稿では，今回開発したS 8400シリーズについて，ハードウェア面から見た高速・高性能を実現するための構造，工夫および拡張性等について述べる．また，製品についての理解を深めるために，製品の特徴の一部についても述べる．

## 2. 製品の特徴

本シリーズのハードウェアは，モトローラ社製のRISCプロセッサMC 88100およびMC 88200*（25/33 MHz）を採用し，基本キャビネットに64メガバイト（4メガビットのDRAM使用時は，最大256メガバイト）大容量メモリの搭載ができるとともに，補助記憶装置として，フロッピ・ディスク装置，カートリッジ・テープ装置，ハード・ディスク装置，さらに，外部記憶装置として1/2インチ磁気テープ装置等の接続ができる．また，多彩な入出力装置を接続できるようにイーサネット**，セントロニクスインタフェース***を各1ポートおよびコミュニケーション・インタフェースを8ポート標準装備した．

外観形状は，図1に示す通りである．フロントパネルのスタイルに独創的な波形を

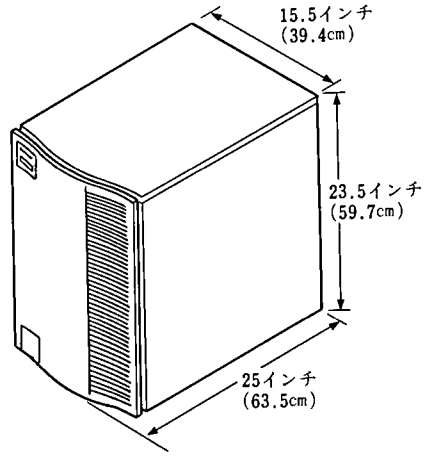



**図1 外観図**

Fig. 1 Isometric view

* MC88100およびMC88200は米国MOTOROLA社の商標である．

** イーサネット（ETHERNET）は米国XEROX社の登録商標である．

*** セントロニクスは米国CENTRONICS社の登録商標である．

表1 諸　元
Table 1 Specifications

| 仕　様 | | 内　容 |
|---|---|---|
| 外形寸法(幅×奥行×高さ) | | 394×635×597 mm |
| 重量 | | 最大 79.4 kg |
| 入力電源 | | 100～240 VAC, 50/60 Hz |
| 消費電力 | | 1440 VA |
| 発熱量 | | 1053 Kcal(キロカロリ)/時 |
| 騒音 | | 55 db 以下 |
| 環境条件(稼働時) | 温度 | 13～35°C |
| | 湿度 | 20～85% |
| 電波障害 | | VCCI 第一種 |

採用したことで，高性能機の能力を直接的に表現するのでなく，人に優しいイメージを与える形状とした．フロントパネルの設計では，上下方向を対称とする設計で拡張キャビネット（隣接のオプション・ユニット：VME*バスの拡張あるいはペリフェラル・ボックス用に用意している）のフロントパネルに逆取付けするだけで，そのまま使用できる構造を実現した．

環境条件・諸元（表1参照）は，その配慮として，低騒音，低雑音，低発熱等により使用環境を汚染しないことを前提とし，さらに，オフィスのスペース効率を上げるために JIS 規格 S 1010 の事務机の寸法を考慮して机の下にレイアウトできる筐体寸法とした（市販されている同種クラスのワークステーションの中で，最も高さ寸法を抑えたものとした）．

ソフトウェアは，オペレーティング・システムに最新の UNIX System V リリース 4.0 (UNIX SVR 4) を基本に，ネットワーク機能，システム管理機能，ウィンドウ機能，各種言語の充実を図った．また，標準化のために IEEE 1003.1, 1988 POSIX 規格，X/OPEN**移植ガイド（XPG 3），AT&T System V インタフェース定義 (SVID 3)，88 オープン・コンソーシアム（モトローラ社製の RISC プロセッサ M 88000 ファミリの利用を国際的に普及させるために設立した機関であり，88 オープンの BCS および OCS 勧告に完全に準拠することで，他システムで開発されたソフトウェアを稼働させることができる）の各規格/勧告に完全準拠させた．

## 3. S 8400 シリーズの開発

本シリーズの開発にあたっては，背景，商品化戦略あるいは製品開発方針を受けて技術的狙いを設定した．これらの決定にあたっては，日本市場の要求を盛り込む必要があることから，数多くの折衝/調整を行い，その上で決定に至ったものである．以下に代表的なものを列挙する．ただし本稿では，商品化戦略・製品開発方針については

* VME (Versa Module Europe) は，IEEE P1014 と IEC47b の両標準委員会による規格である．
** X/OPEN は X/OPEN 社の登録商標である．

取り上げない．

1) モトローラ 88000 RISC アーキテクチャ
2) UNIX SVR 4
3) Real Time OS
4) EWS ユース vs ビジネス UNIX ユース対応
5) 1メガビット/4メガビット×20メモリ SIMM (Single Inline Memory Module)
6) ECC (Error Correction Code) プロテクト
7) VME オプションボード，4スロット，6U/9U カード
8) I/O プロセッサ：8ポート・コミュニケーション・インタフェース
   イーサネットインタフェース (IEEE 802.3)
   パラレルインタフェース
   SCSI*インタフェース
9) 補助記憶装置内蔵
10) 国際的に通用する入力電源対応
11) 筐体の机下設置
12) 拡張性の考慮：マルチ CPU，キャッシュ容量，64 ビット対応，等々

以下に，上述の狙いを具現化するために，とくにポイントとなったものを取り上げ，それらについての設計上での工夫，考慮点および実現方法について述べる．

### 3.1 ハードウェア構成

ハードウェアは，IOP ボード，CPU/Memory (CPU/MEM) ボード，VME インタフェースボード，VME オプションボードとそれらを収納するロジックカードケージ，電源，SCSI デバイスを格納するためのストレージ・デバイス・ケージから構成されている．それらの物理的位置については図2に示す．

IOP ボードは，RS 232 C インタフェース×8ポート，イーサネットインタフェース×1ポート，セントロニクスインタフェース×1ポート用コネクタを持ち，これらのコネクタは筐体背面からアクセス可能なように IOP ボードのハンドル部分に位置させた．また，IOP ボードの提供する SCSI インタフェースは，ボードに隣接するストレージ・デバイス・ケージに接続するため，IOP ボードからロジックバックプレーン，SCSI インターコネクト基板および SCSI ミッドプレーンを介して内蔵される SCSI デバイスに接続する．この SCSI インタフェースは，ストレージ・デバイス・ケージの最下段に位置する SCSI 拡張ボードにより，筐体背面に外部装置接続用の SCSI ポート (50 ピンシールド型コネクタ) を用意した．外部装置を接続しない場合は，このコネクタに SCSI ターミネータを接続し，SCSI バスの終端処理を行うように設計した．

ストレージ・デバイス・ケージは，SCSI デバイスを筐体内に収納するためのケージであり，物理的には筐体右側面に位置している．このケージは SCSI デバイスをハーフハイトで最大7台 (フルハイトでは最大4台) まで収納できる構造とし，それぞれのデバイスへは筐体前面および背面からアクセス可能とすることにより，カートリッジテープ装置，フロッピディスク装置等のリムーバブルメディアを扱う装置の搭載を可

* SCSI (Small Computer System Interface) は ANSI X3. 131-1986 として承認された規格である．

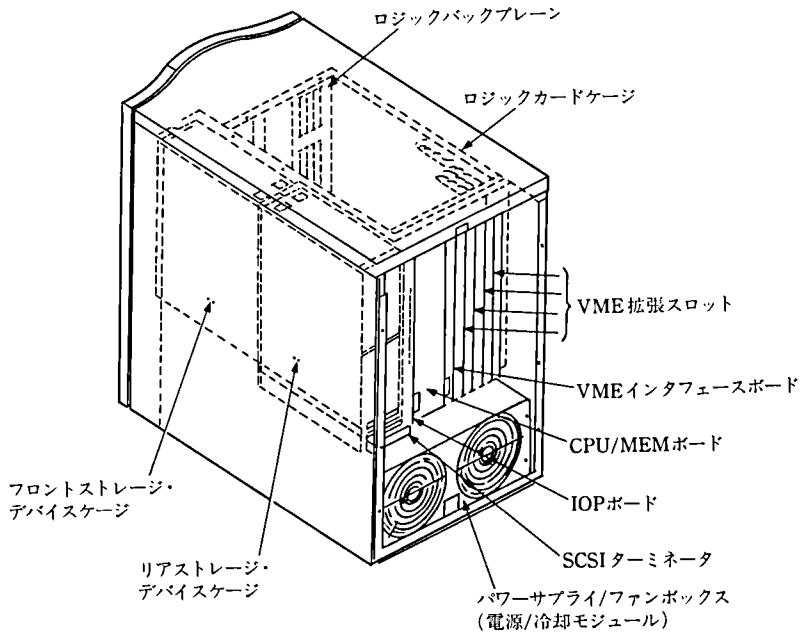



**図 2　ハードウェア構成図**

Fig. 2　Group assembly drawing

能とした．さらに，保守性/操作性を考慮して，各デバイスには，専用ブラケットを用意することで特殊ツールを使用することなく内蔵デバイスを容易に着脱できる構造とした．

CPU/MEM ボードは，IOP ボードの隣に位置し，その隣には VME 拡張スロットを使用する場合に必要となる VME インタフェースボードを位置させた．

VME インタフェースボードは，ビジネス UNIX マシンとしての使用形態を考慮し，オプション扱いとした．

VME インタフェースボードの右隣には，4 セット分の VME ボードが搭載できるエリアを確保した．このエリアの 4 スロット目には，VME のダブルスロット分のスペースを必要とするボードが搭載できる物理的スペースを確保した．このスペースは，VME バスを拡張キャビネット（オプション）に接続する時の，ロジックバックプレーン上で必要なスペースを流用したものである．拡張スロットのサイズは，汎用性を持たせるため，ユーロボード*サイズの 9 U×400 mm を採用した．また，コネクタのピンアサイメントは，P 1，P 2 を IEEE P 1014/D 1.2（VMEBUS Rev. C. 1）に完全準拠させることで，市販されている各種オプションボードの利用を可能とした．P 3 のピンアサイメントは，他社機との互換性を保つデザインとした．

電源は，筐体の最下部に配置し重量バランスをとることにより，筐体の物理的安定性を確保した．この電源は，筐体内部で使用するすべての直流電源（内蔵される SCSI デバイスおよび VME オプションボードに必要な電源を含む）を供給するものであり，

* ユーロボード（Euroboard）は ANSI/IEEE std 1101-1987 の規格である．

その電流容量は+5 V/150 A, +12 V/16 A, −12 V/3.6 A, −5.2 V/10 A の容量を持っている．さらに，入力交流電源としては，世界各地での使用を考慮して，AC 100 V ～240 V の範囲の電圧を無調整で連続的にカバーできるように設計し，その必要電力はスイッチング電源の電力変換効率を高めることにより最大 1440 VA に抑えた．

筐体内の冷却処理に関しては，筐体背面方向に空気の流出を行わせる方式とし，そのためのファンを 2 基配置するとともに，ロジックカードケージ上の温度をセンシングすることでファンの回転数を制御し空気の流出量を変化させるデザインとした．

前述の基本キャビネットに対し，VME 拡張スロットおよびストレージ・デバイス・ケージの増設用としてオプションの拡張キャビネットを用意した．このキャビネットは，VME 拡張スロットとして 7 スロット，ストレージ・デバイス・ケージとして 2 ケージ持つ構造とし，拡張キャビネット全体としては，ハーフハイトで最大 14 台(フルハイトでは最大 8 台)までの SCSI デバイスが収納できる．また，この二つのケージはそれぞれ独立した SCSI バスに接続するようにデザインした．拡張キャビネットの VME オプションスロットは，基本キャビネットの VME バスと接続することにより使用できる構造とし，接続する場合は基本キャビネットのサイドパネルを取り外し，そのパネルを拡張キャビネットのサイドパネルに流用する構造とした．これにより，拡張キャビネットを基本キャビネットの左横に隣接設置することで両キャビネットの一体化を図った．両キャビネット間の VME バスの接続は，ロジックバックプレーンをドッキングさせることで実現した．また，拡張キャビネットは基本キャビネットと同容量（同一品）の電源を持ち，VME 拡張スロットと二つのストレージ・デバイス・ケージに必要電力を賄うデザインとした．

### 3.2 アーキテクチャ

高速・高性能を具現化するため，以下に示すような，1) 機能ブロック，および 2) バス構造からなるアーキテクチャを生み出した．

1) 機能ブロック（図 3)

① CPU/MEM ボードは，最大 128 メガバイトまで増設可能な基本メモリ部と，MC 88100 CPU，MC 88200 CMMU（Cache Memory Management Unit）を搭載した CPU/Cache 部および増設可能な拡張メモリ部から構成

② IOP ボードは，I/O コントロール専用プロセッサとしてモトローラ社 MC 68020*MPU (Micro Processor Unit) を採用し，IOP ボード上の各種インタフェース用コントローラを制御

③ VME インタフェースボードは，IM バスと VME バスとのバスプロトコルの変換

2) バス構造（図 4)

① CPU と CMMU 間は，モトローラ社が提供している P バスを採用

② CMMU とメモリ間は，CMMU が提供している M バスを介して接続

③ CPU/MEM ボード，IOP ボードおよび VME インタフェースボード間の接続は，IM バスを開発

④ IOP のボード内部は，U バス，D バス，F バスの 3 種類のバスを開発

* MC68020 は米国 MOTOROLA 社の商標である．

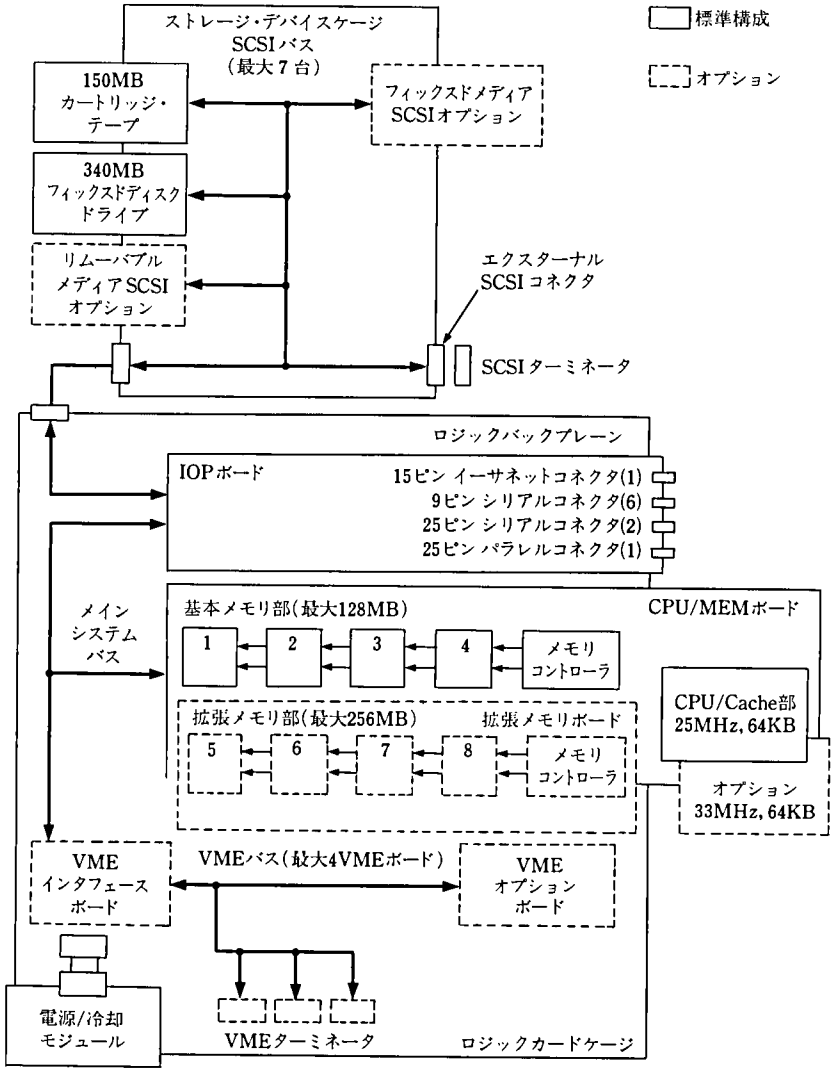



**図 3　機能ブロック図**

Fig. 3　Functional block diagram

⑤ ロジックバックプレーン上では，CPU/MEM ボード，IOP ボードおよび VME インタフェースボードを接続する IM バス，IOP ボードから出力される SCSI バス，VME インタフェース間を接続する VME バスの 3 種類のバスを用意

## 3.3 CPU/MEM

本シリーズでは，前述の通り CPU および CMMU にモトローラ社製の MC 88100, MC 88200 を採用し，これらの持つ機能・性能を引き出すことをテーマとした．

### 3.3.1 物 理 的 構 成

CPU/MEM ボードは，次の三つに分割した基板から成っている（図 5）．

① メモリコントローラ（GPMC : General Purpose Memory Controller）を含

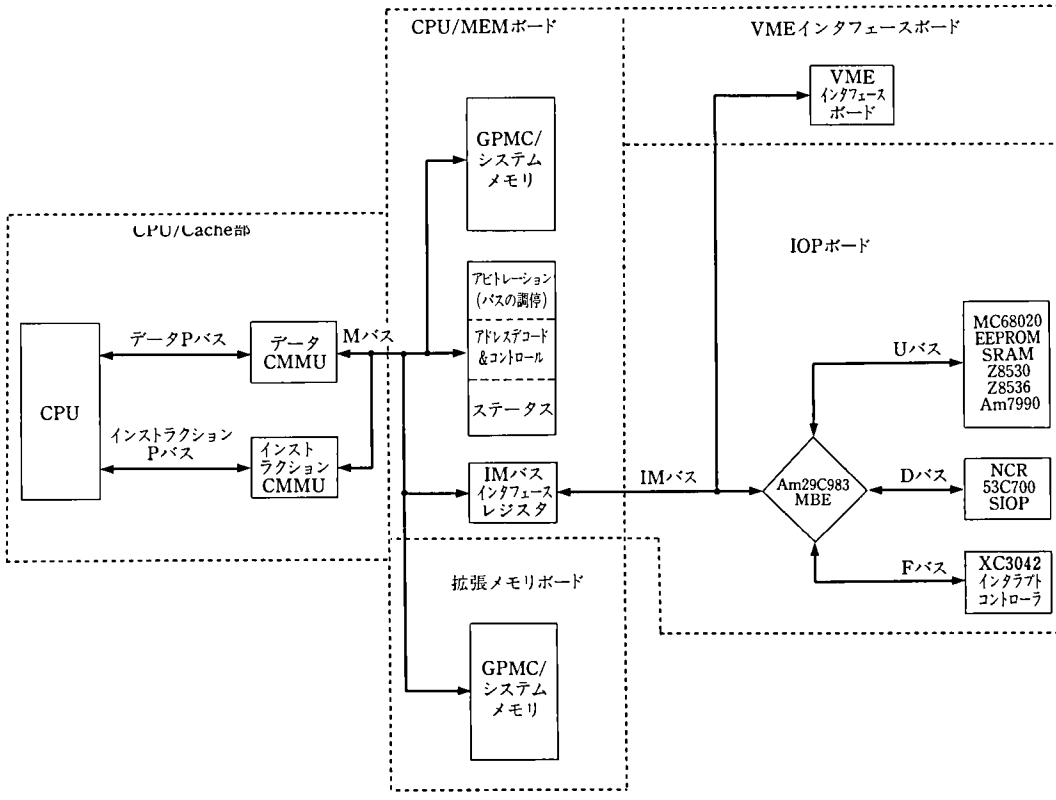



**図 4　バス構造**

Fig. 4　BUS architecture

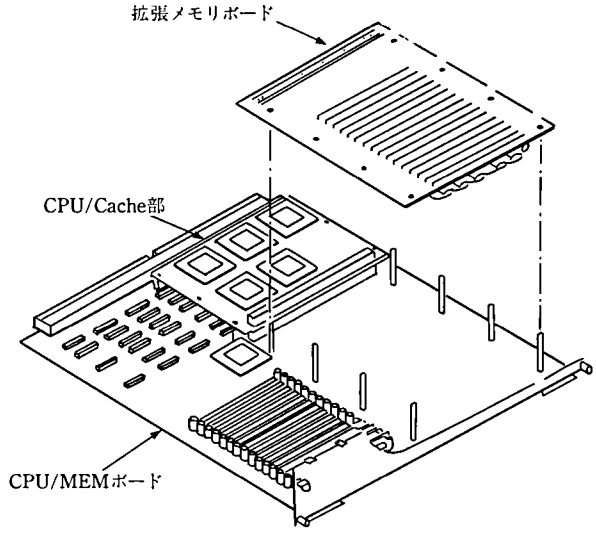



**図 5　CPU/MEM ボード**

Fig. 5　CPU/MEM board

む，基本メモリ部分（4 メガビット DRAM 搭載時：最大 128 メガバイト）を実装したベースボード

② CPU および CMMU を実装した CPU/Cache モジュール

③ メモリモジュールを増設するための拡張メモリボード（基本メモリ部分と合わせて最大 256 メガバイトまで拡張可能）

上述の 3 分割した設計上の狙いは，マルチ CPU への対応，キャッシュの容量の変更，メモリの増設等が容易に行えるようにすることである．これらを実現するため，ベースボード，CPU/Cache モジュール，拡張メモリボードの各ボード間を 240 ピンのコネクタを介して 2 段/3 段重ねの構造で接続する方式をとり，同時にこの間をバスとする構造にした．

その結果，

① 物理的スペースを確保，

② ボードの交換あるいは追加での拡張，

③ バス構造とすることでの電気的な設計上のむずかしさを極力抑えること，

等に成功した．これらにより，S 8400 シリーズを物理的，電気的に拡張性に富んだシリーズとすることが可能となり，CPU のグレードアップ(64 ビット化，クロックレートのアップ）が図られた場合においても容易に対応できる構造とした．

### 3.3.2 CPU

MC 88100 CPU の主な特徴[1]を表 2 に示す．また，CPU 内部のブロック図[1]を図 6 に示す．

**表 2　MC88100 の特徴**

Table 2　MC 88100 Major features

（日本モトローラ社半導体事業部，「MC88100 Technical Summary 32 ビット第三世代 RISC マイクロプロセッサ」，FEB '90 から）

- 整数，論理，ビットフィールド，分岐およびストア命令の 1 クロックサイクル実行
- 51 種の命令および 6 種のデータタイプ
- きめ細かな並列処理：
  - 4 本の完全並行動作パイプライン
  - レジスタ・スコアボード・ハードウェアによる実行同期化
- 非破壊型レジスタおよび条件コードモデル
- 32 個の汎用レジスタ
- 単精度および倍精度 IEEE 754 浮動小数点演算（各クロックサイクルごとに 1 動作）
- データおよび命令用の独立したメモリポート
  - 30 ビットのデータアドレスバス（32 ビット境界アドレス指定）
  - 30 ビットの命令アドレスバス（32 ビット境界アドレス指定）
  - 32 ビットの命令バス（32 ビット固定命令長）
  - 32 ビットのデータバス（32 ビットワード）
- パイプライン化されたロード動作およびストア動作
  （33 MHz で最高 133 M バイト/秒）
- 割り込み待ち時間が最小の高速割り込み処理
- チェッカモードによるフォールト検出（冗長機能）
- 完全にハードウェア化された動作（非マイクロコード）
- バイト順序はビッグエンディアンまたはリトルエンディアンのいずれかを選択可能
- メモリまたは MC 88200 CMMU への直接インタフェース
- 高級言語実行モデル
- 特殊機能ユニットにより拡張可能なアーキテクチャ

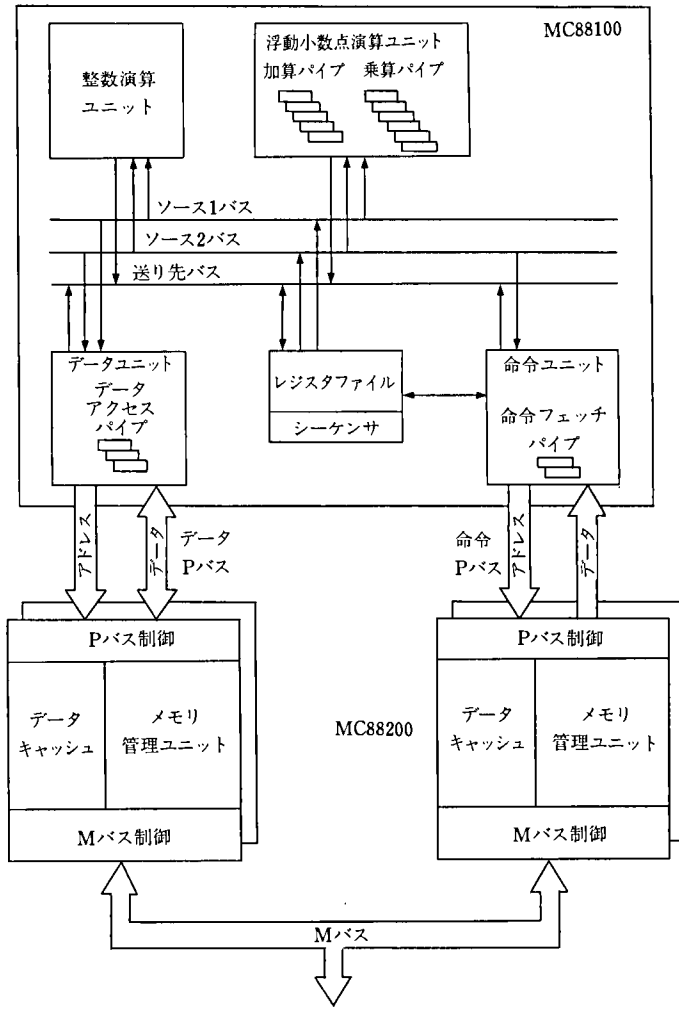



**図 6　CPU/CMMU ブロック図**

（日本モトローラ社半導体事業部，「MC 88100 Technical Summary 32 ビット第三世代 RISC マイクロプロセッサ」FEB '90 から）

Fig. 6　Block diagram of CPU/CMMU

このCPUは，最適化された内部のパイプラインにより，フローティングポイント演算を除くほとんどの演算が1CPUクロックで実行可能であることが最大の特徴である．CPUはその内部に持つ命令ユニット，整数演算ユニット，データユニット，浮動小数点ユニットの各ユニットのパイプラインで演算を並行処理し，命令ユニット以外のユニットではパイプライン間のデータの共有，同期化および命令のフロー制御はハードウェア制御のレジスタ・ファイル/シーケンサによって自動的に行われる．

しかしながら命令ユニット内部のパイプラインの場合は，分岐命令の実行等ソフトウェア的要因によりパイプラインが無効化され，このことが実効演算速度の低下を招くことになるが，CPUの命令セットの中には，命令ユニット内部のパイプラインの無効化を防ぐための遅延分岐命令（Delayed Branch: br. n, jmp. n, jsr. n, bb 0. n, bbl. n, bcnd. n, bsr. n）が用意されており，分岐時これらの命令を使用することにより分

岐を実行するか否かに係わらず分岐命令の直後の命令を実行できる．

つまり，これらの命令がパイプラインの実行ステージに移った時の翻訳ステージ上の命令は無効化されることはなく，次のクロックで実行される．また，この時の命令フェッチステージのアドレスは実行ステージ上の分岐命令により決定されるため，次の命令フェッチでは分岐先の命令がフェッチされることになる．本シリーズでは，コンパイラにより生成される分岐命令のほとんどが遅延分岐命令となるように，コンパイラの設計に工夫をこらすことで，CPU 内部のパイプラインの順序制御の乱れを起こさないよう配慮した．

### 3.3.3 CMMU

MC 88200 CMMU は，RISC バーチャルシステムを構築する際に必須となるメモリ管理ユニットとキャッシュメモリの両機能を合わせ持った LSI である．キャッシュメモリ部は LRU 置換アルゴリズムにより制御されたノーウェイト・ステート・アクセス可能な 16 キロバイトの 4 ウェイセット・アソシエイティブ物理キャッシュで構成される．メモリ管理ユニット部にはハードウェアにより管理されるページアドレス変換キャッシュ(PATC)，およびソフトウェアにより管理されるブロックアドレス変換キャッシュ(BATC)を持っている．この CMMU の主な特徴を表 3[2]に示す．この CMMU

表 3　MC88200 の特徴
Table 3　MC 88200 Major features
(日本モトローラ社半導体事業部，「MC 88200 Technical Summary 16 キロバイト・キャッシュ/メモリ管理ユニット(CMMU)」，FEB '90 から)

| |
|---|
| MMU 部： |
| ・それぞれ 4 G バイトの論理アドレス空間が 2 個（ユーザ/スーパバイザ） |
| ・自動的に管理される PATC とソフトウェアにより管理される BATC |
| ・ユーザアクセスとスーパバイザアクセスに対する書き込み保護 |
| ・ページ変換テーブル内に保持される使用済みフラグと更新済みフラグ |
| ・メモリロケーションの状態をテストするためのプローブ機能 |
| キャッシュ部： |
| ・16 K バイト，4 ウェイセット・アソシエイティブ物理キャッシュ |
| ・物理キャッシュへのノーウェイト・ステート・アクセス<br>（アドレス変換はキャッシュへのアクセスと並行して行われる） |
| ・キャッシュエントリの割り当てはコピーバック方式またはライトスルー方式により行われる． |
| ・各キャッシュセット用の LRU 置換アルゴリズム |
| マルチプロセッサ支援： |
| ・キャッシュと他のキャッシュとの間およびキャッシュとメインメモリとの間のコヒーレンシはバススヌーププロトコルにより維持される． |
| ・キャッシュのフラッシュと無効化はソフトウェアによって選択的に，またハードウェアによって自動的に開始される． |
| ・エリア，セグメント，ページおよびブロック単位のキャッシュ禁止フラグ |
| ・キャッシュライトスルー，キャッシュコピーバックのいずれかが，エリア，セグメント，ページ，ブロック単位で選択可能 |
| ・効率的なマルチプロセッサ同期化のためのセマフォ（メモリとキャッシュ内） |
| ・データキャッシュと ATC は任意のプロセッサまたは IO デバイスにより，フラッシュ可能 |
| フォールト・トレーラント・アプリケーション支援： |
| ・チェッカモードによるフォールト検出（冗長機能） |
| ・パリティによるメモリバスの保護 |
| ・キャッシュライン・ディセーブルフラグ |

は，CPU に直接接続された最大構成時 1 CPU あたり 8 個までの CMMU が接続可能である．ただし本シリーズでは 4 個の接続とした．

効率面から見た時に CMMU を採用することの最大の目的は，CPU からのノーウェイトのメモリアクセスおよび実メモリへのアクセス頻度を減らすことによるメモリバスの使用効率アップである．RISC バーチャルシステムの場合，アドレス変換テーブルの参照によるメモリバストランザクションの増大と，RISC 命令での命令数の増加による同トランザクションの増大が予想された．そこで，PATC，BATC の機能を最大限に利用することにより実メモリ上のアドレス変換テーブルの参照を最小限に抑えることとし，さらに CPU からメモリへのライト時のトランザクションを減らし，メモリアクセスによる CPU のウェイトロスをなくすために，動作モードとしてコピーバックモードを採用した．これにより，メモリへライトされるべきデータは，いったんキャッシュ内部にライトされるだけでメモリへのライトは行われないことから，ライト時のメモリトランザクションを減少させることができ，さらに，ライト時の CPU のウェイトサイクルを大幅に軽減することができた．しかし，この状態ではキャッシュ内部のデータと実メモリ上のデータとの間で不整合が生じることから，これを防ぐために，バススヌープ・プロトコルによってデータのコヒーレンシ（coherency：一貫性）を維持した．このプロトコルは，メモリバス上のグローバル・トランザクションをハードウェアで常に監視し，メモリバス上に存在する他のバスマスタ（IOP ボード，VME インタフェースボード，マルチ CPU の場合の他の CPU）から不整合が生じているメモリへのアクセスが行われたとき，そのアクセスを先取りし実メモリ上のデータをキャッシュデータで更新することにより，データのコヒーレンシが維持される．一方，この CMMU はメモリバスとして M バスを採用しており，M バスのバスマスタとしての機能も合わせ持っている．

### 3.3.4 メモリコントローラ

メモリコントローラは，効率，冗長性，データの信頼性（完全性）および実装密度（省スペース）をテーマに表 4 に記載した機能をゲートアレイ（GPMC）にて実現した．

**表 4　GPMC の機能概要**
Table 4　Functional description of GPMC

- シングルワード（32 bit）リード/ライト
- バイトライト
- ブロックリード/ライト
- リフレッシュサイクルコントロール
- 2 ウェイインタリブコントロール
- 1 MB および 4 MB SIMM の提供
- ECC プロテクション：　マルチビットエラーの検出
  シングルビットエラーの訂正
- シングルビット，マルチビットエラーのエラーロギング機能

GPMC の開発に際しては，RISC プロセッサにおけるメモリトランザクションの増大によるメモリバスネックの懸念から，

① 前述の CMMU の機能および M バスの持つバーストリード/ライト機能を使用できること，

② GPMC のブロックリード/ライト機能を M バスの機能に合わせること，

③ GPMC-メモリ間を 2 ウェイインタリブ方式とすること，

の実現によりキャッシュからのトランザクションの減少を図り，データリード/ライト時間を削減することでバスの効率アップを図った．メモリについては，1 メガビットあるいは 4 メガビット SIMM のどちらでも使用できるようにするため，SIMM の出力ピンに ID ビットを持たせることで両タイプの SIMM をハードウェアによって識別し，アドレスラインを自動的に変更することでソフトウェアに意識させることなく SIMM の変更ができる設計とした．このことはメモリの増設時に，1 メガビット/4 メガビット SIMM の混在を可能とした．(注意：拡張メモリボードとの組み合わせが取扱い上むずかしいことから混在での販売は行っていない)

また，ECC 機能を組み込むことで，ハードウェアによるメモリデータ上のシングルビットエラーの自動訂正およびマルチビットエラーの検出を可能とし，さらに同エラー時のロギング機能を持たせたことでデータの信頼性および保守性も高めた．

この結果，上述の機能を 1 個のゲートアレイに搭載したことで大幅に省スペース化を図ることができた．

### 3.3.5 バ ス 構 造

CPU/MEM ボード内のバス構造については図 4 に示す．

CPU と CMMU 間は，モトローラ社より提供されている P バスを採用した．このバスは，単一の CPU と複数の CMMU 間を接続するための専用バスである．バス・プロトコルはパイプライン化および最適化が行われており，CPU に対し 4 バイトを 1 クロック・サイクルで転送できる (33 MHz で最大 133 メガバイト/秒) 能力を持っている．

CMMU とメモリ間は，CMMU が提供する M バスを採用した．このバスはマルチ・バスマスタ/マルチ・バススレーブ・デバイスを接続可能なアドレス/データ時分割の同期式 32 ビットバスであり，バスの持つ機能として単一リード/ライト，バーストリード/ライト，不可分サイクル機能，外部キャッシングの禁止/許可，バスエラー通知等の機能を持っている．

CPU/MEM ボードと IOP ボード，VME インタフェースボードを接続するバスについては，IM バスを開発した．

この IM バスのデザインとしては，

① M バスの持つ機能を最大限に利用すること，

② バスに接続されるデバイスのバス制御回路の負担を軽減すること，

③ IOP ボード上のマルチプルバス・エクスチェンジャに対する制御回路を簡素化すること，

④ VME インタフェースボード上のバス制御回路の簡素化をすること，

から M バスライクな 32 ビットの非時分割同期バスの構造とした．

さらに，IM バスが物理的に位置するロジックバックプレーンの配線は，多層化技術により将来予測されるクロックレートの高速化にも十分対応できる構造とした．

## 3.4 IOP

本節では，IOP ボードの機能・性能仕様，それらを実現するための方法および設計上の考慮点について述べる．

### 3.4.1 機能・性能仕様

IOP ボードに接続する IO 用のインタフェースの仕様について表5に示す.

表5 インタフェース仕様
Table 5 IO Interface specifications

| 物理/電気的インタフェース仕様 | | 性 能 |
|---|---|---|
| RS 232 C ASYNC | D-SUB 9ピン・タイプ・コネクタ 6ポート | 19.2キロビット/秒 |
| RS 232 C SYNC/ASYNC | D-SUB 25ピン・タイプ・コネクタ 2ポート | 38.4キロビット/秒 |
| ETHERNET(IEEE 802.3) | スライドラッチ・タイプ・コネクタ 1ポート | 10メガビット/秒 |
| CENTRONICS | D-SUB 25ピン・タイプ・コネクタ 1ポート | 300キロバイト/秒 |
| SCSI | シールド 50ピン・タイプ・コネクタ 1ポート | 4メガバイト/秒 |

### 3.4.2 構　　成

CPU (RISC プロセッサ) の負荷を分散するため, IO 処理専用のプロセッサとして, モトローラ社製の MC 68020 の採用と各種 IO 処理用の LSI とを組み合わせるハードウェア構成とした. また, IOP に接続される IO の処理を効率よくするため内部バスを階層化構造にすることとした (図4). 以下, 主要構成品と階層化した内部バス構造について述べる.

### 3.4.3 内部バス構造

システム全体のバランスおよび IO のデータ転送処理の能力を確保するため IOP 内の内部バスを階層構造とし, U バス, D バス, F バスを開発した. また CPU/MEM ボードと接続される IM バスと内部の3種類のバスを接続するために, AMD*社の MBE (マルチプルバス・エクスチェンジャ) を採用した. この MBE[3]は高速4ポートデジタル・クロスポイント・スイッチの機能を持ち, 1ポートあたり9ビットの双方向ラッチタイプ IO ポートと各ポートを接続するクロスポイントバスから構成されている. この LSI はマルチバスシステムのバス接続を行う時に最適な LSI であり, 一つのポートから任意のポートにデータを通過させると同時にそのデータを保持することも可能である.

この機能を利用し, 同期式 IM バスと非同期式のその他のバスの速度差をこのラッチ機能で吸収させることにより, IM バスの使用効率の向上を図った. 本シリーズでは, データバス幅, アドレスバス幅, 制御信号のボリュームから, この MBE を8個 IOP ボード上に搭載し, 簡潔な回路設計を実現した.

1) U バス……このバスの主要構成品は, MC68020MPU, SRAM (256キロバイト), EEPROM (256キロバイト), ザイログ社**の Z8536CIO×2, Z8530SCC×4, AMD 社の Am 7990 LANCE/Am7992SIA***, MK48T02TOD-Chip****, イーサネット・ステーション・アドレスを格納した PROM からなる.

モトローラ社の MC 68020[4]を IO 分散処理用の専用プロセッサとして使用することから, U バスは IOP ボードの制御上中心的位置付けにあり, バスには

* AMD は米国 Advanced Micro Device 社の登録商標であり, MBE は同社の商標である.
** ザイログは, 米国 ZILOG 社の登録商標であり, Z8536CIO, Z8530SCC は同社の商標である.
*** Am7990LANCE, Am 7992 SIA は米国 AMD 社の商標である.
**** MK48T02TOD は, 米国 Thomson Components/Mostec Corp.の商標である.

MPU が提供するバスをそのまま採用した．したがって，このバスは 32 ビット非同期バスであり，MPU の持つダイナミック・バス・サイジング機能と合わせて，種々のデータ・バス幅のデバイスに対して高速にアクセスが可能となる．この特徴を利用し比較的低速で，MPU に対して負荷のかからない RS 232 C インタフェース制御用の Z8530SCC[5] およびセントロニクスインタフェース制御用の Z8536CIO[6] をこのバス上に配置した．また，イーサネットインタフェース制御用の Am7990LANCE[7] については，あらかじめ本バス上の SRAM 内にバッファ・マネージメント・ストラクチャを組み上げておけば，MPU の介在なしでパケットデータを Am7992SIA へ，あるいは SIA からのパケットデータを DMA 転送でき，MPU に対しての負荷をかけない作りが可能となることから本バス上に配置した．

これにより，MPU はバスアビトレーション回路を制御することで，前述の MBE を介して，IM バス，D バス，F バス上のデバイスをアクセスできるように設計した．

その他の主要構成品の役割について以下に示す．

① SRAM（256 キロバイト）
- システムロード後の MPU ファームウェアの格納エリア
- システム OS と MPU ファームウェア間の命令パケットの受渡エリア
- MPU ファームウェアで使用する IO バッファエリア
- MPU ファームウェアの作業エリア

② EEPROM（256 キロバイト）
- モニタプログラム格納エリア

モニタプログラムはシステムロードされる前に実行される MPU ファームウェアで，主な機能として以下のものがある．

POC 機能（電源投入時のシステム自己診断機能）
モニタコマンド機能
IPL 機能（システムブート機能）

③ MK48T02TOD[8]-Chip（2 キロバイト）
- システム上のカレンダ情報の保持
- システムのブートパス，ルートデバイス，ダンプデバイスの保持
- システムの主要構成の保持

④ PROM
- イーサネット物理層のアドレス（イーサネットアドレス）の保持

2) D バス……このバスの主要構成品は，NCR*社の 53C700SIOP[9] (SCSI IO Processor) である．

53C700SIOP は，インテリジェントな SCSI ホストアダプタであり，SCSI・SCRIPTS プロセッサ，SCSI コアロジック，SCSI インタフェース，DMA コアロジック，システムインタフェースロジックを内蔵している．SIOP は，通常の動作モードでは，システムメモリ上に組み立てられた SCSI・SCRIPTS をフェッチし，

* NCR は米国 NCR 社の登録商標であり，53C700SIOP，SCSI・SCRIPTS は同社の商標である．

その命令により内部レジスタおよびDMAをセットアップする．DMAは32ビット転送機能を持ち，さらに効率の良いバーチャルシステムの構築では必須となるスキャッタ/ギャザ機能も有している．

このような機能を持つSIOPを採用した最大の目的は，UNIXシステム上でのディスクアクセスの高速化と，SCSIプロトコルによるソフトウェアのオーバヘッドを最小限にするためであり，この目的を達成するために，

① バスアビトレーションのシンプル化と高速化，

② バス専有による他デバイスへの影響を最小限にとどめること，

が必要であった．そこで，SIOPを他のデバイスと共存しない独立したバス(Dバス）上に配置し，メモリとDバス間の高速転送を実現した．

3) Fバス……このバスの主要構成品は，Xilinx*社のXC 3042[10]（プログラマブル•ゲートアレイ）とその他システム全体の状態を保持するレジスタ群からなる．

XC 3042は，システム立上げ時にUバスからMPUにより自動的にプログラムされ，システム立上げ後はインタラプト・コントローラとして機能し，内部にインタラプト•マスクレジスタ，インタラプト•ロウレジスタ，インタプロセッサ•コミュニケーションレジスタ，プロセッサ・プライオリティレジスタを備えたものとなる．

このコントローラは，ノンマスカブル割り込み，クロック割り込み，VMEの各レベルの割り込み，MPUからの各レベルの割り込みの状態を検出しMC 88100 CPUに対し割り込み信号を発生させる．CPUには，割り込みレベルの概念はなく，マスク可能な割り込みが1種類のみであるため，割り込みを受け付けたCPUはこのコントローラのインタラプト・ロウレジスタを読み出すことで，外部割り込みの種類を判別する．また，UNIXカーネルによる割り込みレベルの処理に対処するためにインタラプト・マスクレジスタを用意した．

このようにMC 88100 CPUを使用したシステムでは，インタラプト•コントローラは必須であり，また割り込み処理時には必ずコントローラのアクセスを伴うため，このコントローラのアクセス時のロスを防ぐことが非同期に発生する割り込みの処理速度の向上につながる．

本シリーズではインタラプト・コントローラを独立したFバスに持つことにより，他のバスのアクセス状態に依存することなく，割り込み処理を行えるよう設計した．またシステム全体の情報を保持するレジスタ群の中には，電源異常，ファン異常，温度異常等の情報を持ち，これらの割り込み時の詳細情報として使われる．

### 3.5 VMEインタフェース

S 8400シリーズのVMEインタフェースボードの開発にあたっては，

① 市販VMEオプションボードの搭載を可能にすること，

② 最大40メガバイト/秒以上のデータ転送能力を持つこと，

等を目標にして開発を進めた．

しかし，本ボードは前述の通りオプション扱いとすることからシステム的に疎結合の接続とし，IMバスを介してシステムと接続することとした．そのため，VMEバス

* Xilinxは米国Xilinx社の登録商標であり，XC3042は同社の商標である．

のバスコントローラとしての機能の他に IM バスに対してのマスタ/スレーブ機能および VME バスのマスタ/スレーブ機能を合わせ持つこととした．このボードは，IM バスと VME バスのバスプロトコルの変換を行うと同時に，非同期バスである VME バスの転送速度のばらつきが IM バスに直接影響を与えないようにするため，内部に 32 エントリの FIFO を持たせることでブロック転送時のデータ転送効率を上げた．この結果，PAL による高速制御回路との組み合わせで，最大 40 メガバイト/秒のデータ転送速度（ブロック転送時）が実現できた．

市販 VME オプションボードの搭載を可能とするため，VME オプションスロットは物理的には 3.1 節でも述べたようにユーロボードサイズの 9 U×400 mm を採用し，コネクタ P 1，P 2 に関してはピンアサイメントおよび電気的仕様とも IEEEP1014/D1. 2（VMEbus Revision C. 1）に完全に準拠させた．また，表 6 に示すような機能を本ボードに組み入れることにより，機能的にも市販ボードに十分対応できるものとした．とくに，バスアビタの機能を固定優先順位方式としたことで，多種のオプションボードを同時に搭載した場合の，バス取得の優先順位を明示的に決められるものとした．

**表 6　VME インタフェースボードの機能概要**
Table 6　Functional description of VME interface board

| | | |
|---|---|---|
| VME マスタ | アドレスオプション | A 32，A 24，A 16 の提供 |
| | データオプション | D 32, D 16, D 8 (Even および Odd) の提供<br>非整列データ転送機能の提供 |
| VME スレーブ | アドレスオプション | A 32，A 24 の提供 |
| | データオプション | D 32, D 16, D 8 (Even および Odd) の提供<br>非整列データ転送機能の提供<br>ブロック転送機能の提供 |
| VME バスタイマ | | 64 マイクロ秒のバスタイマ |
| VME バスアビタ | | バスリクエスト 0〜3 に対応した固定優先順位方式（プライオリティ方式） |
| VME インタラプトハンドラ | | インタラプトリクエスト 1〜7 に対応した優先割り込み方式 |

これは，オプションボードを利用した場合のシステム構築には不可欠のものである．

P 3 のピンアサイメントは，他社機との互換性を保つデザインとすると同時に，B 列の未定義ピンを使用して，自社開発の VME オプションボード搭載時の VME 上の自動ボードアドレッシングと，IOP ボード上のファームウェアによる自動ボード認識/判別を実現し，さらに POC（電源投入時のシステム自己診断機能）実行時の診断範囲を VME オプションボードまで拡大することを可能とした．

ユーロボードサイズの 6 U×160 mm のボードの搭載については，9 U×400 mm のサイズに変換するための汎用アダプタを開発することで対応を図った．

## 4. 性 能 検 証

S 8400 シリーズ上で，高速・高性能を実現するためのハードウェア面から見た工夫および実現形態を前項で述べてきた．

その結果，具現化した該機がRISCプロセッサの特徴であった，

① データや命令語をパイプライン処理することで実行中の命令を停止させない，

② 演算や使用頻度の高い命令語が使用するレジスタの数を多くしている，

等の機能を生かし，RISCプロセッサの持つ能力（1クロックサイクル・ノーウェイトで一つの命令語を実行できる）を十分に実現できたかを検証するために，一般的に使用されているベンチマークテスト(Dhrystone*，Whetstone**，Linpack*** 等)をシングルタスクモードでの性能検証用ツールとして使用し評価した．

この結果は，該機において狙いとした性能，公称値44 MIPS**** (33 MHz版でのDhrystone換算値）をクリアし，さらにコンパイラの最適化により，その上をいくデータを得ている．

## 5. おわりに

新S 8400シリーズワークステーションは，ハードウェア面から見て，モトローラ社製プロセッサM 88000ファミリのアーキテクチャを忠実に守り，加えて内部ではRISCチップとCISC（Complex Instructions Set Computer）チップの特徴を生かした組み合わせ技術，最新の超LSIテクノロジを取り込むことでの利用技術，随所に盛り込んだ高速化のための工夫およびバス構成で代表されるようなアーキテクチャ等から高速・高性能を達成できたものであると確信する．

今後の課題としては，高速化のみならず，製品の低価格化・高機能化・差別化による製品の品揃えが求められており，それらをタイムリに用意することが開発者の使命であると考えている．

---

参考文献 [1] 日本モトローラ社 半導体事業部，「MC88100 Technical Summary 32ビット第三世代RISCマイクロプロセッサ」，FEB/'90.
[2] 日本モトローラ社 半導体事業部，「MC88200 Technical Summary 16キロバイト・キャッシュ/メモリ管理ユニット（CMMU)」，FEB/'90.
[3] Advanced Micro Device, Am29C982/Am29C983 Multiple Bus Exchange Handbook.
[4] MOTOROLA, MC68020 32-bit Microprocessor User's Manual.
[5] Zilog, Z8030/Z8530 SCC Technical Manual.
[6] Zilog, Z8036/Z8536 CIO Unit Technical Manual.
[7] Advanced Micro Device, Local Area Network Controller Am 7990(Lance) Technical Manual.
[8] Thomson Components/Mostec Corp. MK48T02TOD Data Manual.
[9] NCR, 53C 700 SCSI I/O Processor Data Manual.
[10] Xilinx, The Programmable Gate Array Data Book.
[11] 山下英男監修，日本ユニバック総合研究所編，「共立総合コンピュータ辞典」，1月25日1990.

---

* Dhrystone：文字列処理および整数演算能力やコンパイラ性能の測定を主体としたベンチマークテスト
** Whetstone：科学技術計算能力に比重をおき，浮動小数点演算を中心としたベンチマークテスト
*** Linpack：とくにベクトル演算中心の科学技術計算用の能力を表す場合の評価尺度に使用
**** MIPS：計算機の性能を表す指標の一つとして一般的に広く使用されている単位で，Million Instructions per Second の略であり，1秒間に実行される命令数を百万単位で表す．

**執筆者紹介**　吉　田　欣　司（Kinji Yoshida）

昭和 30 年生．51 年国立石川高専電気工学科卒業．同年日本ユニシス(株)入社．90/800，1100/60，70 ハードウェア保守業務担当．63 年より UNIX 関連のハードウェア機器開発に従事．現在，オープンシステム推進本部開発管理部に所属．

唐　下　　勉（Tsutomu Toge）

昭和 35 年生．57 年岡山理科大学理学部電子理学科卒業．同年日本ユニシス(株)入社．BTOS/CTOS の日本化ハードウェアのプラットホームの開発等を担当．平成 2 年より S 8400 シリーズの関連機器開発に従事．現在，オープンシステム推進本部開発管理部に所属．

# 統合オフィス・システム EXOS

## 1. EXOSとは

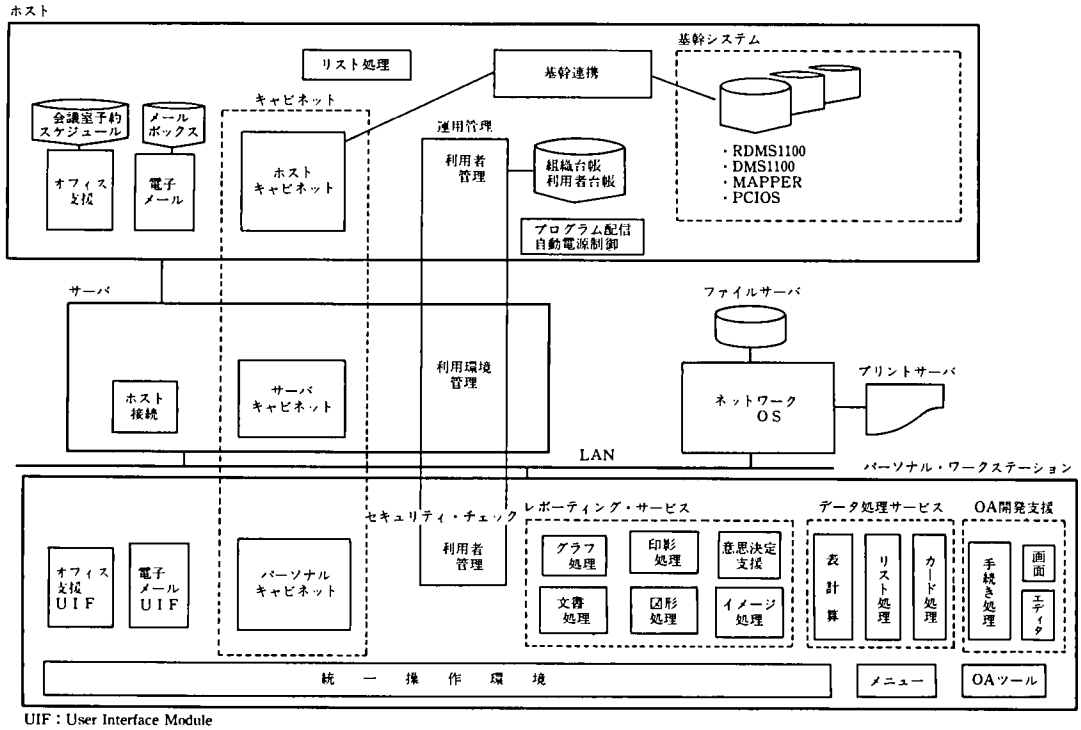
EXOS (EXcellent Office System) はオフィスにおける「文書作成/保管」,「データ処理」,「コミュニケーション」,「データベース検索」等の処理を支援するための統合オフィス・システムである. EXOSの概要を説明する前に「統合オフィス・システム」とは何か, 今なぜそれが話題になっているかについて簡単に触れておきたい.

1980年代前半における第1次OAブームをきっかけに個人レベルのオフィス業務の機械化が進展した. しかしながらオフィスでの情報活用は, 個人が情報収集・蓄積・加工しているだけでは省力化, 合理化の効果が現れてこない. たとえば, 一度コンピュータに入力された情報は別の人に伝えられたり, 共通のデータベースとして活用することにより, 個人で活用した場合に比べて, より一層オフィス全体の情報収集力, 情報活用力が向上することとなる.

1980年代の前半に比べて, 国際化の進展や企業経営環境が変化した現在では, 第1次OAブームの時以上にオフィスの情報活用力の向上, 知的生産性の向上が求められている.

一方, ISDNやLAN等のネットワーク環境, マイクロエレクトロニクスやコンピュータ技術等のシーズの面での環境も1980年代前半とは一変している. OAについても, 先進的な企業においては, 単にオフィスの省力化・合理化だけではなく, 情報化によるオフィスの生産性の向上, あるいは知的作業支援のツールとしての「統合オフィス・システム」が求められている.

ところで, オフィスの生産性, 知的作業支援を行うためには, ワープロや単体のパソコンを利用した, 個人レベルのデータ処理, 単能型の情報シ



UIF : User Interface Module

図1 EXOS概念図

表1　EXOSの主なサービス機能とソフトウェア群

| | | |
|---|---|---|
| 主なサービス機能 | | |
| データ処理サービス | EXCALC | 表計算データ処理 |
| | EXLIST | リスト型データベース処理 |
| | EXCARD | カード型データベース処理 |
| レポーティング・サービス | EXWORD | マルチメディア文書処理 |
| | EXDRAW | 図形処理（ベクトル・イメージ） |
| | EXPAINT | ドット・イメージ処理 |
| | EXCHART | グラフ処理 |
| | EXSEAL | 印影処理 |
| | EXFCS | 意思決定支援 |
| オフィス支援サービス | EXMAIL | 電子メール |
| | EXDOCM | 文書管理 |
| | EXTOSS | 会議室予約，スケジュール管理 |
| OAアプリケーション開発支援サービス | EXMILD | OA処理システム化 |
| | EXSD | スクリーン・デザイナ |
| | EXED | テキスト・エディタ |
| 基本機能／付加機能 | | |
| 統一操作環境 | EXBASE | ウィンドウ・コントロール<br>メニュー，OAツール<br>キャビネット・コントロール<br>利用者管理，利用環境管理 |
| 基幹システム連携 | EXMDL | 基幹データベース連携 |
| 運用管理 | EXOPS | プログラム配信，自動電源制御等 |

ステムではなく，情報の収集，分析・加工・文書化，管理・保管，コミュニケーションといった一連のオフィス業務を一つのワークステーションで実行できる環境が必要である．また，このような「統合オフィス・システム」では実行環境を構築したり，運用を利用者自らができるいわゆるエンドユーザ・コンピューティングの視点が重要である．

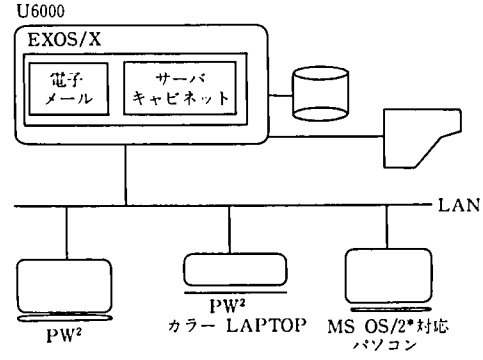
EXOSは上記で述べた「統合オフィス・システム」の一つであり，単に文書の作成・保管やデータの加工を支援するだけでなく，データの共有化，コミュニケーション，等のオフィス業務を統合的に支援する機能を持ったシステムである．

## 2. EXOSの概念

EXOSの概念を図示すると図1のようになる．

また，EXOSの主なサービス機能とソフトウェア群は表1のようになっている．

なお，UNIXを利用したシステムではUNISYS U6000シリーズに電子キャビネットを持つことができる．さらにU6000シリーズをホストに見立てた電子メールも可能である．ワークステーションとして従来から使用しているPW²500に加えてカラーLAPTOPやAX対応のパソコンにも順次対応していく予定である（図2）．




*MS OS/2は米マイクロソフト社の登録商標である．
**UNIXオペレーティング・システムはUNIX System Laboratories, Inc. が開発しライセンスしている．

図2　UNIX**を使用したEXOS

## 3. EXOS利用のポイント

EXOSは多くの機能を持ったソフトウェアの集合であり，最初はある部分だけを利用するというケースが多いと考えられる．ここでは，EXOSの機能の一部を利用することにより，今までのやり方よりもさらに効率的で生産性が向上すると考えられるポイントをいくつかあげてみたい．

1)　ソフトウェア間のデータ連携……EXOSではデータ処理サービス，レポーティング・

サービス等多くの機能やソフトウェアを組み合わせて活用している．これらのソフトウェア間でのデータ受け渡しがスムーズにいくようになっているのがEXOSの特徴の一つである．この点について簡単に触れておきたい．

一番簡単な例として，表計算ソフトEXCALCとグラフ処理ソフトEXCHARTとの連携がある．EXCHARTは自分自身でグラフのデータ作成機能を有しているが，一般的には表形式のものからグラフを作成することが多い．EXCALCでグラフにしたいデータの範囲の指定をして棒グラフ，折れ線グラフ，レーダーチャート等いろいろなグラフを描くことができる．EXCALCの表データを変えることにより，連動してグラフの表示が変化するようになっている．これらは，マルチウィンドウ，マルチタスクの機能を活かしたものの一つと言えるであろう．

この他にもEXCALC⟷EXLIST間のデータ受け渡しは，3)で述べるように，クリップボードと呼ばれるメモリ上の作業領域を利用した切り取り/貼り付けによる方法，あるいはディスク上のファイルを仲介とする方法が準備されている．

上記の方法はデータを文書に取り込む場合にも利用できる．文書ソフト(EXWORD)には，ファイル読み込みの機能によりワープロソフトからのテキストデータを渡すことが可能である．

イメージデータ間では図形処理ソフト(EXDRAW)で作成したイメージや画面に表れているイメージを切り取り/貼り付け等の方法によりEXPAINTのオブジェクトとして簡単な操作で取り込むことができる．

2) ISを利用したホスト連携……EXOSを支える通信機能としてIS（インフォメーション・システム）と呼ばれるソフトウェアが提供されている．ISを使うことによりMS OS/2のプレゼンテーション環境でホストシステム（UNISYS 2200•1100シリーズ）と通信ができる．これによりPW²をホストの端末として利用できることは勿論であるが，MAPPERとEXCALCを一つの画面で同時に動かすと言うことが可能となる．MAPPERデータを切り取って，EXCALCの表に貼り付けるというようなことが簡単に行える．

また，デマンドの処理（TSS処理）結果を文書に貼り付けること等も可能である．さらに，PW-APMENUを使用することにより，ISでホストに接続する場合のセッション番号の入力等をあらかじめ設定することができ，メニューを選ぶだけでMAPPERランやオンラインのプログラム等を複雑な操作をせずに実行させることが可能である．

3) MAPPER 1100データの取り込み，加工，加工データのMAPPER 1100への反映……MAPPERデータをEXCALCに取り込み，データ加工ができると非常に便利である．MAPPERデータをEXCALCに取り込む方法として以下の二つの方法がある．

① 読み取り，貼り付けによるデータ授受

1)で説明したとおり，文字データの他にグラフ情報もイメージとして読み取りが可能である．

② ファイル経由でのデータ授受

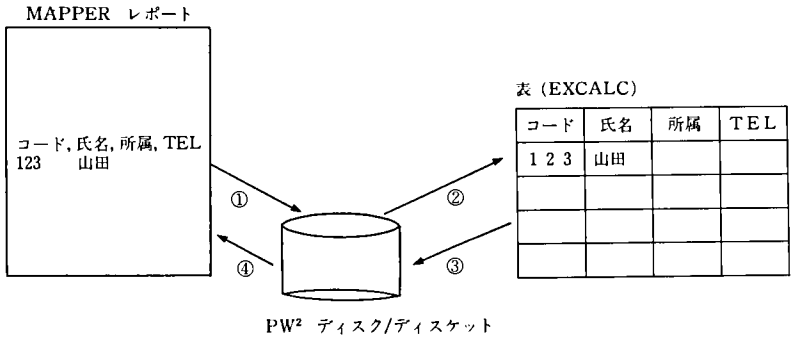
MAPPERには利用しているワークステーションのディスク，ディスケットにデータを送受信するための転送機能がある．

この機能を利用してディスクに書き込んだデータをEXCALCの「ファイル読み込み」機能を使うことにより，表の内部に取り込むことができる．

なお，この方法はISを使わないMAPPERでも可能であり，ワークステーション上のEXOS (EXCALC) とUTS (Universal Terminal System) エミュレータを利用して行うことができる．また，EXCALCからMAPPERへのデータ授受も同様の操作で行うことができる（図3）．

また，MAPPERとEXLISTでのデータ授受についても，EXCALCと同様のことが可能である．また，EXCALCのマクロでファイルからの読み込みができるので自動的・定型的な読み込み処理も可能である．

4) オフィス業務の定型化……EXOSでは，オフィス業務の中で何度か同じ操作を行う場合にはその操作手順自体をシステムに覚えさせて，2回目からは手順の名前を選び出して指定すれば再現できるようないくつかの工夫がされている．それらの工夫は概略図4のよう




① EXOSDB・・EXOSとMAPPERのファイル転送用ラン
② EXCALCのファイル読み込み機能
③ EXCALCのファイル書き込み機能
④ EXOSDB・・①と同じMAPPER　ラン

**図3　MAPPER⟷ EXCALCのデータ授受**

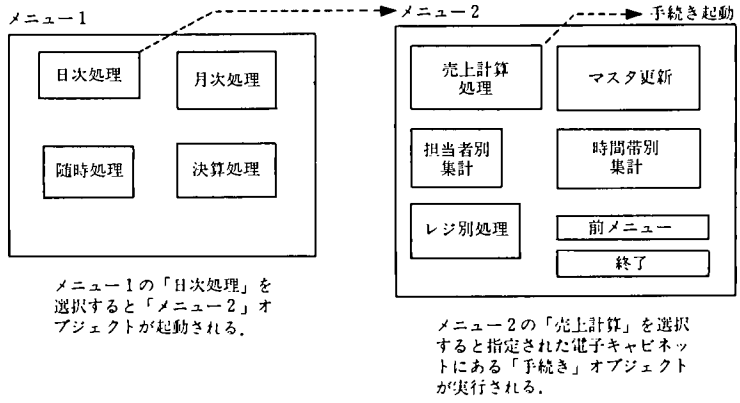


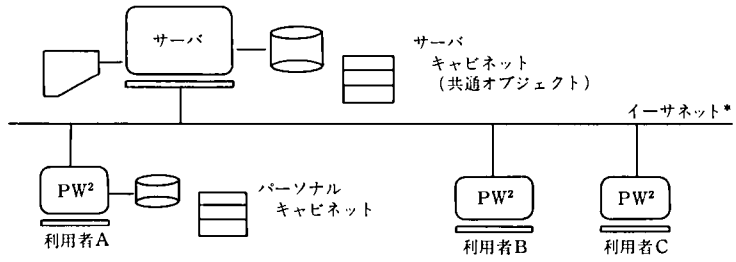

**図4　メニューからの定型的手順を起動する例**

になっている.

ここでは，非常に簡単な例を紹介したが，メニューからはオブジェクト（手続き，メニュー，その他）およびプログラム（電子メール，IS，その他プレゼンテーション対応プログラム，非プレゼンテーション対応プログラム）を動かすことができる．また，メニューの特殊なケースとして，EXOSが起動された時の画面の一番上の行には「UIMボタン」が表示されるが，このボタンからオブジェクト，プログラムを動かすことができる．また，特定のフォルダを開くこともできる．さらにサインオンや電源オンと連動して自動的にボタンに設定されたオブジェクト等を開くことも可能である．

手続きからメニュー・オブジェクトや別の手続きオブジェクトを起動したり，プログラムを起動することも可能である．

5) グループによるデータ利用……EXOSは単独で利用するよりも，グループで利用する方がより威力を発揮すると言えるであろう．たとえば，ある研究を数人で行いレポートを共同で作成したとする．個々で作成した文書や表，グラフ等をサーバの共用キャビネットに保管して置くことにより，互いに参照し合うことができる．あるいは一つの文書を数人で作成・加工したり，複数のレポートを一つにしたりすることが容易にできる．また，個々のデータを集計してグループとしてのデータとしてまとめることができる（図5）．




*イーサネットは米国Xerox社の登録商標である．

図5 グループ利用例

## 4. EXOSの特徴

EXOSは，1台のワークステーションですべてのオフィス業務を支援することができ，かつコンピュータ・システムに馴染みのない人でも利用しやすい環境を提供することを目指している．

この目標実現のために，以下のような特徴を持ったシステムとなっている．

① 統一されたオブジェクト指向の優しいユーザ・インタフェース
② MS OS/2，PM（プレゼンテーション・マネージャ）ベースのマルチタスク，マルチウィンドウ処理
③ 階層化された電子キャビネットによるオブジェクト管理
④ 多彩なデータ処理サービス
⑤ マルチメディア文書に対応したレポーティング・サービス
⑥ 対話型OAアプリケーション開発支援サービス
⑦ 基幹システムとのデータ連携
⑧ 電子メール，会議室予約等，豊富なOA機能

1) 優しいユーザ・インタフェース……オフィスで利用されるシステムのまず第1の条件は，使い勝手の良いシステムであることが求められる．すなわち簡単に覚えられ，わからないところはシステム自身が親切にガイドしてくれたり，援助（ヘルプ）してくれたりすることが必要である．

一度作成した文書や図形，グラフ，表等(通常これらを目的のものと言う意味でオブジェクトと呼ぶ)は名前を付けて保存しておけば，二度目からはその名前を選び出すだけでよい．その文書や図形等を作成したソフトウェアは自動的に呼び出されるといったことは基本的な機能として欲しいものである．

これは，文書や図形等を保存する時に，作成したソフトウェア名，オブジェクトの状態(画面のサイズや更新状態等)をその文書や図形の属性情報として覚えている(カプセル化)ことにより可能となる．このように目的のもの（オブジェクト）を指し示すことにより，カプセル化した処理まで同時に行わせようとする処理方法をオブジェクト指向(操作)と呼び，EXOSの大きな特徴となっている．

したがって，EXOSの文書作成用ソフトウェアがEXWORDであると言うこと等は，利用者は知らなくてもすむ．また，EXOSは文書や表等を扱う多くのソフトウェアが集まってシステムを構成しているが，オブジェクトを取り出したり，保存したり，印書したりするような基本的な操作は統一した仕様に基づいて作られている．したがって一つのソフトウェアの操作を覚えると，他のソフトウェアでも基本的な操作は類推して操作することができる．

2) マルチタスク，マルチウィンドウ処理……これらの操作を1台のワークステーションで実行するためには，同時に二つ以上タスクが動けること，同時に二つ以上の作業のための表示ができることが前提となる．

EXOSでは，これらのマルチタスク，マルチウィンドウ機能をオペレーティング・システムとして使用可能な米マイクロソフト社のMS OS/2 PMを採用することで実現している．

3) 電子キャビネットによるオブジェト管理……作成した文書やグラフ等のオブジェクトを保管するための保管庫として，EXOSではキャビネット・システムを採用している．キャビネット・システムは一般の文書等を保管する方法として広くオフィスで利用されている．キャビネットは引出し(ドロワ)，フォルダ等から構成されている．EXOSでは，個人レベルのオブジェクトはワークステーション上のキャビネットに，共有して利用するオブジェクトはサーバまたはホスト上のキャビネットに保管できるようにしている．キャビネットがどこにあっても利用者の操作はまったく同一であり，キャビネット間でのオブジェクトの複写，移動も極めて簡単な操作で行うことができる．

また，キャビネットやドロワの作成，削除，セキュリティ等の機能も完備している．

4) データ処理サービス……一般的なデータ処理機能としては，リレーショナル型データベース処理（以下データベース処理）と表計算(スプレッドシート)処理が代表的なものである．データベース処理としては，必要なデータの抽出（検索)・集計・並べ替え・データベースの結合・更新等が代表的な処理であろう．また，表示形式として台帳のように一覧形式で表示する方法と，1件ごとに詳細な内容を一画面で表示する方法が欲しいところである．

EXOSでは台帳一覧形式のデータベース処理（リスト処理：EXLIST）と一画面で1件の処理（カード処理：EXCARD)のどちらの処理も可能としている．

表計算処理はLotus 1-2-3* 等で周知の処理であるが，EXOSではEXCALCとして提供している．表データからは簡単な操作でグラフを表示することが可能である．

これらのリスト・オブジェクトや表オブジェクトは，ホストの基幹データベースやMAPPERデータ，あるいはDOSフォーマットのデータとのデータ交換が可能となっている．

5) レポーティング・サービス……情報システムが進展しディスプレイ処理が使われるようになっても，紙による表現力にはなかなか及ばないところも多い．したがって，統合オフィス・システムの中間的アウトプット，あるいは最終的アウトプットとして，いかにきれいな報告書を印書できるかがポイントとなる．

また，ホストのデータベースやキャビネットに保管された文書，図形，表等のさまざまなオブジェクトを組み合わせた，いわゆるマルチメディア文書を簡単に作り出すことができると非常に便利である．

マルチメディア文書を構成するオブジェクトとして，ベースとなる「文書オブジェクト」，ベクトル情報で構成される「図形オブジェクト」，ドット情報としての「イメージ・オブジェクト」，棒・折れ線・レーダーチャート等の「グラフ・オブジェクト」，表計算の結果としての「表オブジェクト」等がある．

これらのオブジェクトは単独に作成・加工ができ，キャビネットに保管できることはもちろん，上記のように文書の一部として関連付けられ一つのマルチメディア文書を作ることができる．

このように関連付けられたオブジェクトは参照型と呼ばれ，グラフ・表等，元となったオブジェクトが更新されていた場合，文書を表示すると最新の情報が表示されることとなる．

もちろん，更新情報が反映しない方が望ましい場合もあり，そのような場合には取り込み型という方式をとることができる．参照型を用いると月報処理等でグラフや表を参照した文書を作っておけば，その文書を開いただけで当月のデータ処理結果の表やグラフが表示されるので，あとは文書を修正すれば完成してしまう等といったことも可能である．

文書処理システムであるEXWORDはビットマップをフルに利用したワープロであり，6ポから76ポまでのサイズの異なる文字の扱いや簡単な図形を描いたりする機能も有している．

6) 定型的オフィス業務の手続き化……オフィス業務の中でも定型的な処理が数多く存在する．たとえば，週報や月次報告書の作成，あ

* Lotus 1-2-3は米ロータス・ディベロップメント社の登録商標である．

るいは商品を抽出する場合，コードを変えて何度も操作を実施すること等である．

このように何度も行う処理は，やはり手順(処理手続き)そのものをシステムに覚えさせておき，名前をつけて保存しておけば，名前を指定するだけで実行できるようにしておきたい．

そのようなオフィス業務の定型化のためのオブジェクトが「手続きオブジェクト」であり，その手続きオブジェクトを作るための機能がOAアプリケーション開発支援サービスと呼ばれるものである．キャビネット内のオブジェクトを開いたり，データベースのデータを抽出したり，メールを出したりすることができる．当然，手続きとして条件指定や利用者からのデータ入力が必要な場合がある．また，メッセージを出したり，対話処理にまかせたいケースもある．

これらの「手続きオブジェト」を対話形式で作成・変更するのがEXMILDであり，実行時点で利用者との対話をするための画面を作成するツールがEXSDである．

対話で作成された手続きオブジェクトは日本語の文書として生成されるので，それを読めばどんなことを行っているのか判断できるようになっており，あらためてドキュメントを作成する必要はない．手続きオブジェクトは名前付けをしてキャビネットに保存しておき，再利用できることは言うまでもない．

なお，表計算機能 (EXCALC) の中では，CALCの機能として演算機能を持っているのでこれを活用することにより，いろいろな計算処理ができる．また，マクロ機能と手続き処理とを連動することも可能である．

また，これらの手続きを定型的なメニューから実行するために，「メニュー・オブジェクト」を作成するツールがEXOSの基本機能として準備されている．メニュー・オブジェクトのメニューの一要素として，レポーティング・サービスで作成したイメージ・オブジェクトや図形オブジェクトを利用できるので，カラフルでユニークなメニューを作成することが可能である．

このような，メニュー・オブジェクトにより，階層型メニューを作成したり，日報等の特別なオブジェクトを開いたり，特定の手続きオブジェクトを実行させたりすることができる．また，UTSエミュレータやIS等のプログラムを起動してホストの既存アプリケーションを実行することができる(図4参照)．

7) 基幹システムとのデータ連携……「統合オフィス・システム」の機能の中でも重要視されているのが，ホストのデータを取り出して加工する機能である．EXOSでは，UNISYS 2200・1100シリーズの基幹データベースからホストのキャビネットにデータを取り出す機能を準備している．キャビネットに取り出されたデータは，EXOSのデータベース処理用ソフトウェアEXLISTによりさまざまな加工が行える．

もちろん，加工したデータを基幹データベースに反映させるインタフェースも準備されている．これらの基幹データベースの取り出しはEXMDLと呼ばれるソフトウェアで行うが，取り出しの指定はSQLに準処したインタフェースにより行っている．

また，MAPPERデータベースとEXLIST, EXCALC等とのデータ授受は，切り貼り機能やファイル転送機能を応用して容易にできるようになっている．

8) 汎用的オフィス業務の支援……コミュニケーションも「統合オフィス・システム」の重要な課題である．電子メールはコミュニケーションの手段として最も基本的なものであり，EXOSでもEXMAILとして電子メール機能を提供している．文書や図形，イメージ等のオブジェクトはもちろん，マルチメディア文書やその場で作ったメッセージの発信等もできるようになっている．また，承認の証しとして文書に印影を押下することも可能である．

メールは利用者単位に宛先を指定することができ，利用者は自分が見たい時に見ることができる．また，どのワークステーションからでも見ることが可能である．たまたま，ワークステーションで文書作成やデータ加工を行っている時に自分宛のメールが発信された場合には，使用しているワークステーションにメールが届いたことを知らせる「メールアイコン」が点滅する．

また，EXOSでは，「会議室の予約」や「個人・グループ単位のスケジュール管理」，「文書管理」といったどこのオフィスでもあるような汎用的な処理を「オフィス支援サービス」として提供している．

9) UNIXサーバ機能……UNIXをサーバとするEXOS/Xが1991年11月にリリースされた．2章の「EXOSの概念」でも簡単にふれたが，EXOS/XではU6000シリーズ上に電子キャビネットを持つことができる．また電子メール機能を利用することが可能である．

PW²上のEXOSの機能であるレポーティングサービス，データ処理サービス，OA開発支援サービスはそのまま利用することができるので，EXOSの世界が一段と広がることになる．

■　■

# RISC ワークステーション S 8400 シリーズ

近年，急速に台頭してきたRISC（Reduced Instruction Set Computer：縮小命令セットコンピュータ）マイクロプロセッサはワークステーションのCPUとして採用され，ダウンサイジングの一方の旗頭であるワークステーションの拡大の大きな原動力となった．

RISCマイクロプロセッサは命令セットを使用頻度の高い基本命令にしぼり込み，命令数を減らすことにより従来型のCISC（Complex Instruction Set Computer：複合命令セットコンピュータ）マイクロプロセッサより高速性を実現している．

日本ユニシスでは，ワークステーションに求められている市場要求である，①コスト・パフォーマンス，②高速・高性能，③標準化，④システムの拡張性，⑤移植性，⑥マンマシン・インタフェースに対応すべく，またEWS（エンジニアリング・ワークステーション）市場におけるUNISYSワークステーションの地位を確立すべく，1991年4月にEWS“S 8400”シリーズを発表した．

S 8400シリーズにはS 8421（25 MHz）/S 8431（33 MHz）の2モデルがあり，CPUはモトローラ社の最新のRISCマイクロプロセッサMC 88100*を採用した．また，デスクサイド型のシステム・ユニットには，業界標準の種々のインタフェースを実装しており，各種周辺機器の接続を可能としている．

ソフトウェア環境では，AT&T SYSTEM Vリリース4.0 UNIX オペレーティングシステム**を基本に日本語機能等の充実が図られており，ウィンドウ環境として，X window***/Motif***とX. desktop***を提供する．さらに通信ソフトウェア，グラフィックス・ライブラリ，4GL，データベース等のソフトウェアの提供も行う．

なお，S 8400シリーズは高速3次元グラフィックス・アクセラレータを搭載することによりGWS（グラフィックス・ワークステーション）としても利用することができる．

## 1. S 8400シリーズのハードウェア概要

### 1.1 モトローラ社MC 88100RISCプロセッサの採用

S 8400シリーズのCPUは，MC 88100マイクロプロセッサとキャッシュ・メモリ・コントロールユニット（CMMU）MC 88200から構成される．

MC 88100は浮動小数点ユニット（FPU）を内蔵したマイクロ・プロセッサであり，実行ユニットは複数段のパイプラインを備え，ほとんどの命令を1クロックで実行することができる．また，MC 88100とMC 88200を1 CPUとして，複数CPUが共通の主メモリをアクセスするマルチプロセッサ対応も可能である．

S 8400シリーズでは25 MHz/33 MHzのクロック周波数の2機種が用意され（VAX MIPS換算****で33 MIPS/44 MIPSの性能がある），キャッシュ・メモリは32キロバイトの命令・キャッシュと32キロバイトのデータ・キャッシュの計64キロバイトを実装している．また，マルチプロセッサ対応モデルも予定されている．

SC8400シリーズ筐体概要

* M 88000, MC 88100, MC 88200はモトローラ社の登録商標である．

** UNIXオペレーティングシステムはUNIX System Laboratories, Inc.が開発し，ライセンスしている．

*** X Windowは MIT，Motifは Open Software Foundation，X. desktopは英IXI社の登録商標である．

**** VAX MIPS：エンジニアリング・ワークステーションで標準的に使用される処理速度の表現方法で，VAX-11/780の性能を1とした時の相対性能比である．

表1　S8400シリーズのシステム・ユニット

| モデル | | S8421 | S8431 |
|---|---|---|---|
| 型式 | | デスクサイド型 | |
| プロセッサ | | MC88100 | MC88100 |
| | クロック周波数 | 25MHz | 33MHz |
| | キャッシュ・メモリ | 64KB | 64KB |
| 浮動小数点演算機構 | | MC88100に内蔵 | |
| 主記憶容量 | | 8MB～256MB | |
| 磁気ディスク容量 | | 380MB～5.2GB(アンフォーマット) | |
| 基本装備<br>IOPインタフェース | | ・SCSIインタフェース　1ポート（ANSI X3.131準拠）<br>・シリアル・インタフェース　8ポート（ASYNC 6ポート，ASYNC/SYNC 2ポート）<br>・LANインタフェース　1ポート（IEEE802.3準拠）<br>・パラレル・インタフェース　1ポート（セントロニクス準拠） | |
| IOバス | | VMEバス，SCSIバス | |
| VMEスロット | | 4 | |

### 1.2　システム・ユニット

S 8400 シリーズでは，幅 394 mm，奥行 635 mm，高さ 599 mm のデスクサイド型の筐体を採用（日本の事務机の下に入るように設計されている）しつつ，周辺機器の搭載能力を高め，コストパフォーマンスの良い EWS として提供される．標準搭載でセントロニクス・インタフェース，SCSI インタフェース，RS 232 C インタフェース（8 ポート），イーサネット*LAN，380 メガバイトディスク，16 メガバイトメモリ，および VME インタフェース・ボード用として 4 スロットが用意されている．

メモリは 2 ウェイ・インタリーブ方式を採用し高速処理を実現した．また，CPU ボード上にメモリを最大 256 メガバイトまで実装できるため，専用の拡張メモリスロットを必要としない．磁気ディスクは 380 メガバイトから 5.2 ギガバイトまでの拡張が可能である．

表 1 にシステム・ユニットの仕様，図 1 にハードウェア体系を示す．

### 1.3　豊富な VME インタフェース・ボードと周辺機器

S 8400 シリーズでは，拡張バスとしてワークステーションでは標準の VME バスを採用し，以下に示す VME インタフェース・ボードを提供する．また，これらのインタフェース・ボードに接続される各種周辺機器も提供する．

1)　各種 VME インタフェース・ボード

①　増設 SCSI インタフェース・ボード
　ANSI X 3.131 準拠の SCSI インタフェース・ボード．SCSI インタフェース用機器増設用に使用

②　増設 LAN インタフェース・ボード
　IEEE 802.3 準拠の LAN インタフェース・ボード

③　コミュニケーション・コントローラ・ボード
　RS 232 C，V.35 等のインタフェースを提供する同期式コントローラ・ボード

④　16 ビット・パラレル・インタフェース・ボード
　DR 11-W**準拠のパラレル・インタフェース・ボード

⑤　32 ポート・シリアル・インタフェース・ボード
　非同期式シリアル・インタフェース．32 ポートを提供するボード

2)　各種周辺機器

①　380 メガバイト/780 メガバイト/1.6 ギガバイト（予定）磁気ディスク装置

②　磁気テープ装置

---

* イーサネットは米 Xerox 社の登録商標である．

** DR 11-W は，Digital Equipment Corporation の登録商標である．

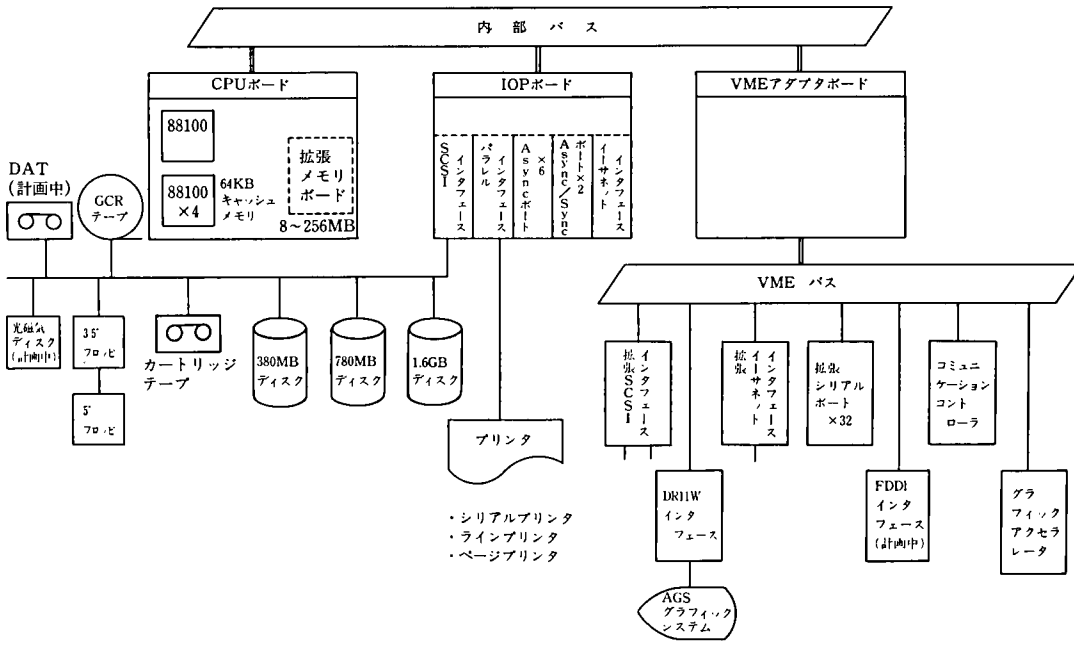



図1 ハードウェア体系

③ 150メガバイトカートリッジ・テープ
④ 3.5インチ/5インチフロッピ・ディスク装置
⑤ 光磁気ディスク装置（予定）
⑥ 4 mmDAT 装置（予定）
⑦ AGS 2500/3200 シリーズ・グラフィック・ディスプレイ
⑧ シリアル・プリンタ/ライン・プリンタ/ページ・プリンタ
⑨ 静電プロッタ/ペン・鉛筆プロッタ
⑩ ディジタイザ
⑪ 漢字ターミナル，X Window 支援ターミナル

### 1.4 3次元高速グラフィックス・アクセラレータ

3次元高速グラフィックス・アクセラレータはS 8400 シリーズの VME バスに実装される VME インタフェース・ボードであり，本アクセラレータを実装することによりS 8400 シリーズを GWS（グラフィックス・ワークステーション）としても使用することができる．

グラフィックス・アクセラレータは，現在日本ユニシスが提供しているグラフィックス・ディスプレイ AGS 2500 シリーズ/AGS 3200 シリーズの機能を1ボード化したものであり，その性能は400 K ベクトル/秒，30 K ポリゴン/秒である．また今後ミッドレンジ，ハイエンドクラスのグラフィックス・アクセラレータも予定されている．

## 2. S 8400 シリーズのソフトウェア概要

### 2.1 最新の UNIX オペレーティング・システムを採用

S 8400 シリーズのオペレーティング・システムは AT&T SYSTEM V リリース 3.2，バークレー版 4.3 BDS，および SUN OS の機能を統合した最新の SYSTEM V リリース 4.0（SVR 4）を採用している．日本語機能は AT&T の国際化機能である MNLS（マルチ・ナショナル・ランゲージ・サプリメント）を基本とし，ユーザインタフェースを日本ユニシスで拡張している．

また標準機能としてイーサネット LAN を構築するための TCP/IP (Transmission Control Protocol/Internet Protocol)，NFS* (Network File System) および ANSI 言語標準（X 3 J 11）準拠の C 言語を提供する．さらに S 8400 シリーズのオペレーティング・システムは，以下に示す各国際規格に準拠している．図2にソフトウェア体系を示す．

* NFS は SUN MICRO SYSTEM INC. の登録商標である．

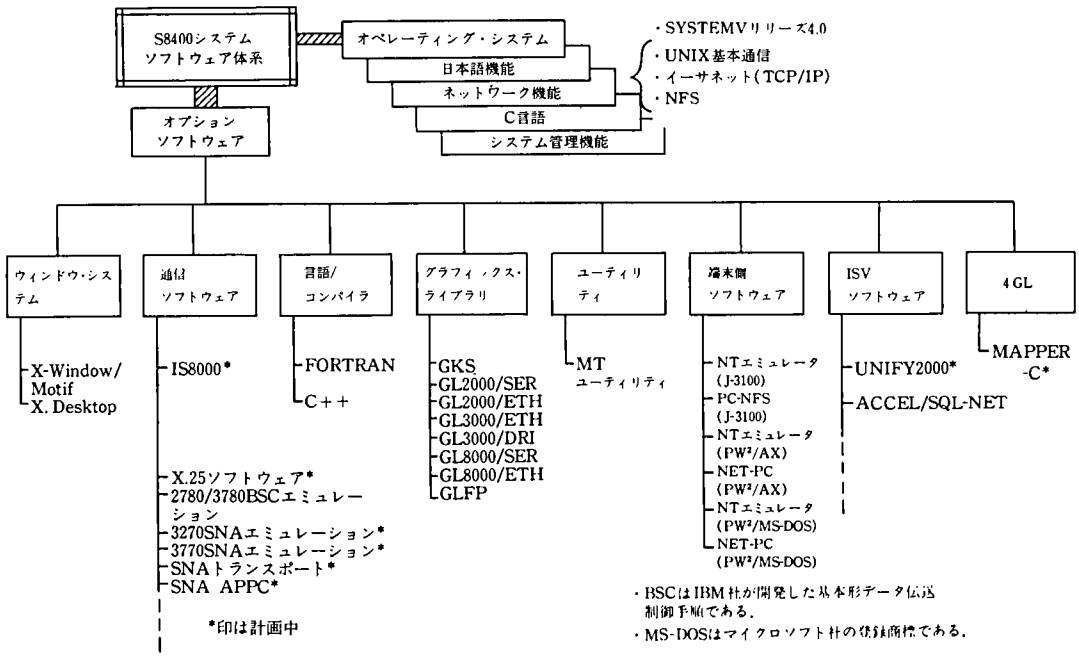



図 2　ソフトウェア体系

① AT&T SYSTEM Vインタフェース定義書ｱﾝ3版 (SVID 3)

UNIX SYSTEM Vへのインタフェースを定義するためにAT&Tが発行．すべてのUNIX SYSTEM Vインプリメンテーションの間で一貫したオペレーティング・システムとプログラム間のインタフェースを規定．

② X/open* 移植ガイドｱﾝ3版 (XPG 3)

X/openは欧州で設立され，マルチベンダ間の共通アプリケーションの開発環境の構築をめざし，ソフトウェア開発の効率化のための標準インタフェース環境のための勧告を行っている．

③ IEEE 1003.1-1988 POSIX規格

POSIX (Portable Operating System for Computer Environment) はUNIXのユーザ団体として設立され，UNIXのアプリケーション・プログラムの移植性を保証できる標準を制定している．このユーザ団体はIEEEに編入され，P 1003委員会としてPOSIX 1003.1勧告を行っている．

④ 88オープン・コンソーシアムのBCS/OCS

### 2.2 88オープン・コンソーシアムの定義に完全準拠

88オープン・コンソーシアムはモトローラ社のRISCプロセッサM 88000ファミリの利用を国際的に普及するために設立された機関である．

88オープン・コンソーシアムでは，M 88000ファミリを搭載しているUNIXワークステーション間でのバイナリ・レベルでの互換性を実現するための標準仕様BCS（バイナリ・コンパチビリティ・スタンダード），およびオブジェクトレベルでの互換性を実現するための標準仕様OCS（オブジェクト・コンパチビリティ・スタンダード）を定義している．S 8400シリーズでは，この88オープン・コンソーシアムのBCS/OCS定義に完全準拠しており，M 88000ファミリを採用したシステム上で開発され，また88オープン・コンソーシアムで認定されたサードパーティ・ソフトウェア（ISV開発ソフトウェア）を容易に稼働させる環境を提供する．

### 2.3 最適化コンパイラの提供

RISCプロセッサの能力を最大限に引き出すためには，各RISCプロセッサに適合した最適化コ

* X/OPENはX/OPEN社の登録商標である．

ンパイラが不可決である．最適化コンパイラは高水準命令の実行に必要な実行命令の数を減らし，システム性能を強化する機能を持つコンパイラである．S 8400 シリーズで提供する FORTRAN, C 等の言語コンパイラは MC 88100 RISC プロセッサ用の最適化機能をそなえている．

### 2.4 ウィンドウ環境

S 8400 シリーズでは X window/Motif, X. desktop からなるウィンドウ環境を提供する．Motif は X window リリース 11.4 に日本語対応を行った GUI（グラフィカル・ユーザ・インタフェース）であり，X window 上のプログラム開発を容易にしている．Motif の表示形態（アピアランス）の特徴は，ウィンドウ，アイコン等に立体的な陰影を付けた表示が可能であり，ウィンドウ内のレイアウトおよび動作内容（ビヘィビア）は MS OS/2 *のプレゼンテーション・マネージャ**に似ている．

X. desktop は Motif の上で動作するアプリケーションの一つであり，UNIX の操作を誰にでも簡単に行えるようにわかりやすいアイコンで表示し，マウスでの操作ができるようにしたものである．

### 2.5 各種通信ソフトウェア

S 8400 シリーズでは，標準提供されている TCP/IP, NFS とともに，ユニシス/他社システムとの接続を可能にする通信ソフトウェアを提供する．これらの通信ソフトウェアにより，ホストシステムとの親和性を持たせた水平型分散，垂直型分散ネットワークの構築を可能にしている．

以下に各種通信ソフトウェア（予定商品も含む）を示す．

① BSC 2780/3780 エミュレーション
IBM 2780/3780 端末エミュレーションを行う．

② SNA 3270 エミュレーション（予定）
SNA***手順で接続し，IBM 3270 端末のエミュレーションを行う．

③ SNA RJE (3770) エミュレーション（予定）
SNA 手順で接続し，IBM 3770 RJE（リモート・ジョブ・エントリ）端末のエミュレーションを行う．

④ SNA APPC LU 6.2（予定）
IBM APPC LU 6.2 プロトコルを使用している異機種間どうしを SNA 手順で接続し，分散処理，トランザクション処理を行う．

⑤ IS 8000（予定）
シリーズ 2200・1100 ホストシステムと UDLC, X.25, イーサネットを介して接続し，分散システム環境を提供する．

⑥ X.25 ソフトウェア（予定）
X.25 PDN（パケット・パブリック・データ・ネットワーク）広域網との接続を行う．

### 2.6 グラフィックス・ライブラリ，4GL，データベースソフトウェア

S 8400 シリーズでは，グラフィックス・ディスプレイ AGS 2500/AGS 3200 シリーズ，静電プロッタ AGS 8000 シリーズ，鉛筆・ペンプロッタを使用するためのグラフィックス・ライブラリを提供する．

また，グラフィックス・アクセラレータ用のソフトウェアとして，AGS 2500/AGS 3200 エミュレーション・ソフトウェアおよび 3 次元図形処理言語 PHIGS, PHIGS API を提供する．

さらに 4GL として Mapper-C, データベース・ソフトウェアとして ISV 扱いの UNIFY2000**** 等を提供する予定である．

### 参考文献

[1] 日経エレクトロニクスブック「88000 ファミリの設計思想」, pp. 111～124, 1989, 9.
[2] 日経コンピュータ別冊「充実するビジネス分野での UNIX, アプリケーション環境」pp. 113～130, 1990 7 11.
[3] 日経コンピュータ別冊「パソコンからメインフレームまでの一貫性を持つ OSF/Motif」pp. 91～98, 1989 7 28.

---

* MS OS/2 は米マイクロソフト社の登録商標である．
** プレゼンテーション・マネージャは MS OS/2 上で動作する複数のウィンドウ操作ができる GUI，米マイクロソフト社の登録商標である．
*** SNA は IBM 社のネットワーク・アーキテクチャである．
**** UNIFY 2000 は UNIFY 社の登録商標である．

# 新世代パーソナル・ワークステーション PW² LT386/20C

日本ユニシスは，高機能ワークステーションPW²ファミリのラップトップタイプとして10インチTFT(Thin Film Transistor；薄膜トランジスタ）カラー液晶ディスプレイを搭載したPW² LT 386/20 Cの販売を開始した．

LT 386 / 20 Cは，Intel社の32ビットCPU i 80386*(クロック周波数20 MHz，ノーウェイト）を採用．標準2メガバイト最大10メガバイトのメモリ，平均アクセスタイム17 msの100メガバイトのハードディスクを内蔵するという高性能デスクトップマシンに相当する機能/パフォーマンスを実現している（表1仕様概要，図1システム構成図）．

## 1. PW² LT386/20Cの特徴

PW² LT 386/20 Cは，以下のような特徴を有している．

1) TFTカラー液晶ディスプレイを採用……解像度640×480ドット，64色中16色同時表示が可能なTFTカラー液晶ディスプレイを採用．従来CRTでしか表現できなかった美しいカラー表示をラップトップワークステーションで実現．

2) AX仕様に準拠……PW² LT386/20CはAX仕様に準拠しているため，一太郎**やLotus 1-2-3**といった市販の流通ソフトウェアが使用可能である他，IBM PC/AT***用の数万種類のソフトウェアが使用可能．

3) デスクトップに相当する性能……PW²

表1　PW² LT386/20C仕様概要

| | | |
|---|---|---|
| CPU | | i80386(20MHz)，ノーウェイト |
| RAM | | 2MB（標準）～10MB（最大）<br>2MBずつの増設が可能<br>LIM/EMS　4.0対応 |
| 表示 | 表示パネル | 10インチTFTカラー液晶 |
| | 解像度 | テキスト　：漢字40桁×25行<br>ANK 80桁×25行<br>グラフィック　：640×480ドット |
| | 表示色 | 64色中16色同時表示 |
| キーボード | | 90キー（AX小型キーボード仕様準拠） |
| FDD | | 3.5インチ（1.44MB/720KB）×1 |
| HDD | | 3.5インチ（100MB）×1 |
| I/F | | プリンタ×1<br>RS232C×2<br>増設メモリ×4<br>内部拡張(PW² NOTE/286E用I/Fボードと互換)<br>拡張スロット×1 |
| OS | | MS-DOS Ver 3.21（標準）<br>MS OS/2 Ver 1.21（オプション） |
| 電源 | | AC100V（50/60Hz） |
| 消費電力 | | 100W |
| 外形寸法 | | 318(W)×399(D)×94(H)mm |
| 重量 | | 6.9Kg |

MS-DOS，MS OS/2は米マイクロソフト社の登録商標．

* i 80386は米インテル社の登録商標である．
** 一太郎は(株)ジャストシステム，Lotus 1-2-3は米ロータス・ディベロップメント社の登録商標である．
*** PC/ATはIBM社の登録商標である．

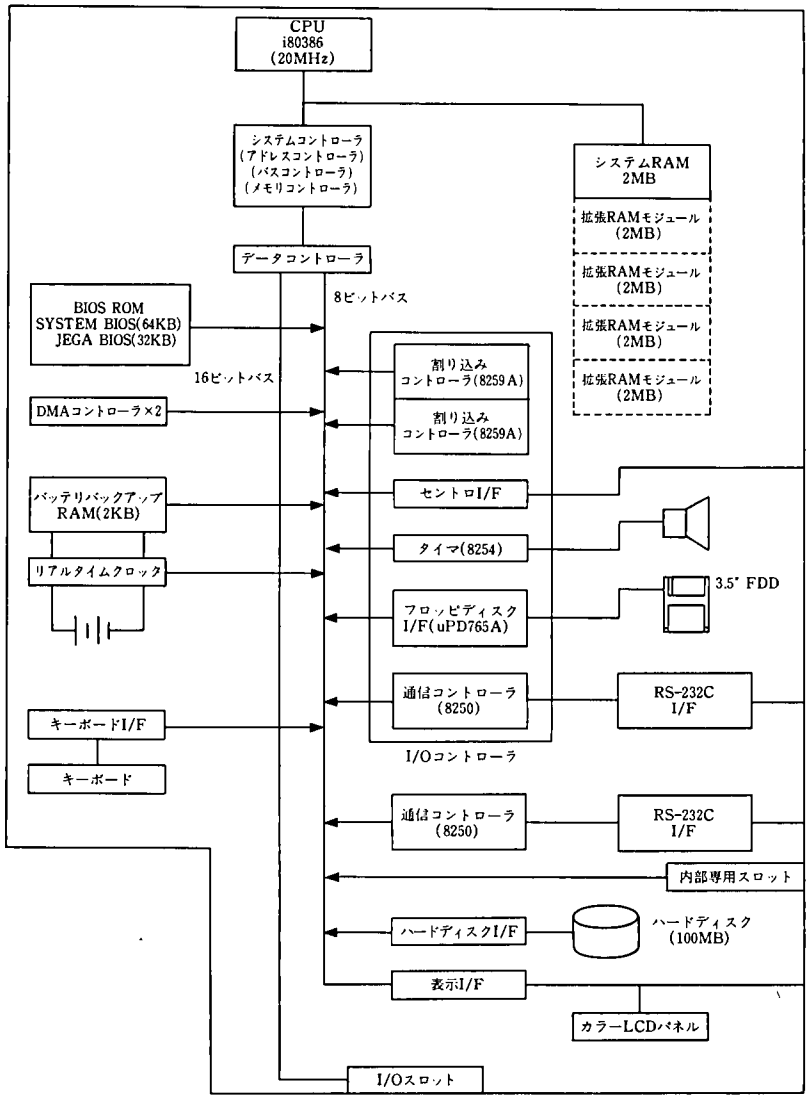



図 1　PW² LT386/20C システム構成図

LT 386/20 C は，CPU に 32 ビットの i 80386 (20 MHz，ノーウェイト) を採用，標準で 2 メガバイトのメモリを搭載しており，2 メガバイト増設メモリにより最大 10 メガバイトまで拡張が可能，高速大容量の 100 メガバイトのハードディスク（平均アクセスタイム 17 ms) を内蔵，等デスクトップに相当する機能/性能を持っている．

4)　MS-DOS，MS OS/2 の 2 種類の OS を提供……PW² LT 386/20 C は，オペレーティングシステムとして MS-DOS* を標準で提供．オプションとしてシングルユーザ，マルチタスク OS である MS OS/2* も搭載可能．

MS OS/2 を搭載することにより，日本ユニシスの新世代統合 OA システムである Excellent Office System (EXOS) のワークステーションとしても使用することができる．

* MS-DOS，MS OS/2 は米マイクロソフト社の登録商標である．

## 2. PW² LT386/20C が採用している技術

### 2.1 TFT 液晶カラーディスプレイ

液晶ディスプレイは，1963 年 RCA 社で液晶に電気的な刺激を与えると，光の通し方が変わることを発見．その 5 年後に同社で，この性質を応用した表示装置を作ったのが始まりである．

液晶ディスプレイは，その駆動方式によって，単純マトリックス方式と，アクティブマトリックス方式に分けられる．LT 386/20 C が採用している TFT 液晶ディスプレイは，アクティブマトリックス方式に属している．液晶ディスプレイに使用されている液晶には，ネマティックという材料が用いられている．このネマティックは，捻った方が応答が早くなる性質を持つ．アクティブマトリックス方式とは，この捻ったネマティックを各画素ごとに薄膜トランジスタを付け応答速度をより速くしたものである．

PW² LT386/20C

液晶ディスプレイは，液晶を，配向膜，カラーフィルタ，透明導電膜，ガラス，偏光板で挟んだ構造となっている．また液晶は自発光でないため，背後からバックライトを当てる必要がある．

### 2.2 AX 仕様

AX 仕様の基礎になる IBM PC/AT は 1981 年に発売されたパーソナルコンピュータである．PC/AT は徹底したオープンアーキテクチャ政策がとられ，全回路図や BIOS プログラムの公開等を積極的に行った．このため，世界的な規模で数多くのメーカから互換機が発売されるようになり，PC/AT とその互換機はアメリカやヨーロッパ等の英語圏だけでなく，東南アジア，中国，韓国に至るまで世界各国で 1,000 万台以上の規模で生産されている．

AX 仕様は，この世界の標準仕様ともいわれる PC/AT のアーキテクチャをベースに日本語仕様を加えたものである．したがって，PC/AT 用の 5 万とも 6 万ともいわれる莫大なソフトウェア資産を有効に利用できることを前提にしている．AX のキーボードは，PC/AT のキーボードに日本語処理を行うためのキーを加えたもので，統一した仕様に基づいており PC/AT のキーボードと上位互換を保っている．したがって，AX 仕様に基づい

表2　PW² LT386/20Cで使用可能なMMLソフトウェア

| ソフトウェア名称 | 機能概要 |
|---|---|
| MT983Kエミュレータ | Aシリーズの端末として使用するための端末エミュレータ |
| U-TRANS/A | Aシリーズとの間でデータの送受を行う |
| A-MAPPER/GR-DKT | A-MAPPER（Aシリーズ用MAPPER）を使用するためのソフトウェア |
| LBP日本語プリンタファシリティ3 | Aシリーズに接続されたLBPの書式を簡単に作成する． |
| データ転送システム | Aシリーズとの間でデータの送受を行うソフトウェア・市販流通ソフトウェアで加工できるような形式で転送できる． |
| UTSエミュレータ | シリーズ2200・1100の端末として使用するための端末エミュレータ |
| U-TRANS/1100 | シリーズ2200・1100との間でデータの送受を行う |
| NTエミュレータ | U6000等UNIX*シリーズの端末として使用するための端末エミュレータ |

*UNIXオペレーティングシステムは，UNIX System Laboratories,Inc. が開発し，ライセンスしている．

て作られたマシンではメーカによって操作が異なることはなく，PC/AT とも同じ操作でアプリケーションソフトを利用することができる．

## 3. ワークステーションとしての PW² LT386/20C

### 3.1 ホストコンピュータとの親和性

PW² LT 386/20 C は AX 仕様に基づいて作られているため，PW²ファミリの AX 用に作られたホストコンピュータとの接続用の端末エミュレータやファイル転送プログラム等，マイクロメインフレームリンク (MML) ソフトウェアの使用が可能である（表 2）．

### 3.2 EXOS ワークステーション

PW² LT 386/20 C は，オプションの MS OS/2 を搭載することにより統合オフィスシステム EXOS のワークステーションとしても使用することが可能である．

EXOS は 1989 年新しい統合オフィスシステムとして発表された．ワープロ，表計算等，各種 OA ツールを提供すると共に，ホストコンピュータ/サーバ/PC 間でのデータ連携を実現，OA ツール群の操作性を統一させ，オブジェクト指向の操作環境を提供したものである．

ISSN 0914-9996

技　報
UNISYS TECHNOLOGY REVIEW
Vol. 11　No. 3　(No. 31)

| | | |
|---|---|---|
| 発 | 行 | 日　平成3年11月30日 |
| 編 | 集 | 人　柳 生 孝 昭 |
| 発 | 行 | 人　富 田 和 夫 |
| 発 | 行 | 所　日本ユニシス株式会社<br>東京都港区赤坂2-17-51 〒107<br>TEL(03)3585-4111（大代表） |
| 印 | 刷 | 所　三美印刷株式会社 |