TECHNOLOGY REVIEW

UNISYS 技報

通巻 18

1988年8月発刊 Vol. 8 No. 2

日本ユニシス

ISSN 0914-9996

MAPPERの第一人者であるW.R.Grayは，**システム・インテグレーションと第四世代環境**の中で，4 GLに至るハードウェア／ソフトウェアの世代展開およびMAPPER出現の背景を概観し，エンド・ユーザ主体の情報システムを作り上げるツールとしての評価と実績を述べている．さらに，アプリケーション，オペレーティング・システム，ハードウェア・アーキテクチャを一元化した情報環境の必要性を説き，ユーザ・インタフェース，標準化，およびメインフレームとEDP部門の役割の再認識を一元化に必須の条件としてとらえている．最後に4 GLが対応するテーマとして，情報の分散，オフィス環境，データ処理，意思決定支援を検討している．

日本語文書処理における入力方式は，連文節変換，文章一括変換等ますます便利になってきた．最近では複合語を一度に変換する方式も現われたが，正しい漢字への変換精度は今一つである．複合名詞の末尾の語は，それ以前の語(群)の接尾語として機能していることが多く，両者の間には意味的なつながりがある．この関係を調べることにより，複合名詞の漢字変換の精度向上が期待できる．稲永紘之と新谷隆之の，**意味的なつながりを考慮した複合名詞のかな漢字変換システムの構築**では，複合名詞の接尾語ごとに，それに係り得る語を収集・編集した接尾語辞書を用いることによって，意味のあるかな漢字変換を行えることが検証できたことを報告している．

東京電力(株)では，コンビニエンス・ストアに業務委託し，夜間・休祭日でも電気料金の収納が行えるように，任意のデータをJANコードに変換し，大量に印書することが必要となった．浅井伸之と長谷明拓は，**東京電力(株)における高速日本語印書装置によるJANコードの印書**の中で，各倍率のJANコードが24ドット(10 cpi)，20ドット(12 cpi)，16ドット(15 cpi)の組合わせで表現でき，そのうち24ドット(10 cpi)のみで表現できるJANコード(倍率1.1倍)の読取りテストを行った結果，実用に耐える品質が得られたことを報告している．

知識工学ワークステーションKS-301は，Common Lispが稼働するLisp専用マシンである．このKS-301で日本語の使用を可能にすべく日本語Common Lisp環境(NCL)を開発した．大田一久は，**知識工学ワークステーションKS-301の日本語Common Lisp環境（NCL）**の中で，NCLのCommon Lispへの実現について述べている．NCLではCommon Lispで日本語文字が使用できるだけでなく，KS-301のプログラミング環境全体を通して日本語文字の使用を可能にした．

TSX-PASEは，統合的ソフトウェア開発支援システムTSX 1100を構成するソフトウェア・ツールの一つである．TSX-PASEは，ファイル処理を定型化した言語PASE 1100による仕様記述からCOBOLソース・コードを生成する仕組みに，ソフトウェア部品の再利用の技法と，情報資源を管理するデータ・ディクショナリを活用する技法を応用したものである．長谷川邦夫は**プログラム自動生成システム——TSX-PASE**の中で，機能と部品定義言語について紹介している．

日本ユニシスの住宅CADシステムは，一般住宅の設計・製図・積算および躯体／部材展開を行う一貫システムである．住宅CADシステムの一部である部材展開は，家モデルをもとに出荷する部材を決定する機能である．本間一郎は，**住宅CADシステムにおける部材決定のためのデータ構造**の中で，この部材決定の際に影響のある各種条件と，それによって決まる部材との関係を定義したデータ構造について紹介している．

新しいコミュニケーション・メディアとしてパソコン通信が注目を集めている．日本ユニシスでは，シリーズ1100／2200をホストとしたパソコン通信ネットワーク構築機能を提供するJUST-PC方式対応パソコン通信パッケージPCHOSTを開発した．このファイル転送機能のプロトコルとしてOSI FTAMを適用した．広田雅史は，**パソコン通信におけるFTAMの実装**の中で，パッケージ概要，ファイル転送サービス機能，効率および今後の動向について述べている．

システム・インテグレーションと第四世代環境

Integration, the Key to Success in a Fourth Generation Environment

W. R. Gray

要　約　第四世代言語(4 GL)に至るハードウェア，ソフトウェアの世代展開とDBMSの概念に触れ，MAPPER出現の背景を概観する．エンド・ユーザが主体となり情報処理システムを作り上げるツールとしての評価と実績を述べる．さらにアプリケーション，オペレーティング・システム，ハードウェア・アーキテクチャを一元化した情報環境の必要性を説く．ユーザ・インタフェース，標準化，およびメインフレームとEDP部門の役割の再認識を一元化に必須の条件としてとらえる．

最後に4 GLが対応するテーマとして，情報の分散，オフィス環境，データ処理，意思決定支援を検討する．

Abstract　Overviews histrical walks toward the generation concept of computer hardware and software to get the fourth generation languages (4GL) started, which forms the framework of MAPPER system. Then analyses and evaluates the drive to end-users oriented computing. The author stresses that integration of applications, operating systems, and hardware/software architecture is essential to making 4GL successful business tool. Other key area includes user-interface, standardization, and restructuring traditional role of mainframe as well as EDP related people.

Discusses such topics as information distribution, office environment, data processing, and support for decision making, which the 4GL has to go through.

1. はじめに

およそ18,000本余りの真空管を抱えた電子機器，エニアック（Eniac）が登場した時代（1943～1946）以来，電子計算機の領域では数多くの重要な出来事があった．

われわれの関心事は，もっぱらソフトウェアおよび関連するプログラミング技法にあるが，ごく簡単にハードウェアの動向にも触れておきたい．ハードウェア，すなわちデータ処理分野で使用されているコンピュータのCPUは，今日，高度に洗練された言語ツールによって駆動されているからだ．

第一世代のハードウェアは1945年に誕生し，1959年頃まで使用された．真空管の固まりでできた怪獣とでも表現できる代物で，三極管または五極管が論理装置のすべてであった時代である．トランジスタの発明をきっかけとして第二世代のコンピュータが生まれ，1964年頃までの時代を担った．ハードウェアは1964年に集積回路(IC)が実用化されるに至り，採用技術は急速に発展し，第三世代のコンピュータを実現させた．しかし，このコンピュータも1980年代に大規模集積回路(LSI)や超LSI(VLSI)が実用技術の域に達すると終りを告げる．種々議論の分かれるところであるが，専門家の多くは1980年代のハードウェアを第四世代コンピュータと呼称している．

科学者と電子技術者は現代のハードウェア技術をなお一層，進化させようと試みて

いる一方で(光の速度を超えられないという限界はあるが)，ソフトウェアに関心を寄せる技術者集団は言語とユーザ・インタフェースの最新技術を急速に発展させている．

ソフトウェアもハードウェアも同様に進化を続けてきた．コンピュータはオンとオフの二つの状態を動作の要素としているため，最も初期段階のコンピュータ言語は二進数であった．1がオンの状態を，また0がオフの状態を表すのに適していたからである．したがって，機械語と言えば，命令とデータを二進数で記述したものを意味し，そのままハードウェアが解析可能であった．

現代のわれわれが考えても，オンとオフに点滅するメンテナンス・パネルをながめても，面白いはずはない．当時の科学者や技術者がアセンブラを考案するまでに長くはかからなかった理由もここにある．同時に実行するプログラムは一つという古典的フォン・ノイマン型コンピュータ理論を採用して，命令コードは記号で記述し，メモリ・アドレスに記号名が付けられるようになる．アドレスも記号名で参照して算出でき，プログラマは数定数や文字定数が使用できるようになった時代だ．アセンブラは現代でも利用されている．

ハードウェア装置を直接利用したり，ハイレベル言語では利用できないアーキテクチャの中枢機能を効率よく使用する場合に必要なことがある．アセンブラを使用するか否かは，システムのパフォーマンスとシステムの稼働効率，あるいはそのいずれかの観点で判断するのが普通である．

アセンブラは確かに大きな前進の手段ではあったが，何と言ってもFORTRANとCOBOLの出現がコンピュータ産業にとっては大いなる福音となった．技術者は数式で，また事務所では一般の事務所用語でコンピュータに話しかけることができるようになったからだ．ここに第三世代言語が誕生したわけだ．

2. コンピュータ言語，第四世代へ

第三世代の言語が出現するにおよんで，アプリケーション・ソフトウェアを開発する人間はようやく光明を得た心地がした．ハードウェア・アーキテクチャを完璧に理解しなくてもよくなったし，機械語の命令セットやさまざまなレジスタ・セットを覚える重荷からようやく開放されたのであった．ハードウェアへの依存性も弱められて，プログラムもある程度まで可搬性が持てるようにまでなった（図1）．

いずれもハードウェア・アーキテクチャの進歩をはるかに上回る，ソフトウェア上の発明であった．なぜなら，これらのソフトウェアがなければ，現代のハイテク時代も到来しなかったであろうし，高度情報化社会などは考えられもしなかったのだから．

最初の成果は科学技術分野の用途に開発されたFORTRANで，最初に広く実用に

機械語	アセンブラ	COBOL
741300435100	LMJ X11, MORE	PERFORM MORE

図1 コンピュータ言語の三世代

Fig. 1 Three generations of language

供された高水準言語となった．FORTRAN は 1950 年代の中頃，IBM の J. バッカスが中心となって開発した言語である．

科学技術アプリケーションが一歩先んじた形で初期のコンピュータをリードしたが，事務用途がまったく顧みられなかったわけではない．事務分野向けの最初の言語として登場したのはフローマチック (Flow-matic) と呼ばれるもので，やはり 1950 年代の中頃，レミントン・ユニバック*の G. ホッパー女史が中心となって開発した言語であった．この Flow-matic は英語に類似した体系を持つ最初のコンピュータ言語で，その後 COBOL の開発に重要な役割を果たすことになる．

ソフトウェアの開発努力は言語の分野だけでなく，データ処理の面でも続けられた．とくに，今日，データベース・マネジメント・システムと呼ばれる技術分野で，三つの大きな手法が出現した．その一つは，ゼネラル・エレクトリック社の C. バックマンが 1964 年頃に提唱した，データ保存のネットワーク・スキーマという概念である．第二はデータの階層構造または木構造と呼ばれる考え方で，1969 年，IBM が情報管理システムの基本概念として発表したものである．第三は 1970 年，IBM の E. F. コッドが発表したリレーショナル・データベースの概念である．このリレーショナルという考え方は，情報の重複がないデータの表を基本とするものであった．

さて，いわゆるアプリケーション，たとえば給与システムを作ろうとするとプログラマやアナリストなど，専門家が必要となる．そこはデータ処理部門の独壇場だ．彼等はプログラムを書き，ソフトウェアをデバッグするばかりではない．出来上がったアプリケーションを改造したり，プログラムの保守要員を提供するなど，彼等の仕事は際限なく続く．

ある調査によれば，ソフトウェアを提供するプロジェクトの費用の 67 パーセントは保守に費やされ，実際の開発にはわずか 33 パーセントしか充当されていないといわれている．実際にソフトウェアのライフサイクルは，その 67 パーセントが保守に占められるとすれば，この領域にもっと目を向けるべきであり，保守用ツールの改善や保守自体を不要にする試みがなされる必要があろう．保守問題の最良の解決策は，保守に耐えうるソフトウェアを作ることなのは当然だ．新しいソフトウェアを開発するときに，過去に起きた誤りを繰り返してはならない．常に“保守を念頭に置く”ことである[3]．

ところでウェインバーグの 80/20 の法則についてはどうか．この法則によれば，保守工程の 80 パーセントが，出来上がったコードのわずか 20 パーセントに費やされるという．この規則は偶然にも，一般の産業界にもあてはまることが知られている．コンピュータ業界だけに通用する事柄ではないのだ．事実，この法則は生産工程における品質管理理論が基になっている．(ジュラン著，Quality Control Handbook 参照[3]) ジュランはこの所説を，「富の 80 パーセントは国民の 20 パーセントが所有する」とする経済原則から導き出している．

以上のような事柄が進んでいる間，データ処理分野ではもう一つの重要な現象が起こりつつあった．それはリアルタイム，マルチ・ユーザ環境におけるデータ処理の進展である．1968 年，ユニスコープ 300 CRT という，入力装置としては十分な実用性を

* 現在の米国 UNISYS 社

備えた機器の出現に伴い，スペリー・ユニバック社*では製造部門向けのリアルタイム，マルチ・ユーザ環境のシステムを設計，開発し始めた．このシステムはデータベースの検索と更新を多用するものであった．インバーティド・ファイル構造やサーチ・テーブル，リンク付き拡張レコードなどを採用した概念モデルによる実験では，階層構造を持たないファイル・アーキテクチャが最も効率がよいことがわかっていた．

このリアルタイム，マルチ・ユーザ環境は後にCRT-RPSレポート処理システムとして知られるようになり，数百のエンドユーザが定常的にネットワーキングを扱えるスペリー・ユニバックの社内ツールとして広く利用されるようになった．同社内におけるCRT-RPSの人気は高く，また全社的な規模のサービスを求める声も高くなり，種々の機能拡張と合わせて1972年に至り，このシステムはスペリー社*のシリーズ1100プロダクト・ライン上で稼働するシステムとして生まれ変わった．こうしてMAPPER**システムが誕生したのだ．

1976年偶然の結果として，ユニバック*の社外では初のMAPPERシステム購入ユーザが出現した．現在，MAPPERシステムはPC（パーソナル・コンピュータ）からメインフレームまで広く普及し，2000メインフレーム・ユーザと4000を越えるパソコン・ユーザを擁するに至っている．

MAPPERは事務処理用ソフトウェアとしての名声を確立し，ユーザに高い生産性を提供している．また業務処理問題への対処策として，また意思決定支援ツールとして業種業態に問わず広く活用されている．MAPPERは情報処理産業における革新的プロダクトになったと言えるだろう．

J. マーチンは高生産性言語に関するレポート（1986）で，次のように述べている：

「MAPPERシステムの価値は，エンド・ユーザが自らの業務経験を生かしてデータを処理し，情報処理の専門家の力を借りず，自分でレポートを作成できるところにある．情報処理の非専門家であるエンド・ユーザがMAPPERシステムを利用することで，情報処理の能力は目を見張るほど向上した．従来型のデータ処理部門はエンド・ユーザが情報をアクセス処理する能力など，ほとんどないものと過小評価していた．しかしMAPPERなどの適切なツールを与えれば，驚くほど複雑な情報システムでもエンド・ユーザが作り上げることができるのだ．このようなツールをエンド・ユーザが利用できれば，業務上必要な解決策を直接，得ることができる[2]．」

MAPPERは今や，一般企業のほか官公庁，教育機関などに広く普及し，コンピュータ活用の主要なツールとして使われている．その理由はユーザが使いやすいことにある．またデータ処理機能のスループットを劇的に向上させ，所要コストを大幅に低減させているのも確かである[12]．

MAPPERシステムは，コンピュータのあらゆる製品を通して利用可能になっている．ユーザはユニシスの大型コンピュータの時間を買い，MAPPERが利用できる．機器を購入する必要がないので，もっとも安全確実に利用できる形態である***．また

*現在の米国UNISYS社

**UNISYS社の提供するエンド・ユーザ向け第四世代言語

***米国ではMEISC（MAPPER Exective Information Support Center）と呼ばれるVANサービスが普及しており，思い立ったらすぐに試行することもできる．

MAPPERシステムは，スペリー社あるいはIBM社のパソコンにも導入して利用できる．さらにミニコンからメインフレームまで幅広い機種でMAPPERが利用できるので，導入後にシステム規模が拡大しても十分に対応可能である．またMAPPERはフランチャイズ・システム*としても利用できる．MAPPERシステムが，いかに強力で機能が優れているかがわかるであろう．現在市場にある第四世代言語でMAPPERを越える製品は，いまだ登場していない．MAPPERのフランチャイズは，業界初の試みである[12]．

3. インテグレーション

ソフトウェア・アプリケーション，オペレーティング・システム，およびハードウェア・アーキテクチャのインテグレーションは，コンピュータ業界で最も注目されている領域の一つと言えよう．このインテグレーションへ至る大通りの一つがオープン・システム・アーキテクチャだと言われている．システムのインテグレーションを実現するには，大方のコンピュータ・メーカが継続的に注目し行動して，データ処理の諸要素を一つの情報環境にまとめ上げることが必要である．

システム・インテグレーションのモデルとして，アムステルダム市（オランダ）の旧市街の例が参考になるだろう．現在の同市は，整然とした水路のネットワークと住宅地，倉庫群が整い，ビジネスと生産の場までもが公共輸送システムに沿ってうまく統合されている．これを見ると都市計画も，ビジネスと生活空間との間に適度な距離と交差点を確保した人間レベルのインテグレーションであることがよくわかる．

情報システムに目を転ずるとき，われわれは人間レベルのインテグレーションと同様の配慮と工夫でその設計にあたって欲しいものだと考える．しかしコンピュータ・システムの環境は往々にして新興都市のごとく，統一した計画もなく，高層ビルが次々と錯綜し自動車が勝手に走り回るように拡大していくことが多い．幸いEDP部門にはこれからでも物事をまとめ上げる好機が多くある．ユーザは業務上求められるさまざまな構成要素から成る情報環境を案出できるのだ．これは新しい動向を見つめ，統合システムを念頭に置いて対応する好機であることを意味する．将来のあらゆる製品を貫く基本計画を整えるチャンスなのである．

ユーザが情報処理環境内で直面しなくてはならない数々の潮流のなかで，システムおよびシステムによる解決策を提供するソフトウェア・メーカの立場から，次の3点がとくに重要だと考える．

1) パソコンの普及……パソコンが企業内の個人および隣接するオフィス環境，ならびに部門環境に与える影響は，情報の品質や価値にも大きく影響する．
2) 通信と情報処理の結合がますます強くなっている……アプリケーション・システム設計の新しい手法や考え方が，通信と処理の結合により実現していくことが多い．
3) メインフレームの存在価値が変化しているという現実的な認識……いまだ企業環境の重要部分を占めているメインフレームであるが，従来とはまったく異なる

*UNISYS社のシステムを借り，その上に付加価値を持ったMAPPERアプリケーションを開発，第三者に提供する形態などのニュービジネスを指す．

役割と価値を追求する傾向が出始めている．

以上の各項はEDP部門やコンピュータ・メーカがソフトウェア製品を設計し開発する上に大きな影響力を持つ．ユーザの真のニーズを的確に把握するため，繰り返しチャレンジしなくてはならない事項なのである．ソフトウェアの設計によりわれわれは情報環境の新しい方向性を見い出し，産業界の期待を実現しなくてはならない．

パソコンと通信機能の進歩により，ここ数年の間にエンド・ユーザの様相は大きく変貌した．いま，データ処理部門の管理者が注目すべきは彼等の動向である．エンド・ユーザは自ら考え行動するためのツールとしてパソコンを利用しており，データ処理専門の部門とは切り離されていることが多くなった．

データ処理部門はこの新個人主義ともいうべき現実を直視し，企業内生産性向上の新たな挑戦の課題と考えるべきである．同時に個人レベルと企業レベルの情報環境の両方を調整する手段を提供しなくてはならないだろう．パソコンの個人ユーザは一層，能力を高める努力が必要であろう．また適切な業務管理に基づいて安定的かつ秩序あるパソコン・システムを確立し，提供する責任を担っている．

とくに注目すべき領域はエンド・ユーザ，オフィス・システム，および部門システム相互間の接点である．個人とビジネスマンの個性が交差するのは，まさにこの接点，つまりユーザとシステムとのインタフェースなのである．パソコン・ソフトウェアは主として個人使用を前提としているが，部門システム向けあるいは一部のオフィス向けソフトウェアは業務処理用途がねらいである．ソフトウェアを提供する側は，そこに生まれる新しい環境にいるエンド・ユーザのエネルギーを活用し管理して，ビジネス情報を十分，行き届かせることができる．エンド・ユーザが自らのアプリケーションを作成し操作できるソフトウェア・ツールを手にすると，どのようなことになるかを知るのにもMAPPERの事例が大いに役立つ．

これまで続いてきた通信とコンピュータとのインテグレーション・プロセスは，最近に至り新たな広がりを見せている．端末とホスト・コンピュータという従来型のパターンは，しばらく前からコンピュータ同志が直接，交信しあう分散型データ処理形態に置き変わりつつある．しかしこの利用形態が完全に確立するまでには，複雑なシステムを相互接続する共通の，プロトコル，および操作ツール，アクセスと制御手段が必要なところで適宜，実現されなくてはならない．このような一般的方法論と手段を，われわれは少しずつ獲得しつつある．

これを標準と呼んでもよい．この標準により，相互接続するすべてのコンピュータはお互いを必要とする時に適切に連携しながら機能することが可能となるのだ．

コンピュータ・メーカは，変わりゆくユーザ環境でメインフレームが果たすべき役割の役どころに大きな関心を持っている．メインフレーム，すなわち大型機以上のコンピュータは，やがて消え去るだろうと言われたのはそう昔のことではない．現実にはほとんどの企業が，分散処理への動きの中で情報処理の中枢をになうコンピュータ・システムを確保している．

つまりメインフレームは移り変わる世の中でデータの貯蔵庫として，また情報の自動制御の最終的な資源としての価値が認められ，分散システムの中でも大きな役割を果たしつつあるのだ．メインフレームのニーズは進化し，また強くなっていることが

わかる．しかもあくまで柔軟な方法でなくてはならず，またパソコンやミニコン，スーパーミニコンなど，それ自体の役割をメインフレーム環境で調整しなくてはならない．システムの構成要素を十分に考慮した前進でなくてはならないのである．

ソフトウェア・プロダクトの開発者はエンド・ユーザと通信機能，およびメインフレームの動向に大きな注意を払うことが必要である．これら機能領域が情報処理環境でカバーする範囲は広い．ソフトウェア・プロダクトでこれら諸環境をインテグレーションし，情報処理環境にあるシステム・プロダクトをインテグレーションすることが，最も広義の目標でなくてはならない．仮に完璧な程度までにインテグレーションを果たしたソフトウェア・パッケージがユーザの手にあれば，かなりの程度まで複雑なシステムの利用環境下においてもアプリケーション環境の真の統合化を実現することができるだろう．

ソフトウェアのインテグレーションを考えるときは，次の四つの機能領域に注目すべきである．

1) 情報の分散—分散システム・サービスと言われるもの
2) オフィス環境—オフィスの情報システム
3) データ処理—業務処理を中心とした領域
4) 意思決定支援—業務上の意思決定を支援するツール

以上の各機能の内部，および各機能間のインテグレーションの両方がわれわれの関心事である．インテグレーションと言うとき，われわれの成果物であるソフトウェアと，そのソフトウェアのアプリケーション・ユーザが，その他一般の通信，各種フォーマット，プロトコル，機能をすべて有機的に結び付けて利用して欲しいものだと願わずにはおられない．

4. インテグレーション対象の四分野

システムのインテグレーションは，データ処理分野の四つの主要分野を一つにまとめたときに初めて実現可能となる．またオープン・システム・アーキテクチャ，オープン・システム・インタフェースの考え方，および業界標準が採用されてたときに初めて達成できる．

そこで本稿ではまず，情報の分散問題とオフィス環境を検討してみたい．いずれも簡単に定義でき，また業界内での認識にも大きな相違のないテーマであるからだ．

4.1 情報の分散

さて，いわゆる分散システム・サービスにおける，システム全体を通した情報分散という一般的なサービス分野について考えてみよう．データ処理において分散処理ソフトウェアがこれまで利用されてきていることは衆知の通りである．相互に接続されたシステム間でファイル，プログラム，および処理の制御を受け渡しする手段を提供するのが，ここでいう分散処理ソフトウェアである．このソフトウェアは業界標準を基にした情報交換と情報操作を提供する分散情報サービスを補完してきたのである．

ソフトウェア・プロダクトは集中管理した情報ライブラリを持ち，IBM の DISOSS サービスのような形態で作られていなくてはならない．これによりホスト・コンピュータは情報の集中管理倉庫として機能し，文書交換アーキテクチャ(DIA)に準拠する

文書データの授受ポイントとなる．IBM の DIA 等の業界標準を採用するメーカ間では，互換性ある文書交換が可能にならなくてはならない．

ファイルとジョブの交換サービスに加えて，汎用の情報保存および検索サービス，電子メール，および分散アプリケーションを提供しなくてはならない．これらの基本機能を利用することで，将来，ドキュメント・コンテント・アーキテクチャ（DCA）など，IBM を中心として進展中の業界標準へ向けて，サービスを拡大していくことができる．

これらのサービスはパソコンからメインフレームまで，あらゆる規模と種類の製品レベルで実現されなくてはならない．各サービスの構造は当然，ISO が提唱する OSI プロトコルに準拠しなくてはならない*．その全体をインテグレーションするには，高性能で高効率のディクショナリとディレクトリが必要となる．

4.2 オフィス環境

オフィス用途のプロダクトはその設計の基本を，メインフレームとオフィスおよび個人レベルのコンピュータ環境を連結し一元化することに置いている．これをユーザ側から見ると，メインフレームによる集中型システム環境から，完璧な分散型システム環境までの範囲内で自由に選択することができる．

前述したように，これら多様なシステム環境に共通するのは，文書交換用のストラクチャで，オフィス環境にとり理想的な考え方のように見える．また文書サービスに並行して音声情報処理システムもあり，すでに音声によるメッセージ交換が可能になっている．

このような状況下のオフィス・システムで最も注目すべきは，UNIX**環境であろう．UNIX オペレーティング・システムは部門向けの情報処理用途でもますます広がってきている．事務部門の業務負荷は，そのほとんどが部門レベルで発生することから見ても，UNIX の大きなターゲットはここにあると考えてよい．先に述べたように，OSI とのインタフェースと標準に準拠することがインテグレーションにおいては重要である．UNIX はまさに OSI および業界標準を提示しており，1990 年代のオペレーティング・システムとしての優位さが高まり始めるだろう．

4.3 データ処理

インテグレーションを考える上で最も複雑な事柄は，現に業務用として稼働中のデータ処理環境である．なぜ複雑なのか．その理由は主に多数のソフトウェアが稼働するためである．とくに第四世代言語の領域が複雑である．

ここではインテグレーションの見地から EDP 部門が細心の注意を払うべき，データの本番業務処理にかかわる事項をいくつか検討してみたい．いずれもユーザのアプリケーション・システムから見ても実務的な関心を呼ぶところである．

1) トランザクション処理
2) データベース・マネジメント一般
3) 第四世代言語の業務使用

---

*ISO では文書内容を ODA (Office Document Architecture)，文書交換形式を ODIF (Office Document Interchange Format) で規定している．

**UNIX は AT&T ベル研究所の登録商標である．

4)　稼働環境

4.3.1　トランザクション処理

トランザクション処理を通してシステムのインテグレーションを考えるとき，相互接続した複数のコンピュータにより，きわめて高いトランザクション処理能力を達成する統合化トランザクション処理システムの実現可能性と，そこで得られる実質的な効率増を思い描く．これは，またリカバリ機能の強化と障害時待機システム構成（ホットスタンバイ・システム），ならびに障害許容範囲により実現する，システムの弾力性にも影響を与える．

複数ハードウェアを相互接続するには，前述した分散システム・サービスも必要となる．プログラム間の情報授受，および対話方式による情報交換機能により，相互に接続したシステム上に存在するソフトウェアのコピー間ではハイレベルな手段で直接，交信することが可能となる．

4.3.2　データベース・マネジメント

データベース・マネジメント・システム（DBMS）は，システムの設計上，重要な部分となる．DBMS の設計には，各種データベース・モジュールをインテグレーションする問題に直面しなくてはならない．複数の異なるデータ管理方式（Codasyl 型，RDMS，メーカ独自のファイル・システムなど）が支援できなくてはならず，また将来のことを考えてオープン・システム・アーキテクチャに準拠させる必要があるだろう．

このような考慮をしておけば，システム・レベルでインテグレーションした単一構造の共通データベース・マネジメントを構築することが可能である．

表1は標準コンパイラと照会／更新機能から成る第三世代アプリケーション環境と，アプリケーション・ジェネレータや構造化言語などで代表される，より新しい第四世代アプリケーション環境との関係を示している．ここから今後の展開の方向を読み取ることができる．

表1　多様なアプリケーション環境の支援
Table 1 Multiple application environment support

4GL 環境	アプリケーション・ジェネレータ	SQL 照会/更新
データ・モデル	従来型	Codasyl 型& リレーショナル
3GL 環境	QLP 照会/更新	3GL コンパイラ

つまり，リレーショナル・データベース・マネジメントを採用することが，ソフトウェアのインテグレーションを実現する目標の一つでなくてはならないことがわかる．同時にアプリケーション環境に応じたデータ・モデルが選択できる，広汎な基盤を持たなくてはならない．

見方を変えると各種データ・モデルとアプリケーション環境要素が存在することと，3種類のオペレーティング・システム上で稼働することを勘案すれば，Codasyl 型ではなく，リレーショナル・モデルが異種システム間を統合するモデルだと言うことができよう（表2）．

表2 パソコン対メインフレームの整合性
Table 2 PC to mainframe consistency

	メインフレーム	UNIX	MS-DOS
SQL Query/Update	あり	あり	あり
リレーショナル	あり	あり	あり
Codasyl	あり	—	—
従来型	あり	あり	あり
4GL アプリケーション・ジェネレータ	あり	あり	あり
3GL コンパイラ	あり	あり	—
MAPPER	あり	あり	あり

さて，ここでデータベース・システムからアプリケーション開発言語に目を転じてみる．あっさりと第四世代言語が解答だと結論づけてしまうこともできる．しかし，本稿で論じているソフトウェア・インテグレーションの戦略の意味を確認するため，もう少し掘り下げた検討を加えていこう．

4.3.3 第四世代言語の業務使用と稼働環境

第四世代環境という考え方は，現実にインテグレーションを考える上で大きな関心が持たれるところであり，アプリケーションの開発方法とその実施方法についての基本的な想定条件に大きく影響する．ここではMAPPERプロダクトを本節のテーマから切り離して考えたい．確かにMAPPERは十分，ここで検討するに値するのであるが，まずは一般論から始めよう．

1) 第四世代言語とは何か……まずこれまでの第三世代のプログラミング言語よりも優れた言語があるかと自問自答してみる．答えは当然，イエスである．今や第四世代言語の時代である．第四世代言語はコンピュータの専門家がデータ処理部門で生産性を高めるための試みから生まれたツールであることは事実だ．その目的はソフトウェア開発の所要時間を短縮すること，また，所要資源，とくにデバッグやソフトウェアの保守に要する時間を節減し，最終的にプログラムの変更作業工数を大幅に削減することにある．

リレーショナル・データベース・マネジメント・システム（RDBMS）と第四世代言語は，将来にわたる潮流と考えられる．ソフトウェア保守上の問題を軽減するプロトタイピングもまた，第四世代言語を採用することにより実現可能となる．DBMSと連動する4GLは数多い．データに依存するエラーを避けるため，ディクショナリはDBMSに内蔵するのが一般的な考え方になっている[3]．

物事には裏表があり，ここで検討しているテーマについても例外ではない．Datamation誌によれば，われわれの見解は次のようになる：「いわゆるエンド・ユーザの多くは，第四世代言語などという高級な代物は使いこなせない．データ構造も十分に理解しておらず，必要に応じて臨機応変に，簡単にデータが取り出せるという期待はなかなか持てない．」（同誌アドバイザ，I. ネービット談）．

もう一方の旗頭にJ. マーチンがいる．衆知の通り彼は第四世代言語とは何かについてだけでなく，同氏が独自に設定した第四世代言語の基準を満たすソフトウェアを設計する方法について多大の力を注いでいる．

J. マーチンの高生産性言語に関する論文は，第四世代言語のあらましをうまく記述している．同論文によれば，次の特性を持つ言語が第四世代言語であると定義する[2]．

① 第三世代言語を用いた場合と比較して，プログラム開発の所要時間が 10 分の 1 以下になること．

② 使い方が簡単で修得が容易であり，覚えやすい言語システムであること．（ここでは「使い方が簡単（easy to use）」を挙げているが，これは必ずしも，ユーザ・フレンドリー（ユーザが馴染みやすいこと）を意味しない．）

③ データベース・マネジメント・システムを含む諸要素を，十分な程度まで統合化していること．

④ 情報処理の専門家とエンド・ユーザの双方が，より効率よく使用可能であること．（ここで言うエンド・ユーザとは，プログラムやアナリスト，その他の EDP 専門家以外のユーザのことである．）

これをより的確に，エンド・ユーザおよびアプリケーションを作成する人々の側には，どんなメリットがあるかという視点から 4 GL を眺めてみよう．

エンド・ユーザのメリット
• 開発の生産性が向上し効率よく作業できる．
• プロトタイプ・システムが作れる．
• 可搬性，移植性のあるソフトウェアができ上り，マルチベンダ環境に適する．
• 保守が楽にできる．
• ユーザ・部門，全社レベルでシステムのセキュリティが保てる．
• 十分なシステムの応答時間が得られる．
• パソコンおよび関連するソフトウェア，システムとインテグレーションが図れる．
アプリケーション作成者のメリット
• 生産性が向上する．
• アプリケーションのプロトタイプが作れる．
• 第三世代の手法と一元化が可能なインタフェースが使える．
• 業界標準のハードウェア／ソフトウェアが使用できる．
• 教育訓練が楽で，ツール選択の幅が大きい．
• 保守が楽にできる．効率のよいアーキテクチャが利用できる．

第四世代言語の理解を一層深めるためには，手続き型言語と非手続き型言語の機能性について検討しなくてはならない．この二つは簡単に判別できる．手続き型言語は，ある仕事をする“やり方”（方法）を記述する．たとえば，COBOL や FORTRAN，あるいは ALGOL でアプリケーションを記述するときは，目的とする処理のやり方を正確に各言語で書く．

一方，MAPPER や ALLY など，非手続き型言語で同じアプリケーションを記述するときは，その“やり方”ではなく，“何を”するかを書くのである．

次にデータベースおよびデータベース・マネジメント・システムと第四世代言語との関係について若干，触れておきたい．

2) データベース・マネジメント・システムと第四世代言語……DBMS はデータベースの主要なタイプ（階層構造，ネットワーク，リレーショナル）が単一の言語

体系で取り扱えなくてはならないとする4GL支持者は多い．これにより，アプリケーションに最適なDBMSを選択する幅が得られると考えられているからだ．DBMSの理想モデルは，ANSIが提唱する3階層スキーマ・アーキテクチャであると言われている．

このスキーマ・アーキテクチャは三つの明確な階層に分けることができる．第1層はユーザがどのようにデータを見るかを定義するもので，データベース内の物理的位置付けや場所からは切り離される．この定義はトップダウンおよびボトムアップ方式のデータモデリング技法によって行う．物理的階層は内部に格納してある実データとその定義，および格納の方法で構成される．

実務的には，一連のスクロールと非手続き型および手続き型言語のサブセットを用いて，スキーマとテーブルを作成，変更，更新する．これらの操作は直接，有効なディレクトリと連動している．そのためデータベース管理者は多数の異なるストラクチャを簡便かつ迅速に作成，プロトタイピングでき，また検査することが可能となる．

第四世代言語はコンピュータ・プログラミング言語の発展過程において，次の段階に位置しており，ソフトウェアを完全にインテグレーションするために必要なツールとして存在し続けるであろう．前述した第一世代から第三世代までは，それぞれの先行世代におけるプログラミング技法と比較して，大きく抜きん出た方法論を展開した．

一般に第四世代言語は，ある作業を完了するための所要時間と必要な技能を節減することにより，生産性向上をもたらす可能性を持つツールだと考えられている．加えて，この4GLはアプリケーションを保守，変更するのに都合のよいツールである．プログラムのライフ・サイクル全体に費やす資源トータルの大部分が，まさにこの分野へ充当されることを考えれば，それも道理であることがわかる．

最後に重要な点として，4GL環境がディクショナリおよびディレクトリという形式で，ソフトウェア・インテグレーションの“スーパー・ツール”を提供することを挙げておきたい．

3) アプリケーション開発と第四世代言語……アプリケーションの開発生産性を高めることは，明らかに4GLの顕著な特性の一つである．問題は誰が4GLをアプリケーション開発に利用できるかである．データ処理の専門家か，あるいはエンド・ユーザかである．アプリケーションにはあらゆる規模と複雑さの違うものがあるが，それらすべてをデータ処理専門家かあるいはエンド・ユーザのいずれかがデザインできるだろうか．あるいは，複雑なデザインのシステムは専門家だけしか手が出せないのだろうか．アプリケーション開発に採用するシステムが何であれ，エンド・ユーザが主要アプリケーション開発に簡単に使用できないならば，生産性向上の期待と現実には大差が出るに違いない．

4GLシステムの言語とデータベースの考え方はまだ曖昧で複雑であり，重要アプリケーションのデザインに使用できるのは専門家に限定されてしまって，エンド・ユーザは単純なレポート作成程度でお茶をにごすとすれば，積み残しされてきた未開発アプリケーションの一部が少々解消する程度に終わるであろう．とこ

ろが，仮に4GLの言語とデータベースの概念が単純でエンド・ユーザでも十分にアプリケーションのデザイナになれるとしたら，コンピュータ利用の潜在的な生産性の伸びを阻害するのは，ユーザ側の創造性が貧困か，あるいは革新的発想力の不足くらいしか考えられないことになる．

一部に4GLを警戒の眼差しで見るプログラマがいても驚くにはあたらない．プログラマは誰でも，初めに脅威を感ずる．エンド・ユーザがプログラムを書けるなら，プログラマは不要になるのではないか，という懸念を持つためだ．4GLを拒絶するプログラマも多い．これまで長年，COBOLやFORTRANでプログラムを書いて給料を貰ってきたのだ．今さら若い者と一緒に新しい言語を修得したくはないというのが率直な感想だろう．ここで彼等プログラマが見落しているのは，プログラマ自身の役割が変化しているという点だ．

現代のプログラマはコーディングのような煩わしい作業を減らして，4GL利用のコンサルティングに時間を注げるはずである．彼等プログラマこそ，ソフトウェアのインテグレーションを実現できる人々であると指摘しておきたい．

プログラマがよく陥る誤りの最大のものは，4GLを第三世代言語のカッコの良いものだと考えて3GLと同じように使ってしまうことだ．4GLをCOBOLやFORTRANの代用として使用すれば，その結果は悲惨である．どのようなツールでもそうであるが，4GLも正しく使用して初めて価値が生まれる．プログラマ諸氏は仕事の生産性を上げ，エンド・ユーザ支援ができるまで，もっと熱意を持って4GLを修得すべきである．また前述したようにプログラマがソフトウェア・インテグレーション実現のキーとなるが，それは第四世代のシステム開発環境を理解し，活用して初めて言えることでもある．

だがこれで十分とは言えない．生産性向上ツールを手にするだけでは生産性向上が約束されたわけではないのである[4]．アプリケーションのバックログ解消を目的としてシステム開発ツールへの投資を進める産業界は，すでにこのことがわかり始めている．事実，いわゆる生産性ツール，すなわち独自アプリケーションのデザインやプログラミングに関する全部または一部の作業を機械化するためのソフトウェア／ハードウェア製品を利用すると，システム開発の過程に新たな遅延が生ずるという事実を指摘する専門家がある．

「システム開発の生産性を高めなくてはという圧力から，生産性ツールの導入に躍起となってきた企業の情報処理部門は数多い．しかし，導入したツールが開発工程になじまなかったり，あるいは誤った工程を自動化するだけだと，まったくの期待はずれに終わる．生産性ツールは，確かに開発生産性とアプリケーションの信頼性を驚異的に改善する可能性を秘めているが，同程度以上に問題を起こす可能性もある．」(G.ファーガソン氏，アーサーアンダーセン社談)．

ファーガソン氏によれば，生産性ツールを採用した開発環境で効率を阻害する要因は，次の4点である．

① 果たすべき役割の誤り：企業内の業務や風土に根差した使命でなく，技術上の使命だけを考えて採用する誤り．

② 適用業務選定の誤り：自動化に値しない工程を自動化してしまう誤り．

③ インテグレーションへの対応なし：導入ツール相互間にインテグレーションの手段がない．

④ 理念の欠如：マネジメントに長期的対応策を講じようとする理念がない．

生産性ツールは導入こそ比較的簡単であるが，人間の作業環境というカルチャーが変化するためマネジメントの継続的かつ可視的な支援が必要である．「ツールがあれば技術的には迅速に問題を“解決”できるが，その問題はトータル・ソリューションのほんの一部に過ぎない．つまり，生産性ツールを選定するときは，通常のシステム開発プロジェクトを推進するときと同等の方策を用いなくてはならない．

ツールを購入する前にまず計画を立てることだ．次に，目的のソフトウェア開発期間を通して，また開発以後も支援されなくてはならない．ツールの導入に際しては，適切なチャーター（宣言書）を作り，適切な開発工程を自動化し，開発手法を考案しなくてはならない．それに目に見えるマネジメントの支援を与える．生産性ツールを導入するとは，こういう一連の事柄を意味するのだ．」（ファーガソン氏）[4]．ここで追加するとしたら，開発手法にはソフトウェアとハードウェアの各システムのインテグレーションについて，明確な方向性を持たせることである．

業務上必要な処理をいつもプログラマに説明するだけの人ではなく，自分で直接，コンピュータを使用する人は，物事を二つの局面から眺めようとする．この点が彼等の最大の強みであろう．われわれ，情報処理を専門とする人間はこのような社内業務に精通した人々に4GLの使い方を数日間，数週間，まれに1か月を超える期間，指導すべきである．その結果，本来の業務経験を積んだ人々がコンピュータで武装して，各自の仕事の第一線に出動してくることになる．これをプログラマに同じ期間，業務についてのすべてを教え込んでから，アプリケーションを開発させる場合と比較してみるとよい．どちらの方法が効果的であるかは，容易に想像がつくであろう．

現代に生きるわれわれはいま，人間とコンピュータの歴史において，きわめて興味深いマイルストーンに立たされている．ソフトウェア技術の進歩により，コンピュータのサービスを必要とする人（エンド・ユーザ）には4GLを用いて直接，コンピュータを利用できる道が開かれた．

また個人でもパソコンが持てるようになり，大企業の部門ごとに自前でコンピュータが購入できる時代になった．コンピュータが簡単に入手でき，プログラム作成も容易だとなれば，従来型のプログラマのような中間層は不要になる場合が多くなる．データ処理の専門家は自らの役割がエンド・ユーザのコンサルタントになりつつあることを自覚してきている．4GLも含めた情報処理環境をプログラマとエンド・ユーザの双方に等しく与えて生産性を高めることを考えなくてはならないのだ．エンド・ユーザに直接，プログラムを書かせることで，対象業務のいわく言いがたい微妙なニュアンスをもアプリケーションに反映させることが可能になる．

厳密に仕様を決め，プログラマ／アナリストがその仕様を完璧に理解してその

通りのプログラムが書けると信じてきた従来型アプリケーション開発では，このような期待は一切，持てないのである．

それではあらゆるアプリケーションがエンド・ユーザに4GLを渡すことで開発可能であろうか．それは4GLとして何を選ぶかにより，またアプリケーションの複雑さの程度により左右される．いわゆる4GLと言われるものすべてが，同等の機能を備えているわけではないからだ．最も優れた4GLを手にしたとしても，何がしかの実務をエンド・ユーザだけでアプリケーション・プログラムに仕上げるのは，必ずしも容易なことではない．

4GLを活用しようとする企業においては，まずエンド・ユーザが4GLを用いて自らのアプリケーションの一部を作成してみる．その成果がデータ処理部門の要求を満たすものであれば理想的である．そうすればデータ処理部門が4GLを生産性ツールとして，もっと複雑なアプリケーション開発に採用できるだろうし，エンド・ユーザへの技術的支援にも熱が入るであろう．こうして情報処理の専門家とエンド・ユーザの双方が同一の4GLを駆使できるようになると，相手側の経験をお互いに利用し合うことが可能になる．これがチーム・コンピューティングと称する考え方である．

4.4 意思決定支援

さてここで，四つの機能分野のうち最後の一つである意思決定の支援について検討する．ここでは3種類の4GL，MAPPER，LINC，ALLYのうち，MAPPERに焦点を当ててみよう．

MAPPERによる意思決定支援システムにおいては，プログラマなど従来型アプリケーション開発の要員やツールが不要であること，またエンド・ユーザ自身が独自のアプリケーションに対する考え方を仕事の場でデザインし，構築して本稼働させることができるので有利である（表3）．

機能豊かなMAPPERを採用することにより，多数のユーザがそれぞれの意思決定支援ニーズを一つのシステムをアクセスして得ることができる．しかしパソコン時代のエンド・ユーザはさらに柔軟な機能を求めているので，MAPPERとその他のプロダクト環境との統合化を鋭意，進めている最中である．

MAPPERプロダクトはデータ通信機能を内蔵している．そのためMAPPERシス

表3　MAPPERによる意思決定支援
Table 2　Decision support via MAPPER beyond the fourth generation

- ●アプリケーション開発と本番稼働の統合化
- ●広汎なプロダクト群の支援
- ●低支援コスト
- ●総合的エンド・ユーザ向け機能
  - ■意思決定支援環境
  - ■アプリケーションツール，ユティリティ
  - ■グラフィックス機能
  - ■ワード・プロセシング
- ●システムの機能分散
- ●全システムを対象とした統合化

テムを相互に接続し，アプリケーション・レベルで情報の授受が可能である．たとえば，パソコンMAPPERユーザがパソコン上で業務の一部を実行し，その他の部分は相互に接続したミニコンやメインフレーム・システムで処理しているという例などは，まさに分散アプリケーションを実現しているケースと言えるだろう．

現在，MAPPERプロダクトはパソコンからメインフレームまでを対象とした統合化4GLシステムだと自信を持って言うことができる(図2)．そのユーザは各種システムを混在，適合させて，他社が提供するいかなるアプリケーション環境よりも柔軟性，経済性に優れた合目的業務システムが構築できる．またUNIXを基本とする部門システムとMAPPER Cを融合させたプロダクトの登場により，MAPPERワールドの階層構造が明確になったと言える．

MAPPERの統合化に対する方向を見れば，階層データ・モデルおよびリレーショナル・データ・モデルの両面から，データベース・プロダクトと緊密な関係を確立するという目標を継続していくことがわかる．その他のデータベースについてはデータを抽出し，MAPPERで実際に操作するツールを用意していくことになる．

データの分散処理について見れば，MAPPERが対話方式でMAPPER環境外部のプログラム，たとえばALLYやLINCなどのシステムのプログラムを実行する機能がとくに注目に値しよう．

MAPPERのエンド・ユーザは，機能拡大の恩恵を享受できることから，とくに注目を集めている．MAPPER VIEWと称するプロダクトはユーザ・インタフェースの一つを構成している．ユニシスはMAPPER KITおよびPC-Onlineプロダクトの考え方を継承していく．

さらに広範囲にわたるシステムの統合化を達成すれば，システム内にある全データベースをすべてのユーザが，管理とセキュリティの要請を踏まえた上でアクセスできるようになる．競合システムのデータベースも含めた多種多様なシステムをある種の分散処理環境と相互接続するには，自由なアクセスを可能にする共通手法の土台が必要である．また，このことは単一の情報環境を考えるユーザにとっても必要不可な問題となる．

MAPPERプロダクトは，4GLの基準に準拠しているのは明らかである．むしろ4GLを超えていると言いたい．MAPPERを使用すればプログラマや一般ユーザの生産性を一層，向上させることができる．また，両分野の人々が自らアプリケーションを作成できることも確かである．また，驚くほど生産的な全社アプリケーションをあらゆる分野で開発可能なことも多言を要しまい．その一方で，ごく簡単なアプリケー

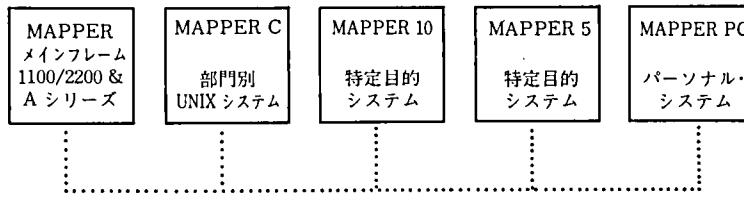


図2　分散情報システムを統合化するMAPPERシステム

Fig. 2　MAPPER system integrating distributed information

ションにも使用できるのだ.

4 GL はその多くが,データ処理専門家向けのプログラミング言語を目的として開発されたものである.エンド・ユーザはプログラマが作成したアプリケーションとかかわりを持つにすぎず,その効用もアプリケーションが可能な範囲に限定されたものであった.時折,発生する臨時のデータ検索用に,ユーザが親しみやすいインタフェースは設けてあっても,いざデータを更新するとなれば同様にやさしいインタフェースは用意されておらず,更新もレコード単位というのが一般的である.これでは,エンド・ユーザはアプリケーションなど開発できるものではない.

これまで何年にもわたり,MAPPER には数多くの機能が追加されてきた.最新レベルのユーザは数百を超える機能が利用可能になっている.データは MAPPER システムの外部から得ることができる.またカラー・グラフィックス機能の他,ユーザ間通信,アプリケーション通信機能を利用した分散アプリケーション機能が支援されている.また,データ・ディクショナリやドキュメンテーションを含む各種のアプリケーション開発ツールが用意されている.

MAPPER システムは独自のデータベース・マネジメント・システムを備えている.この DBMS はその柔軟性と実行速度において,従来の DBMS をはるかに上回るものである.リレーショナル・データベースよりも豊富な機能を持ちながら,高速性能を実現するため基本的考え方は単純になっている.一般に 4 GL は従来型の階層型データベースなどを採用していることが多く,そのため機能が硬直化しデータベース管理も複雑化をまぬがれないことが多い.

さて MAPPER システムはまた,トランザクション・プロセッサでもある.つまり,オンライン,リアルタイムで稼働する大量データ処理システムで,データを直接更新する.ユーザはターミナルの画面から対話方式でデータを扱う.メインフレーム上では同時に数千ものユーザを取り扱うことが可能で,その応答時間を数秒以内にデザインすることもできる.現在市販されている 4 GL の大半は,このような多数ユーザの処理は困難である.

エンド・ユーザは MAPPER システムを利用してアプリケーションが作成できる.アプリケーションを作り上げる構成ブロックとなるのが MAPPER ファンクションと呼ばれる機能である.エンド・ユーザはこのファンクションを組み合わせて記述することにより,実務処理ができる実働システムが作れる.一般に 4 GL の多くは,むずかしいプログラミング機能を用意することでアプリケーション開発ができると称していることが多い.その多くがエンド・ユーザだけでアプリケーションを記述するにはむずかしすぎるのが実状である.

エンド・ユーザが主体となり業務分析や,辞書作成,文書作成などができるツールを備えたデータ処理コンピュータ利用型の 4 GL となれば,MAPPER をおいて他にはない.データ処理部門はまた,こうして開発したアプリケーションをエンド・ユーザに保守させることができるのである.

MAPPER は,個々のユーザ別にシステム資源の利用状況を記録するアカウンティング・システムを内蔵している.この機能は,システムのサービスを利用する部門およびユーザにコスト負担してもらう上で役に立つ.このような機能を備えた 4 GL は

他にはない．普通一般の4GLはオペレーティング・システムがとらえたプロダクト全体の資源利用情報しかなく，ユーザ別の課金情報が得られないためである．

フル・リカバリ機能を備えていることもMAPPERシステムの大きな特徴の一つである．万一，ハードウェアあるいはソフトウェア障害のためにデータが破壊された場合でも，自動的に回復させることができる．一般の4GLは，あらかじめバックアップしてあるファイルをリロードする以外，データを回復する手段を持たないのが普通である．MAPPERシステムは複数レベルのセキュリティ機能を持っており，システムへアクセスするユーザは一日の時刻，アクセスするデータのタイプ，ユーザ別に制約可能なファンクション，レポート別に設定可能なパスワード，ユーザ・ラン（アプリケーション）ごとの資源消費量，およびレポートの暗号化など，多彩な機能により情報の保全が図られている．システムの動作をモニタリングする機能によって，システムをチューニング（手直し）する基礎データが得られる．いわゆる4GLを呼称するものの大半は，限定的な範囲でしかセキュリティ機能，モニタリング機能，システムのチューニング機能などが使えないことが多い．

MAPPERシステムには，さまざまな国語に対応したバージョンが用意してある．システム・メッセージはすべてファイル化してあり，さらにその他の国語に翻訳することもできる．また，一つのMAPPERシステム内で複数の国語を支援することも可能である．システム・メッセージは，ユーザのサインオンに応じた国語で表示される．それぞれの国語特有の通貨記号や照合順序，句読点は正しく使用されている．

MAPPERシステムはパソコン，UNIXコンピュータ，およびメインフレーム上で稼働する．明らかに4GLと名乗る資格は十分にある．4GLはそのユーザの生産性を向上させるだけではなく，企業活動の効率化に役立つ点がより重要である．これが4GL技術の期待される所以である．すでにコンピュータ利用が進む中で，データ処理部門は企業組織に最適な4GLを選定支援しなくてはならない．

コンピュータを必要とするところでは，仮に現在，データ処理部門がなくてもMAPPERシステムでその利用が可能になる．MAPPERシステムはエンド・ユーザだけの手で，システムに内蔵する自動化ツールを使用することにより十分，導入することができる．

5. UNIXの世界

システム統合化を自在に進めるための手段として，高性能の部門コンピュータとAT&TのシステムV UNIXが大いに役立ってきた．

本節では，UNIX，MAPPER，エキスパート・システムという三つの概念について検討する．いずれの考え方も相互に役立つところがあり，現在われわれが知るデータ処理の現実をはるかに超える強力な三位一体のシステムを作り上げる要素となり得よう．一見すると，これらのプロダクトが一つにまとまることなどあり得ないように思える．

たとえばMAPPERはこれまでアセンブラ言語で書かれており，単一メーカのソフトウェア上でのみ稼働するプロダクトであった．また，エキスパート・システムは人工知能という特別な概念を必要とする世界にあり，LISPやPROLOGなどの特殊言

語，また近年は特殊目的ハードウェア・アーキテクチャ上で稼働するプロダクトである．

一方，UNIX を基本とするシステムでは高水準言語 C を使用してきている．UNIX を貫くデザイン思想は，異なるメーカの異なるアーキテクチャ上で移植性に優れた汎用コンピューティングを実現しようとするところにある．この考え方を方法論として実現した UNIX は，アプリケーションのデザインと稼働の互換性を保つ格好の環境を作り出している．これこそマネジメント情報システムやデータ処理を担当する企業組織が長年，渇望していたものであったのだ．

UNIX はパイプ，入出力リダイレクション，シェル命令，および十分すぎるほど豊富で安定したファイル構造を基礎とした統一構造を備えている．UNIX を用いれば明らかに，機能とインタフェースを満載した巨大なツールボックスを共有し，UNIX と同じ方法論でアプリケーションを組み立てることができる．その結果でき上がるアプリケーションは，その他のアプリケーションと互換性を保つことになる．ある人が作成した主プログラムは，ローカル/ワイド・エリア・ネットワークを経由して他の人の分散情報システムの一部となり得る．

またプロセス間で容易にデータ操作ができるのは，UNIX システムであればこそ実現できた互換性によるところが大きい．MAPPER システムなど問題対処型システムは，他のコンピュータやアプリケーションと通信できる機能が必要である．問題を解決する環境は，既存データベース上にある実データの情報処理ができるか否かに依存するからである．現実世界の情報が得られなければ，貧弱で不適切な，陳腐化した情報によってしか問題に対処できないことになる．

エキスパート・システムが重要な点は，それが知的システムであることではなく，特定分野の専門家の知識を包み込んだシステムであることだ．しかし人間の専門的能力をとらえていることだけが重要なのではない．エキスパート・システムは時系列的な知識ベースを持たねばならず，そこにおいて MAPPER プロダクトが役立つと考えられる．MAPPER システムの機能性，柔軟性，適合性をもってすれば，知識ベース構築などはそう困難ではないであろう．

この三位一体の構成においては，それぞれが必要不可欠な要素である．UNIX，MAPPER，エキスパート・システムの三つが結合して初めて，第四世代をはるかに凌駕する最新技術となる可能性が出てくると考える．

6. おわりに

システムの統合化は単なるプロダクトの機能ではなく，またシステムとアプリケーション，そのユーザ間に横たわる障壁を取り除くことに終始するアーキテクチャ/デザイン上の考え方でもない．それはデータへの自由なアクセスこそ，複雑な人間の組織の要望を満たす単一で柔軟な情報環境を構築することを意味する．システムおよび企業組織は，数多くの規制と制約条件をその自由さに課するであろう．しかしデータの自由な交換という基本的考え方がなければ，技術的な制約で束縛された世界でアプリケーション環境を設計するしかなくなる．

従来，データ交換に自由さがなかったため，やがてその不自由さが当然のこととし

てシステムの中にはびこる結果となった．システムとそのアプリケーションのほとんどが，それ自体で独立した存在であったためだ．そして効率的であることが柔軟さよりも重要視されてきた．これこそ，われわれ情報処理を専門とする人間自らがその考え方と対応を変えなくてはならないことを示唆する，最も明瞭な現象の一つであったのだ．情報処理部門こそ，いわゆる“第三の波”[14]を生き残るため，システム，アプリケーション，そしてユーザとの間の最も開かれた関係を模索しなくてはならないであろう．

（統合OAシステム部　横山正敏　訳）

参考文献

[1] J. Martin, *Fourth Generation Language*, Vol.1 Savant Research Studies, Carnforth, England, 1983.

[2] J. Martin, P. R. Mimno, *The James Martin Report on High-Productivity Languages*, Vol.1, High Productivity Software, Inc., Marblehead, Massachusetts, 1986.

[3] G. Parikh, *Handbook of Software Maintenance*, John Wiley & Sons, Inc., New York, New York, 1986.

[4] *EDP Weekly*, Vol.27, No.47, Computer Age ®, Springfield, Virginia, Dec. 1986.

[5] G. M. Booth, *The Design of Complex Information Systems*, McGraw-Hill Book Company, New York, New York, 1983.

[6] J. Martin, *Applications Development Without Programmers*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1982.

[7] *A Guide to the Relational Syntax Analyzer*, RSA Level 2R1, Sperry Corp., Blue Bell, PA, 1985.

[8] L. Schlueter Jr., *4GL For User Designed Computing*, Sperry Corp., P.O. Box 500, Blue Bell, PA.

[9] L. Schlueter Jr., *Large Scale User-Designed Computing*, Sperry Corp., P.O. Box 500, Blue Bell, PA.

[10] M. T. Schroeder, "What's Wrong with DBMS", *DATAMATION*, Cahners Publication, Vol.32, No.24, 67-70, Dec. 15, 1986.

[11] C. J. Date, *An Introduction to Database Systems*, Vol.II, Addison-Wesley Publishing Company, Reading MA, 1983.

[12] J. Bill, *MAPPER System Software Abstract*, Unisys Corp., P.O. Box 500, Blue Bell, PA, May 1986.

[13] J. Bill, *Sperry's MAPPER ® System-In the Spirit of Relational Database Concepts*, Unisys Corp. P.O. Box 500, Blue Bell, PA, Sep. 1984.

[14] A. Tossler, *The Third Wave*, W. Morrow, Fairfield, N.J., 1980.

執筆者紹介　William R. Gray

Purdue大学卒業．電気工学の学位取得．ビジネス・アプリケーション，プロセス・コントロールから人工衛星の軌道操作のシステムに至るまで，オンライン，リアルタイム・システムの研究および開発に従事．Unisys MAPPERシステム開発の当初より参画，現在も同プロダクトの第一人者としての地位にある．

意味的なつながりを考慮した 複合名詞のかな漢字変換システムの構築

Kana-Kanji Translation System Utilizing the Meanings of the Stem of Derivatives as the Connection Rule to Proper Suffixes

稲永紘之, 新谷隆之

要　約　日本語文書処理における入力方式は，単漢字変換から文節変換を経て，今や連文節変換，文章一括変換等ますます便利になってきた．最近では複合語を一度に変換する方式も現われたが，正しい漢字への変換精度は今一つである．

ところで(派生語を含む)複合名詞の末尾の語は，それ以前の語(群)の接尾語として機能していることが多く，両者の間には意味的なつながりがある．この関係を調べておけば，複合名詞のかな漢字変換に当たって同音異義の接尾語があった場合も，どの接尾語が最適であるかの判断に利用でき，複合名詞の漢字変換の精度向上が期待できる．そこで，複合名詞の接尾語ごとに，それに係り得る語を収集・編集して複合名詞の接尾語辞書とし，できあがった辞書を用いて複合名詞のかな漢字変換を行うシステムを構築し，簡単な実験を行った．

この結果，意味的なつながりを考慮した接尾語辞書を用いることによって，ただの漢字の羅列ではなく意味のあるかな漢字変換を行えることが検証できた．

Abstract　This paper reports the results of Kana-Kanji translation utilizing a special suffix dictionary. Along with the homonym processing, it is crucial to study how to handle unknown words like derivatives in machine processing of Japanese sentences including Kana-Kanji translation. Even massive addition of headwords does not ensure the full coverage of derivatives. Nothing but an improvement of prefix/suffix dictionary both in quality and quantity could solve the problem.

This paper introduces a new suffix dictionary featuring the meanings of the stem of derivatives as the connection rule to proper suffixes. Utilizing the suffix dictionary, it is studied that Kana-Kanji translation for derivatives is improved and thus it offers more reasonable translations.

1. はじめに

日本語文書処理における入力方式は，単漢字変換から文節変換を経て，今や連文節変換，文章一括変換等ますます便利になってきた．当社でも，日本語処理の最新技術を導入すべく，九州芸術工科大学と協力して日本語処理に必要な機械辞書の整備/拡充に携わってきており，61年度は複合名詞のかな漢字変換用辞書の研究を行い，一つの成果を得たので報告する．

日本語の文章に複合名詞の多いことは例をまつまでもなく，それらの存在が変換精度を低下させる要因になっているが，すべての複合名詞をかな漢字変換用機械辞書に登録することは，実用的でない．次善策として接頭語/接尾語辞書により文節変換方式の改善が図られてきた．

複合名詞の構造に注目すると，末尾の語はそれ以前の語(群)の接尾語として機能していることが多く，両者の間には意味的/文法的なつながりがある．したがって，つな

がりの度合いを調べておけば，同音異義語の接尾語が存在する場合でも，どの接尾語が最適であるかの判断に利用でき，複合名詞の漢字変換の精度向上が期待できる．

接頭・接尾語を含む自立語が多数登録されている九州芸術工科大学の自立語辞書(KID-J 86)から複合名詞を自動的に抽出し，その末尾の語(接尾語)ごとに係り得る語(係り語)を収集して複合名詞の接尾語辞書とした手順についてはすでに紹介した[6]．本稿では，この接尾語辞書を用いて複合名詞のかな漢字変換を行う機能を九州芸術工科大学のかな漢字変換システムに組み込み，実験した結果について述べる．

実験の結果，たとえば「ゴウカクシャメイ」に対して「合格者名」だけでなく「合格社名・合格車名」といった意味のある漢字への変換を行い，「合格社命」のような変換は行わないことが確かめられた．

2. 複合語の漢字変換の現状

はじめに述べたとおり，昨今のかな漢字変換方式の進歩には目をみはるものがあり，複合語を一度に変換する方式も現われている．ここでは，当社DS 7用のワードプロセッサの入力処理部(U-μJASTY)および複合語を一度に変換するパーソナル・コンピュータ用の某市販ワード・プロセッサについて，どのようにかな漢字変換が行われるか見てみることにする．

表1に示すものは，その市販ワード・プロセッサでは意図したとおりに変換されなかった例である．

表1 複合語のかな漢字変換テスト結果

Table 1 Kana-Kanji translation test result for compound words

No.	入力文字列	DS7E/U-μJASTY	某市販ワープロ	意図した変換
1	トシコウソウ	都市構想	俊子嘘う	都市構想
2	セイリョクブンヤ	勢力文や	勢力文や	勢力分野
3	ゴウカクシャメイ	合格社名	合格社命	合格者名
4	カイセツショ	解説書	解説所	解説書
5	シュッパンブ	出版部	出版不	出版部
6	コウエンカイ	後援会	工沿海	講演会
7	カヨウサイ	火曜祭	加養再	歌謡祭
8	カイガテン	会合点	会が点	絵画展

DS 7 E/U-μJASTYの一括変換では，DS 7での単文節変換より随分良くなったが「会・合点」等と無意味に自立語を組み合わせてしまうことがある．某市販ワードプロセッサで複合語変換を行っても，

1) 「都市構想」のところを“トシコ”に相当する自立語が選ばれたり，
2) 「勢力分野」の“ヤ”が助詞と判断されたり，
3) 「合格」，「社命」と言う関連のない二つの自立語の組み合わせとして変換されたり，
4) 自立語に不適当な接頭・接尾語が組み合わされたりして，

複合語内の自立語間，自立語-接辞間の連接関係を考慮しないで，かな漢字変換を行う場合の限界が感じられる．

テスト入力を見ると，一口に複合名詞と言っても，入力No. 1, 2のように単に二文

字漢字語の組み合わさったものと，入力 No. 3〜8 のように接尾語を含んだ複合名詞がある．

次に，このような複合名詞の構造について考察する．

3. 用語の定義と複合名詞の構造

国文法用語の定義は，解釈の違いから諸説がある．

ここでは，まず本稿で用いる複合名詞および接尾語の定義を行い，次に複合名詞の構造について考えてみる．

3.1 接尾語，複合名詞の定義

国文法では，意味の面からも，形態の面からも独立していて言葉の基本単位となるものを自立語と呼ぶ．名詞，代名詞，動詞，形容詞，形容動詞，副詞連体詞等である．一方，独立して一語を形成せず，造語成分として作用するものを接辞と言い，いわゆる接頭語・接尾語がこれに相当する．

この接辞と自立語からなる二次語を派生語と言い，たとえば「御｜指導」，「美し｜さ」をあげることができる．これに対して，自立語(または自立語に準ずるもの，たとえば形容詞の語幹)を組み合わせてできた二次語を複合語と言う．「社会｜運動」，「望み｜薄」等である．

このようにしてできあがった二次語の名詞が複合名詞である．

ところで，複合語の「社会運動」と「望み薄」を比べると，「社会運動」では間に自立語をはさんで「社会主義運動」のような別の複合語を作成できるが，「望み薄」は一つにまとまっている．末尾の語を比較しても，「運動」は独自に使用できるが，「薄」は自立語(形容詞「薄い」の語幹)ではあっても単独で名詞として使用されることはなく，接尾語的であると言うことができる．

そこで接尾語を拡大解釈して，上記の「薄」のような複合語の末尾の語をも接尾語とみなす．そして接辞を含む複合名詞を以降では単に複合名詞と呼ぶことにする．

複合名詞内の接尾語以外の部分は接尾語を修飾していることが多く，一種の係り受け関係が成立している．このような修飾語を接尾語に係るという意味で係り語と呼ぶことにする．

さて複合語の漢字変換テストで用いた入力 No. 3「合格者名」では，接尾語「名」に対して「合格者」が係り語であった．ところが，「合格者」の他にも「失格者」，「応募者」等多数の自立語が「名」と結び付いて「——者名」という複合名詞となる．

これらの自立語に共通している末尾の語「者」は，また一つの接尾語であり，かつ接尾語「名」に係る真の係り語であると言うことができる．

「合格者」，「失格者」，「応募者」等をすべて接尾語「名」の係り語として登録せずに，真の係り語「者」のみを接尾語辞書に登録することによって，複合名詞内の意味的な係り受け関係が記述でき，サイズも小さくすることができる．

次に，複合名詞の構造について，それを構成する語の品詞の面から考えてみる．

3.2 複合名詞の構造

■名詞＋名詞　　例：社長室

■名詞＋動詞連用形　　例：挨拶回り

■サ変動詞語幹＋名詞　　例：改造費
■動詞連用形＋形容詞語幹　　例：望み薄
■形容詞語幹＋名詞　　例：うれし涙
■形容詞連体形＋名詞　　例：良い子
■形容動詞語幹＋名詞　　例：割安品
■連体詞＋名詞　　例：我が家
■副詞＋名詞　　例：ニコニコ顔

以上，いくつかの例でもわかるとおり，さまざまな品詞の語が組み合わさって複合名詞を形成している．係り語と接尾語の意味的なつながりもさることながら，接尾語によって接続しうる係り語の品詞が特定できることがある．

したがって接尾語辞書の作成に当たっては，それぞれの接尾語に対して係り語の意味と品詞に注目することにした．

4. 接尾語辞書の作成

4.1 自立語辞書

接尾語辞書作成に当たって，使用した九州芸術工科大学の自立語辞書(KID-J 86)の概要は次のとおりである．

■登録語数：107522 語(名詞 93381 語)
■特徴　：日常新聞・雑誌等で用いられる複合語・派生語を多数含み，語と語の切れ目の情報を持つ．また，すべての名詞には表 2 に示す意味分類コードが付けられている．

表 2　名詞の意味分類コード
Table 2　Meaning category code for nouns

分類コード	意　味	分類コード	意　味
1	他の34の分類の何れにも属さないもの	J	生物
2	時，方向，数量等副詞的用法のあるもの	K	貨幣，金銭，給料，財産等経済関係の語
3	形容動詞的に使われるもの	L	建物の部分や部品
4	人間，神，仏，霊魂，妖精等	M	複雑な機構の機械等
5	動物名，動物，細菌	N	気体，雲，霞，自然現象の語
6	植物名，植物，植物の一部	O	音楽，曲，歌，芸能関係の語
7	鉱物名，鉱物，液体，自然物等無生物	P	時間，期間
8	飲料物，料理等	Q	衣服，織物，装身具の一部
9	人間や動物の体とその一部	R	スポーツや遊びの道具，用具
A	衣服，織物，装身具，履き物など	S	スポーツ，遊び
B	家，ビル，橋，堤等の建造物	T	時刻，時期
C	道具，日用品，簡単な機構の器具	U	人間が自然物に手を加えてできたもの
D	書籍，評論，文章，詩歌等読み書きの対象	V	部屋，室等建物の内部の場所
E	乗り物，交通機関	W	世界や活動分野等の抽象的な場所
F	病気，怪我，傷等	X	団体，組織，人の集まり，展覧会等
G	弾く楽器，楽器一般	Y	光景，様子，状態，形式
H	吹く楽器	Z	具体的な場所
I	打楽器		

4.2 接尾語辞書の作成手順

複合名詞用接尾語辞書の作成手順について簡単に説明する．図1が自立語辞書をもとに接尾語辞書を作成した手順である．

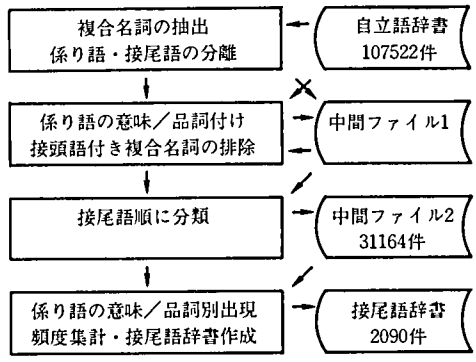


図1　接尾語辞書作成手順

Fig. 1　Flow of generating the suffix dictionary

■ タイプ1（接尾語）レコード

01		06	07	接尾語情報	122	123		128
CNSFX1			読み／漢字／意味／品詞			レコード番号		

■ タイプ2（係り語）レコード

01		06	07	係り語情報	122	123		128
CNSFX2			読み／漢字／意味／品詞			レコード番号		

※ 接尾語ごとに収集した係り語の分だけ作成される．

■ タイプ3（品詞／意味分類）レコード

01		06	07	24	25		52	53	54	56	58	60	62	64	66	68	70
CNSFX3			接尾語読み		接尾語漢字コード			カナ		出現頻度		品詞出現頻度					
												*1	*2	*3	*4	*5	*6

72		107	109	110	111	112		123		128
意味分類頻度			名詞出現頻度					レコード番号		
A	～Z	0	～9	*7	*8	*9	*10			

＊1：係り語がサ変動詞の複合名詞の出現頻度
＊2：係り語がその他の動詞の複合名詞の出現頻度
＊3：係り語が形容詞の複合名詞の出現頻度
＊4：係り語が形容動詞の複合名詞の出現頻度
＊5：係り語が副詞の複合名詞の出現頻度
＊6：係り語が連体詞の複合名詞の出現頻度
＊7：係り語が固有名詞の出現頻度
＊8：係り語が助数詞の複合名詞の出現頻度
＊9：係り語が人称代名詞の複合名詞の出現頻度
＊10：係り語が指示代名詞の複合名詞の出現頻度

図2　接尾語辞書フォーマット

Fig. 2　Suffix dictionary format

図2に作成した接尾語辞書フォーマットを示す.

接尾語辞書は論理的には可変長のデータ構造をしており，接尾語の読みの降順にタイプ1レコード一つ，一つ以上のタイプ2レコード，そしてタイプ3レコード一つがひとまとまりとなって一つの接尾語情報となっている.

4.3 接尾語辞書作成結果

接尾語辞書を作成した結果は表3のとおりである.

表3 接尾語辞書作成結果
Table 3 Results of suffix dictionary generation

自立語レコード数	107522
名詞レコード数	93381
接尾語付き複合名詞数	31164
接尾語数	2090
平均異なり係り語数	12.3

ここに収集した2090の接尾語はあくまでも自立語辞書から自動的に抽出した接尾語候補の語であり，現在九州芸術工科大学で取捨選択作業を行っている．また自立語辞書には地名接辞や人名接辞付きの自立語は登録されていないため，そのような接尾語の追加作業も検討中である.

5. かな漢字変換システムへの導入

5.1 接尾語辞書のインデックス・ファイル化

作成した接尾語辞書は128バイト/レコードの順編成ファイルである.そのままでは変換処理での辞書アクセスに時間がかかりすぎるため，インデックス・ファイル化した.

この結果，接尾語辞書形式は次のようになった.

・128バイト/レコード

・16レコード/ブロック

ただし，ブロック1，2はそれぞれタイプ1，タイプ3レコードのインデックスで，ブロック3からが辞書レコードである.

5.2 複合名詞かな漢字変換の文節モデル

九州芸術工科大学のかな漢字変換システムは，文節分かち書き入力を基本としているので，本来の複合語は2文節以上に分けて入力されるものとし，ここでのかな漢字変換の対象としない．また接頭語付きの文節も対象外である.

図3にこのかな漢字変換システムで取り扱う文節モデルを示す.

5.3 接尾語の接続検定レベル

この研究で作成した接尾語辞書は，各接尾語が，どのような言葉に意味的/文法的につながりやすいかの頻度情報を持っている.

また漢字はそれ自体で意味を担っているので，ある自立語の末尾の漢字が接尾語辞書の係りの語の末尾の漢字と一致するかどうかを調べることによっても，その自立語と接尾語の接続可能性の検査ができる.

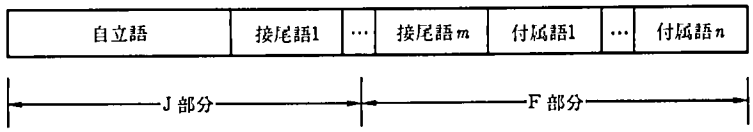
そこで次の四つの接続検定レベルを設けて，かな漢字変換の実験を行った.

レベル0：F部分の全体または先頭の一部が接尾語辞書にあり，J部分がその接尾語の係り語として登録されている．

レベル1：F部分の全体または先頭の一部が接尾語辞書にあり，J部分の末尾の漢字がその接尾語の係り語のどれかの末尾の漢字と一致して，かつJ部分と一致した係り語の意味分類コードが等しい．

レベル2：F部分の全体または先頭の一部が接尾語辞書にあり，J部分の意味的/品詞的結合率が2以上ある．

レベル3：F部分の全部または先頭の一部が接尾語辞書にあれば，J部分との接続検査は行わずに接続可能とする．



5.3節に述べる接尾語の接続検定において接尾語kの検査を行う場合，自立語および接尾語1～接尾語k-1を自立部(J部分)，接尾語k以降を付属部(F部分)と呼ぶ．

図3　かな漢字変換の文節モデル

Fig. 3　Bunsetsu model for Kana-Kanji translation system

5.4　複合名詞のかな漢字変換処理

5.2節で述べた文節モデルに基づく，かな漢字変換処理は図4のとおりである．

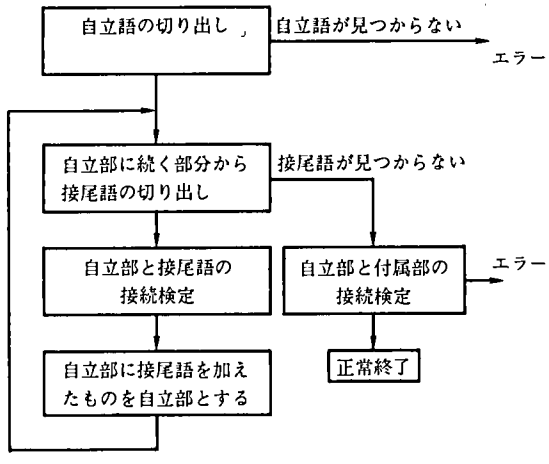


図4　複合名詞のかな漢字変換処理

Fig. 4　Kana-Kanji translation process for compound nouns

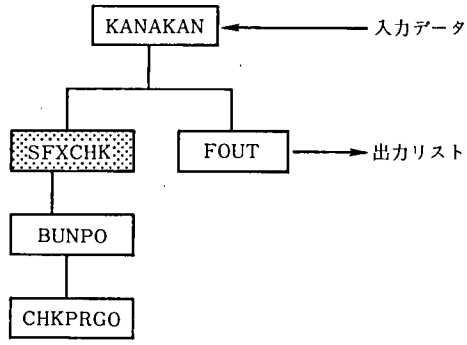


図5　かな漢字変換システムのモジール構成

Fig. 5　Module structure of Kana-Kanji translation system

```
*** THE KANAKAN PROGRAM HAS JUST STARTED. *** 10:58  03-18,'87
BSR HAS BEEN LOADED
BUNPO HAS BEEN LOADED
CHKPRGO HAS BEEN LOADED
*** INPUT SENTENCES SORTING HAS JUST ENDED *** 10:58  03-18,'87
BUNPO : 1984.12.03 VERSION
##### READ BLOCK  NO. = 0011
##### READ M-DATA NO. = 0000 ##### WRITTEN M=DATA NO. = 0002
*** THE STANDARD CONVERSION HAS JUST ENDED *** 10:58  03-18,'87

&&&&&&&&&&&&&&&&&&&&&&&& STANDARD KANAKAN CONVERSION &&&&&&&&&&&&&&&&&&&&&&&&&&&&
彼女はごうかくしゃめいを*書き込んだ。
【カキコ】  掻き込

---------------------- SUFFIX CHECK RTN START --------------------------------
KEYIN DEBUG PARAMETERS OR END
ACCEPT DEBUG-QUERY 999999
DEBUG-QUERY IS CHANGED TO 999999
KEYIN DEBUG PARAMETERS OR END
ACCEPT END
KEYIN SETUBIGO SETUZOKU LEVEL 0-3 OR ?
ACCEPT 3
BUNPO : 1984.12.03 VERSION
##### READ J-DATA NO. 0004
##### READ M-DATA NO. 0019
##### READ S-DATA NO. 0147
##### WRITTEN M-DATA NO. = 0014
PLEASE TYPE-IN NEXT COM
LDEV L1,(FORM,KANA),(ASAVE)
C DC/RES-FL TO LP
PRINT
*** THE SPECIAL CONVERSION HAS JUST ENDED ***
  1 - ｶﾉｼﾞﾖﾊ              6 SN-*  3 KS-1 KP-05000000 KM-  KB-  1 UB- 5
  2 - ｺﾞｳｶｸｼﾔﾒｲ#          10 SN-*  1 KS-1 KP-07000000 KM-  KB-  3 UB- 3
  3 - ｶｷｺﾝﾀﾞ               6 SN-+    KS-  KP-          KM-  KB-  2 UB- 3
  4 - |                    1 SN-     KS-  KP-          KM-| KB-    UB-
KEYIN DEBUG PARAMETERS OR END
ACCEPT END

&&&&&&&&&&&&&&&&&&&&&&&& SPECIAL KANAKAN CONVERSION &&&&&&&&&&&&&&&&&&&&&&&&&&&&
彼女は*合格車銘を*書き込んだ。

【ゴウカクシヤメイ】   合格|車銘，車命，車名，者銘，者命，者名，紗銘，紗命，
                        紗名，社銘，社命，社名，舎銘，舎命，舎名，砂銘，
                        砂命，砂名
【カキコ】             掻き込

*** THE KANAKAN PROGRAM HAS JUST ENDED. *** 10:59  03-18,'87
```

図6　複合名詞の意味的なつながりを考慮した，かな漢字変換システムの実行結果

Fig. 6　Execution of Kana-Kanji translation system using the suffix dictionary

5.5 かな漢字変換システムへの組み込み

複合名詞かな漢字変換ロジックのかな漢字変換システムへの組み込み，および実験は九州芸術工科大学情報処理センターで行った．使用したコンピュータは，MELCOM COSMO 700 IIIである．

開発言語はCOBOLで，図5のようなモジュール構成のうちSFXCHKルーチンが本研究のために作成した部分である．

6. 実験結果および考察

かな漢字変換システムのテストに使用した入力文は，「カノジョハ ゴウカクシャメイヲ カキコンダ．」である．実行リストは図6参照，変換結果は表4のようになった．

表4　複合名詞のかな漢字変換実行結果

Table 4　Execution results of Kana-Kanji translation for compound nouns

接続検定レベル	変換結果
0	合格者名
1	合格者名
2	合格者名，合格車名，合格社名， 合格舎名
3	合格者名，合格者銘，合格者命， 合格車名，合格車銘，合格車命， 合格社名，合格社銘，合格社命， 合格舎名，合格舎銘，合格舎命， 合格紗名，合格紗銘，合格紗命， 合格砂名，合格砂銘，合格砂命

この実行結果を検証するために，自立語辞書/接尾語辞書の「ゴウカク」，「シャ」，「メイ」の意味/品詞情報を示す(表5)．

表5　自立語辞書/接尾語辞書の意味/品詞情報

Table 5　Meaning category code and parts of speech for dictionary items

読み	漢字	意味	品詞	係り語の意味分類/品詞
ゴウカク	合格	1	サ変	
シャ	者	4		DFKSYZ1 3 4/サ変，形動
	車	E		CEXZ1 3 4/サ変，形動
	社	X		D1/サ変
	舎	X		15/サ変
	紗	A		1
	砂	7		7
メイ	名	1		BCDEOSUXZ 1 2 4 5 6 7
	銘	1		BDZ 7
	命	1		1

※□は係り語—接尾語間の関連性を示す．

「合格社名」,「合格車名」, とくに「合格舎名」は普段は使わないかもしれないが, 言っても間違いではなく, 話の内容によっては考えられる複合名詞である.

事実, このように, 複合名詞は時代により人によりあるいはその場の状況により絶えず新しく作られる性格のものであり, 従来のように自立語辞書にその都度登録していくのではきりがない.

本研究で行ったように意味的/品詞的つながりを考慮した接尾語辞書を拡充していくことが一つの解決策となる.

7. おわりに

61年度九州芸術工科大学において行った「意味的なつながりを考慮した複合名詞のかな漢字変換システムの構築」について報告した.

人工知能ブームの今日, コンピュータによる機械翻訳, 自然言語理解もつい目と鼻の先のように思われがちである. しかし, 従来の研究は限られた世界の中で, したがって使用される単語も限られた中でいかに構文・意味解析をするかに重点が置かれてきた. ところが, われわれが日常使用するような自然言語そのものを入力文とする場合には, 本稿で対象としたような複合名詞は自立語辞書に登録されておらず, 構文解析もままならないことになる.

辞書作りは地味な作業ではあるが, 今後機械翻訳, 自然言語理解システムを地についたものにするためにも辞書の研究は続けられなければならない.

本研究について御鞭達を賜わった九州工業大学吉田将教授に深甚の謝意を表する.

参考文献

[1] 稲永・小西, "かな文字文のための機械辞書の構成について", 電子通信学会技報 AL 76-39, 1976.
[2] 村松明編, 日本文法大辞典, 明治書院, 1971.
[3] 稲永・吉田, "日本語処理のための機械辞書", 情報処理 23-2, 1986.
[4] 稲永・橋本, "漢字の意味を利用した係り受け辞書によるカナ漢字変換システムについて", 日本ユニバック技報, No. 10, 1986.
[5] 稲永・新谷, "意味的なつながりを考慮した接尾語辞書の作成について", 情報処理 NL-57-3, 1986.
[6] 新谷, "意味的なつながりを考慮した接尾語辞書の作成", 日本ユニバック技報, No. 13, 1987.

執筆者紹介 稲永紘之(Hiroyuki Inanaga)

1965年九州大学工学部電子工学科卒業. 1970年同大学工学研究科博士課程退学. 同年九州芸術工科大学講師となる. 1967年より, かな漢字変換を中心とする日本語文の機械処理に関する研究に従事. 1975年電子通信学会米沢賞受賞. 電子通信学会および情報処理学会会員.

新　谷　隆　之(Takayuki Shintani)
1974年大阪大学工学部産業機械工学科卒業，同年日本ユニシス(株)入社，CMS/CCR，HIPOS，EXEC，日本語処理の保守・開発に従事．現在ソフトウェア統括部ソフトウェア三部 LINC 1100 ソフトウェア課所属，情報処理学会会員．

東京電力(株)における高速日本語印書装置による JANコードの印書

JAN Code Printing by the High-Speed Non-Impact (Japanese-language) Printer in TEPCO

浅井伸之, 長谷明拓

要 約 任意のデータをJANコードに変換し, 大量に印書することが必要となり, 高速日本語印書装置によるJANコードの印書を検討した. その結果, 各倍率のJANコードが24ドット(10 cpi), 20ドット(12 cpi)および16ドット(15 cpi)の組合せで表現でき, 外字文字として取扱えることが判明した.

そのうち, 24ドット(10 cpi)のみで表現できるJANコード(倍率1.1倍)を高速日本語印書装置で印書し, バーコード・リーダによる読取りテストを行った. 結果は, 95%以上の初回読取り率を達成し, 実用に耐える品質であることが確認できた.

Abstract There was a need to print a great deal of JAN code converted from variable data, and JAN code printing by the high-speed non-impact (Japanese-language) printer has been conducted to ascertain if the printed JAN code is good enough in quality for practical application.

As a result, it has turned out to be possible to produce several different sizes of JAN code that are printed in 24(10 cpi), 20(12 cpi) and 16(15 cpi) dots respectively through the use of special fonts externally available.

Out of those, the 24-dot pattern of JAN code patterns printed by the Japanese-language printer were fed through a bar-code reader to determine the quality of their readability.

The result showed that more than 95% of the printed JAN code was successfully read at the first reading, providing the quality which well suits practical use.

1. はじめに

JAN(Japanese Article Number)コードは, スーパーや小売店などで販売される商品のパッケージに印刷されているバーコードである. このバーコードを使用した代表的システムがPOS(Point Of Sales: 販売時点情報管理)システムであり, その普及は著しい. JAN型POSシステムは, 昭和62年1月末時点で10,866店舗に38,560台のJAN型POSターミナルが稼働しており, 半年で1,697店舗, ターミナル台数で5,762台の増加となっている[1]. また, 大手のスーパーの導入予定や, 小型店での導入が順調であり, 今後さらに増加する見込みである.

このような状況下で東京電力が, すでに設置, 使用されているPOSシステムの新たな利用方法を考案した. これは, コンビニエンス・ストアであるSストアに業務委託し, 夜間や休祭日でも電気料金の収納を行えるようにしようというものであり, 実現に際しては, 月に数百万件のデータをJANコードに変換し, 料金振込票に印書する必要があった. こうして高速日本語印書装置による「JANコード印書システム」の検討が開始され, 東京電力と日本ユニシスとの共同でこれを開発した. 現在では, 実用化

され順調に稼働している．

本稿は，高速日本語印書装置にて任意のデータを効率良く JAN コードに変換し，印書する方法，および印書した JAN コードの品質確認について報告する．

2. JAN コード

JAN コードは，数字(0～9)のみを表示し，桁数は固定長で 13 桁を表示する標準バージョンと 8 桁を表示する短縮バージョンの 2 種類がある．

JAN コードは POS システムで利用することを目的とし，共通商品コード（単品識別）の表示用として世界的に統一規格化されている．その寸法・精度などは JIS 規格(X 0501)で規定されている．

共通商品コードがメーカー段階で印刷，表示されることをソースマーキング(Source Marking)と呼び，そのコード番号は(財)流通システム開発センター・流通コードセンターが管理している．最初の 2 桁は国コード（日本は 49）で，その後に共通商品コードの 10 桁(上位 5 桁は商品メーカーコード，下位 5 桁は商品アイテムコード)と 1 桁のモジュラチェックキャラクタが続き，13 桁となっている．短縮バーコードでは共通商品コードが 5 桁となる．

JAN コードは，前述の共通商品コードの適用に混乱を起こさない限り，各店舗で機器を使用して値付ける（コード番号を決める）インストアマーキング(In store Marking)等，他の用途のために使用可能である．

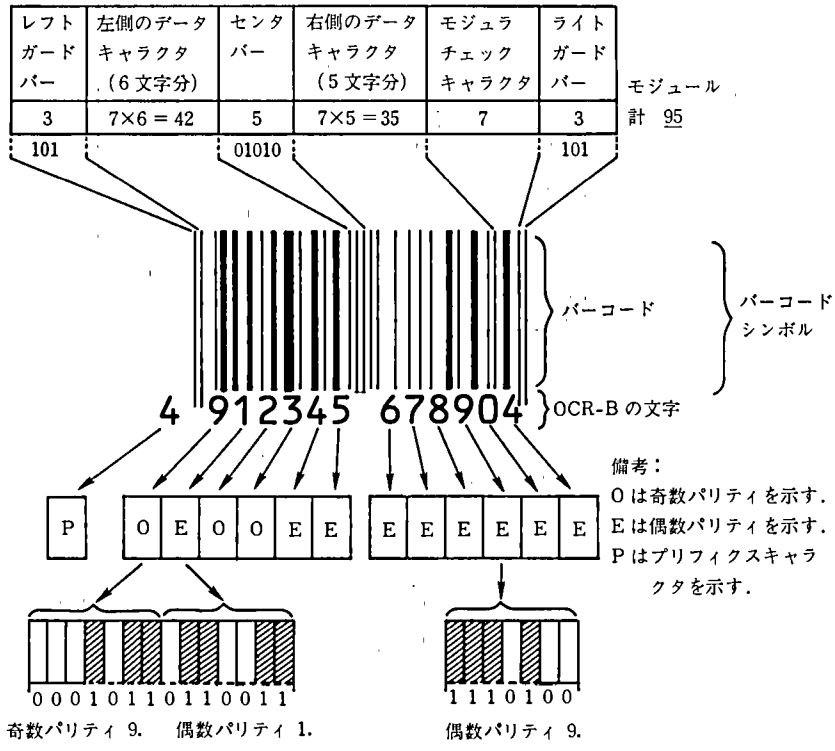


図 1　バーコードシンボルの構成

Fig. 1　Bar code symbol structure

JAN コードでは黒バー，白バーおよびマージン（余白）を構成する基本単位をモジュールと呼ぶ．標準バージョンは，このモジュールが 95 個で構成(マージン部分は含まない)される．この 95 モジュールは，図 1 に示すように左から 3 モジュールのレフトガードバー，7 モジュール構成の左側のデータキャラクタ 6 文字分(7×6＝42)，5 モジュールのセンターバー，7 モジュール構成の右側のデータキャラクタ 5 文字分(7×5＝35)，7 モジュールのモジュラチェックキャラクタ，および 3 モジュールのライトガードバーで構成される．

左側のデータキャラクタ，右側のデータキャラクタおよびモジュラチェックキャラクタは，1 から 4 モジュール幅の黒バー 2 本と 1 から 4 モジュール幅の白バー 2 本とで合計 7 モジュールになるように構成される．表現される数字(0～9)は，各数字ごとに 3 種類のバーコード・パターンがあり，桁位置により使用されるパターンがきまる．この 3 種類のバーコード・パターンは，表 1 の通りである．

標準バージョンは，バーコード自体では 12 桁分しかなく，もう 1 桁分は，左側のデータキャラクタ（6 文字分）の奇数パリティと，偶数パリティとの組合わせで表現する．これをプリフィクス（フラッグ）キャラクタと呼び，第 1 桁目とする（表 2）．

表 1　キャラクタのモジュール構成

Table 1　Encodation for JAN characters

	左側のデータキャラクタ		右側のデータキャラクタおよびモジュラチェックキャラクタ
10進数	奇数パリティ	偶数パリティ	
0	0 0 0 1 1 0 1	0 1 0 0 1 1 1	1 1 1 0 0 1 0
1	0 0 1 1 0 0 1	0 1 1 0 0 1 1	1 1 0 0 1 1 0
2	0 0 1 0 0 1 1	0 0 1 1 0 1 1	1 1 0 1 1 0 0
3	0 1 1 1 1 0 1	0 1 0 0 0 0 1	1 0 0 0 0 1 0
4	0 1 0 0 0 1 1	0 0 1 1 1 0 1	1 0 1 1 1 0 0
5	0 1 1 0 0 0 1	0 1 1 1 0 0 1	1 0 0 1 1 1 0
6	0 1 0 1 1 1 1	0 0 0 0 1 0 1	1 0 1 0 0 0 0
7	0 1 1 1 0 1 1	0 0 1 0 0 0 1	1 0 0 0 1 0 0
8	0 1 1 0 1 1 1	0 0 0 1 0 0 1	1 0 0 1 0 0 0
9	0 0 0 1 0 1 1	0 0 1 0 1 1 1	1 1 1 0 1 0 0

表 2　プリフィクス（フラッグ）キャラクタの表現

Table 2　Construction prefix (flag) character

プリフィクスキャラクタ	左側のデータキャラクタの組合わせ
0	O O O O O O
1	O O E O E E
2	O O E E O E
3	O O E E E O
4	O E O O E E
5	O E E O O E
6	O E E E O O
7	O E O E O E
8	O E O E E O
9	O E E O E O

備考：Oは奇数パリティを示す．
Eは偶数パリティを示す．
左側のデータキャラクタの組合わせは全て奇数パリティであるか、奇数パリティと偶数パリティとが3対3の割合いである．

3. 日本語印書装置での JAN コード印書方法

日本語印書装置で JAN コードを印書する方法としては，外字印書による方法，様式情報（オーバーレイ）による方法，グラフ図形印書による方法，およびイメージ印書による方法の四つが考えられる．

外字印書による方法は，JAN コードの各キャラクタに対応するバーコード・パターンを各々一つの文字として作成し，あらかじめ印書装置のフォント・メモリへ文字登録しておく．そして，JAN コードのデータを1桁または2桁の文字データとして取扱い印書する．印書データが少なく，効率よく印書できる．この方法を使用するためには，バーコード・パターンが文字として作成できなければならない．

様式情報（オーバーレイ）による方法は，ホスト側で JAN コードを，一度，バーコードのドット・パターンに展開した後，様式情報として変換しデジタル・オーバーレイとして印書する．様式情報は，ドット・パターンのいくつかに分割したビット化データと分割単位のコード化データとで構成される．この方式では，任意の JAN コードを印書するには，毎ページごとに転送するデータ量が大きくなり，データ転送時間がかかりすぎる．その結果，印書速度が著しく低下（1ページごとに印書が停止）することが考えられる．

グラフ図形印書による方法はバーコードをグラフ図形と見なし，JAN コードをグラフ図形命令に変換し印書する．この方式は，印書装置にグラフ図形印書機構が装備されている必要がある．

イメージ印書による方法は様式情報による方法とほぼ同じであり，ドット展開したデータをそのまま，または圧縮して転送する．この方式も，印書装置にイメージ印書機構が装備されている必要がある．

グラフ図形印書およびイメージ印書による方法は，いずれも外字印書による方法に比べ転送するデータ量が多くなるため，印書装置の型式によっては最大印書速度が得られない可能性がある．したがって，印書速度やハードウェア機能の制約を受けない外字印書による方法が最良であるが，使用する日本語印書装置の文字サイズでバーコードのパターンを表現する方法を見い出す必要がある．

表 3　モジュール当たりのドット数と JAN バーコードの寸法

Table 3　Size of JAN codes with each dots per module

ドット／モジュール	モジュール 寸法	モジュール 倍率	JIS許容	標準バージョン（95モジュール分）の寸法	各部を構成するためのドット数 各キャラクタ	センターバー	ガード・バー
1	0.106 mm	0.32	X	10.05 mm	7	5	3
2	0.212 mm	0.64	X	20.11 mm	14	10	6
3	0.317 mm	0.96	○	30.16 mm	21	15	9
4	0.423 mm	1.28	○	40.22 mm	28	20	12
5	0.529 mm	1.60	○	50.27 mm	35	25	15
6	0.635 mm	1.92	○	60.33 mm	42	30	18
7	0.741 mm	2.24	X	70.38 mm	49	35	21

3.1 JANコードのモジュール寸法と印書装置の文字サイズ

JANコードの基準寸法は，モジュール(黒バー，白バーおよびマージンを構成する基本単位)寸法を0.33 mm幅とした時の寸法である．JANコードの寸法は，基準寸法に対し0.8倍から2.0倍までの範囲で拡大・縮小してもよく，モジュール寸法では0.264 mm幅から0.660 mm幅の範囲となる．このモジュール寸法を日本語印書装置のドット(1/240インチ；0.106 mm)で表現するには，1モジュールが3ドット(0.317 mm)から6ドット(0.635 mm)までの範囲である．JANコードの各キャラクタ(7モジュール分)，センターバー(5モジュール分)，およびガードバー(3モジュール分)を構成するドット数はモジュール当たりのドットにより，表3の通りとなる．

一方，日本語印書装置の一般的な文字サイズをみると，英字・数字・カタカナ用(ANK用)では24ドット(10 cpi)，20ドット(12 cpi)および16ドット(15 cpi)である．また，漢字用では48ドット(12ポ)，32ドット(約9ポ)，30ドット(9ポ)および24ドット(7ポ)である．これらは，JANコードの各キャラクタの7モジュールを表現できる(7の倍数の)文字サイズではない．また，JANコードを日本語印書装置の外字方式で印書するには，7モジュールの各キャラクタと5モジュールのセンターバーを満足する二つの文字サイズ(文字幅)が必要と考えられていた[2]．このため，JANコードを印書するにはハードウェア上に特別な機能を追加する必要があるとされていた．

その例としては，

1) 漢字用文字サイズ，28ドット(約7ポ)を追加する．各キャラクタ(7モジュール分)は漢字用文字サイズで，センターバー(5モジュール分)はANK用の20ドット(12 cpi)で対応する．
2) 各文字の印字開始位置が任意に変更できる機能を装備する．文字幅の一部が重ね合わされるよう(例：32ドット幅の文字を使用し，その文字幅のうち，常に11ドット分が重なり合うようにする)1文字ずつ重ね合わせ印書を行う．それにより，文字サイズが小さくなる(例：文字幅を21ドットにする)処理をする．
3) 印書装置の水平(横)方向のドットピッチを(1/280インチ：0.091 mm)に変更し，28ドット(10 cpi)の文字を作成する．

等である．

しかし，一般的な日本語印書装置にはこのような機能がなく，外字による任意のJANコードの印書は困難とされていた．

3.2 印書装置の文字サイズに合わせるためのJANコードの分割方法

JANコードを日本語印書装置の一般的な文字サイズ(文字幅)のみ使用して表現する方法がないか検討を加えた．その結果，JANコードの分割単位をモジュール単位で再構成することにより可能であることが判明した．JANコードには，そのバーコードが示すデータの内容にかかわらず，必ず黒バー(1)または白バー(0)となるモジュールの位置がある．この位置を利用するとJANコードの各キャラクタは5から9のモジュールと見なすことができ，それらの組合せで表現することができる．

1モジュール当たり3ドット構成でJANコード(倍率：0.96)を表現すると，8モジュールは24ドット，7モジュールは21ドット，5モジュールは15ドットとなる．

```
                    1         2         3         4         5         6
Module No: 12345678901234567890123456789012345678901234567890123456789012345678
-----------:-------------------------------------------------------------------
   Module: 1010xxxxx10xxxxx10xxxxx10xxxxx10xxxxx10xxxxx1010101xxxxx01xxxxx10x
           :
Character: LGB  L1    L2     L3     L4     L5     L6     C.B    R1     R2     R3

3 Dots/Module(Mag.=0.96)
                    1         2         3         4         5         6
Module No: 12345678901234567890123456789012345678901234567890123456789012345678
-----------:-------------------------------------------------------------------
   Module: 1010xxxxx10xxxxx10xxxxx10xxxxx10xxxxx10xxxxx1010101xxxxx01xxxxx01x
           :
Character: LGB  L1    L2     L3     L4     L5     L6     C.B    R1     R2     R3
Font Size: 10-><15 ><- 10 ->< 10 ->< 12 -><15 ><- 10 -><15 ><- 10 -><15 ><-
cpi(dots):(24)  (16)  (24)   (24)   (20)   (16)   (24)   (16)   (24)   (16)

4 Dots/Module(Mag.=1.28)
                    1         2         3         4         5         6
Module No: 12345678901234567890123456789012345678901234567890123456789012345678
-----------:-------------------------------------------------------------------
   Module: 1010xxxxx10xxxxx10xxxxx10xxxxx10xxxxx10xxxxx1010101xxxxx01xxxxx01x
           :
Character: LGB  L1    L2     L3     L4     L5     L6     C.B    R1     R2     R3
(1)15 & 10 cpi
Font Size: 15><15><15><-10-><15><15><-10-><15><15><-10-><-10-><-10-><15><15><-
cpi(dots):(16)(16)(16) (24) (16)(16) (24) (16)(16) (24)  (24)  (24) (16)(16)
(2)10 & 7.5 cpi
Font Size: 10><-7.5 -><-10-><-7.5 -><-10-><-7.5 -><-10-><-10-><-10-><-7.5 -><-
cpi(dots):(24)  (32)   (24)   (32)   (24)   (32)   (24)   (24)   (24)   (32)
```

図 2　1モジュール当たり3ドットおよび4ドット構成時の文字割付け

Fig. 2　Fonts assignment for 3 and 4 dots per module

表 4　JIS 規格とモジュール数統一方式による各キャラクタ等のモジュールの構成

Table 4　Encodation for JAN characters with JIS and 7 module assignment

モジュール：xx10xxxxx10xxxxx1010101xxxxx01xxxxx01x
通常割付け：　→←キャラクタ→←キャラクタ→←センタ→←キャラクタ→←キャラクタ→←
統一割付け：　→←キャラクタ→←キャラクタ→←センタバー→←キャラクタ→←キャラクタ→←

	通常（ＪＩＳ規格）の割付け			7モジュールで統一した割付け		
	左側のデータ		右側データ	左側のデータ		右側データ
	奇数パリティ	偶数パリティ		奇数パリティ	偶数パリティ	
0	0001101	0100111	1110010	1000110	1010011	1100101
1	0011001	0110011	1100110	1001100	1011001	1001101
キ 2	0010011	0011011	1101100	1001001	1001101	1011001
キャ 3	0111101	0100001	1000010	1011110	1010000	0000101
ラ 4	0100011	0011101	1011100	1010001	1001110	0111001
ク 5	0110001	0111001	1001110	1011000	1011100	0011101
タ 6	0101111	0000101	1010000	1010111	1000010	0100001
7	0111011	0010001	1000100	1011101	1001000	0001001
8	0110111	0001001	1001000	1011011	1000100	0010001
9	0001011	0010111	1110100	1000101	1001011	1101001
センタバー	01010			1010101		
レフトガード	101			0000010		
ライトガード	101			0100000		

これらを 10 cpi (24 ドット)，12 cpi (20 ドット) および 15 cpi (16 ドット) の文字で表現した．7 モジュールと 5 モジュールとは，1 ドット分の差異がでるが，12 cpi と 15 cpi の文字を連ねることで差異をなくすことができる．JAN コードのすべてのパターンは，54 種の 10 cpi の文字と 30 種の 12 cpi の文字，および 31 種の 15 cpi の文字とを組合わせることで表現できた (図 2)．

1 モジュール当たり 4 ドット構成で JAN コード (倍率：1.28) を表現すると，6 モジュールは 24 ドット，8 モジュールは 32 ドットとなる．これらを① 10 cpi (24 ドット) と 15 cpi (16×2＝32 ドット)，②または 10 cpi と 7.5 cpi (32 ドット) の文字で表

現した．JAN コードのすべてのパターンは，①31種の 10 cpi の文字と 24 種の 15 cpi の文字との組合わせか，②33種の 10 cpi の文字と 30 種の 7.5 cpi の文字との組合わせのいずれかで可能となった．

また，センターバーを中心に7モジュールごとに分割することにより，一つの文字サイズ（7モジュール分）で JAN コードが表現できた（表4）．

3.3 0.5 ドット単位のモジュール構成

印書装置で印書した黒バーのバー幅を検査すると，必ずしもドット・ピッチに相当するバー幅が得られず，印書装置の型式により太くなるか，細くなるか，いずれかの傾向を示した（図3，4）．黒バーの太り・細り分を"a"ドットとし，$1 \geq a \geq -1$ とした場合，n ドットの実印書での黒バー幅は"$n+a$"，白バー幅は"$n-a$"である．

印書するための黒バーおよび白バーのドット数は，$a \fallingdotseq 0$ の時は，1モジュールを n ドットとし，各モジュール数(m)に n ドットを乗じて黒バーと白バーとを表現（$m \times n$ ドット）すればよく，$|a| \fallingdotseq 1$ の時は，黒バーと白バーとのいずれか一方に，1ドット分を加算し表現（$m \times n$ と $m \times n+1$ ドット）すればよい[1][3]．しかし，いずれにも a が近似しないときに，前述の方法で印書ドット数を決定すると常に a が誤差となる．

そこで，このようなときモジュールの寸法を"$n+0.5$"ドットとすることで，誤差を最少化することにした．たとえば，$a \fallingdotseq 0.5$ とすると，1モジュールの黒バーを n ドットで印書すれば $n+0.5$ ドット幅となり，1モジュールの白バーを $n+1$ ドットで印書す

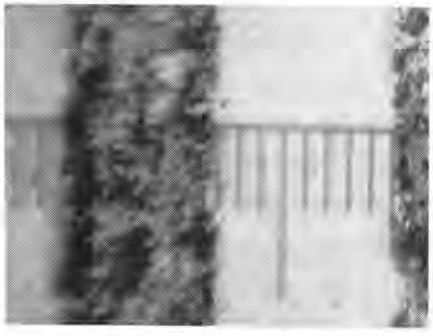


図 3　黒バーが細くなる傾向の印書の拡大写真

Fig. 3　Photograph of print sample with printer which tend to print thinly black-bar

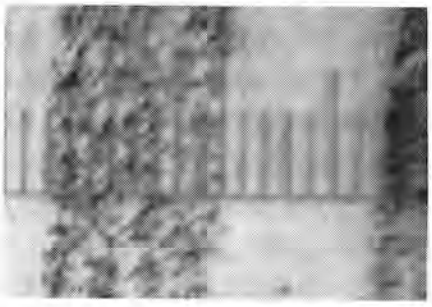


図 4　黒バーが太る傾向の印書の拡大写真

Fig. 4　Photograph of print sample with printer which tend to print thickly black-bar

表5　モジュール当たり3.5ドットおよび，4.5ドット時の印字ドットの割付け

Table 5　Dots assignment for 3.5 and 4.5 dots/module

モジュール		印字ドット数の割付け				備　考
数	幅（ドット数に換算）	太りを考慮		細りを考慮		
		黒バー	白バー	黒バー	白バー	
1	3．5	3	4	4	3	JANコード倍率：1.1倍
2	7	6or7	7or8	7or8	6or7	①ANK用 10 cpi の文字サイズは 24 ドットである。
3	10．5	10	11	11	10	
4	14	13or14	14or15	14or15	13or14	
― ― ―	― ― ― ― ― ―	― ― ―	― ― ―	― ― ―	― ― ―	
7	24．5	24 or 25				
1	4．5	4	5	5	4	JANコード倍率：1.4倍
2	9	8or9	9or10	9or10	8or9	①ANK用 15 cpi の文字サイズは 16 ドットであり、2文字分で 32 ドットである。
3	13．5	13	14	14	13	
4	18	17or18	18or19	18or19	17or18	②漢字用 7.5 cpi の文字サイズは 32 ドットである。
― ― ―	― ― ― ― ― ― ―	― ― ―	― ― ―	― ― ―	― ― ―	
7	31．5	31 or 32				

表6　24ドット（ANK用10 cpi）のみで表現したJANコードのドット・パターン（10-B）

Table 6　Example dots pattern of JAN characters for font size of 24 dots/chr

	左側のデータキャラクタ		右側のデータキャラクタ
	奇数パリティ	偶数パリティ	
0	111000000000001111110000	111000011100000011111111	111111100000001110000111
1	111000000011111110000000	111000011111110000000111	111000000111111110000111
2	111000000111000000001111	111000000011111100001111	111100001111110000000111
3	111000011111111111110000	111000011100000000000000	000000000000001110000111
4	111000011100000000001111	111000000011111111110000	000011111111110000000111
5	111000011111100000000000	111000011111111110000000	000000011111111110000111
6	111000011100001111111111	111000000000000001110000	000011100000000000000111
7	111000011111111110001111	111000000011100000000000	000000000001110000000111
8	111000011111100001111111	111000000000001110000000	000000011100000000000111
9	111000000000001110001111	111000000011100001111111	111111100001110000000111

レフトガードバー	000000000000000001110000
センタバー	111000011100001110000111
ライトガードバー	000011100000000000000000

備考：

Module	黒バー	白バー
1	3	3－4
2	6－7	7
3	10	10－11
4	13	14

れば $n+1-0.5$ ドットで，$n+0.5$ ドットの幅となる．このように誤差を小さくすることが可能となる．

1モジュールを3.5ドットとしたとき，7モジュール分（1キャラクタ分）の印字ドット数は，24.5ドットである．これは，ANK用10 cpiの文字サイズ(24ドット)にほぼ一致する．また，1モジュールを4.5ドットとした時は，31.5ドット/7モジュールで，ANK用15 cpiの文字サイズ2文字分，または漢字用7.5 cpiの文字サイズ(32ドット)にほぼ一致する（表5）．これは，一つのANK用文字サイズで外字のバーコード・パターンを作成することが可能であることを示す．

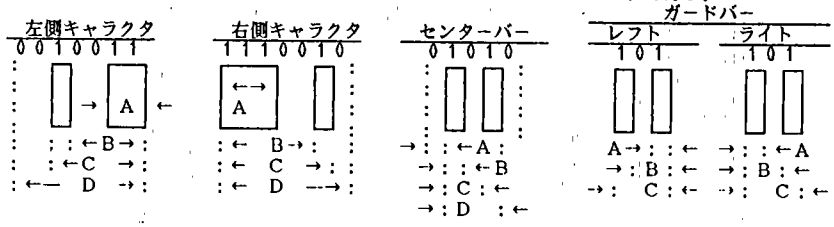
例として，表6に24ドットのみで表現したJANバーコードのドット・パターン(10-B)を示す．このパターン(10-B)は黒バーの太りを見込んだパターンであり，黒バーの構成ドット数が白バーの構成ドット数より少なくなっている．表7に1ドットを1/

表 7 JIS 規格 (1.1 倍) 寸法とドット・パターン (10-B : 24 ドット用) の寸法との差異

Table 7 Difference of size between JIS and dots pattern (24 dots/char.)

キャラクタ		JIS規格(X 0501) 倍率1．1の時の寸法 (μm)				24ドット表現の JANコード (ドット・パターン:10-B)と JIS規格との差異 (μm)			
		A	B	C	D	a	b	c	d
奇数パリティ	0	363	726	1452	2541	−46	15	−76	−1
	1	330	1089	1782	2541	−13	−31	17	−1
	2	693	1452	1782	2541	48	30	17	−1
	3	363	726	2178	2541	−46	15	−61	−1
	4	726	1815	2178	2541	15	−16	−61	−1
	5	363	1452	2179	2541	−46	30	−61	−1
	6	1452	1815	2178	2541	−76	−16	−61	−1
	7	759	1089	2201	2541	−18	−31	−61	−1
	8	1122	1452	2201	2541	−64	30	−61	−1
	9	726	1089	1452	2541	15	−31	−76	−1
偶数パリティ	0	1089	1815	2178	2541	−46	−16	−61	−1
	1	759	1452	2211	2541	−18	30	11	−1
	2	759	1089	1848	2541	−18	−31	−61	−1
	3	363	1815	2178	2541	−46	−16	−16	−1
	4	363	726	1815	2541	−46	15	−61	−1
	5	363	1089	2178	2541	−46	−31	−61	−1
	6	363	726	1089	2541	−46	15	−31	−1
	7	330	1452	1782	2541	−13	30	17	−1
	8	330	1089	1419	2541	−13	−31	−44	−1
	9	1089	1452	1815	2541	−31	30	16	−1
ガード		363	726	1089	−−	−46	15	−31	−−
センタ		363	363	726	1089	−46	−46	15	−31
JIS規格(X 0501)倍率1．1の時の許容値						±115	± 53	±115	±105

Note : ①AからDは下図の部分の寸法である。
②aからdは下図のAからDの部分を1ドット；106 μm としてパターン(10-B)より算出した値 (A'からD') から JIS規格の寸法を引いた値 (A'−A＝a) である。



240 インチ (106μm) としてこのドット・パターン(10-B)より算出した寸法と，JIS 規格(1.1 倍)寸法との差異を示す．その差異 (最大 76μm) は JIS 規格の許容値 (115μm) を満足するものであり，黒バーの太り・細りがない印書時のパターンとしても使用が可能である．

表 8 にモジュール当たり 3 ドットから 4.5 ドットで，JAN コードを表現する時に使用する文字サイズ (cpi) と必要なフォント数を示す．このように，各種の倍率 (サイズ) に対して ANK 用の文字サイズのみで表現することが可能である．

これ以外にモジュール当たり 5 ドット，5.5 ドット，および 6 ドット (倍率 1.60 から 1.92) での表現も可能である．JAN コードの倍率が大きくなると，許容誤差の寸法も大きくなり，ドット・パターンの設計が容易となる (倍率 1.60 でバー幅の許容寸法は ±0.192 mm で 2 ドット分弱相当)．しかし，JAN コードの全幅の寸法が大きくなることにより，バーコード・リーダでの使用上の制限 (リーダの読取りセンサ幅よりバーコードの全幅が大きく，センターバーより右側と左側とを分割して読取るなど) が発生する．

表 8　JANコード各サイズの印字のための必要な文字サイズとフォント数

Table 8　Font size and number of fonts to print JAN codes

ドット/モジュール	倍率	各部の構成ドット キャラクタ	センターバー	ガードバー	使用文字サイズ（cpi）の必要なフォント数 10	12	15	7.5	8	備考：
3	0.97	21	16	9	54	30	31	0	0	全てModule当り3ドット
3.5	1.10	24	18 or 17	10 or 11	30	0	3	0	0	黒バーの太り・細りによりドット・パターンの変更が可能。
					33	0	0	0	0	
4	1.28	28	20	12	31	0	24	0	0	全てModule当り4ドットで構成。
					33	0	0	30	0	
4.5	1.47	32	23 or 22	13 or 12	3	0	0	30	0	黒バーの太り・細りによりドット・パターンの変更が可能。
					0	0	66	0	0	
					0	0	0	33	0	
	1.36	30	23 or22	13 or12	0	0	0	0	33	

また，印字の上では品質の低下（太い黒バーの中央部が周辺部より薄くなるのが顕著となるなど）をまねき，実用的でなくなる．

3.4. 高速日本語印書装置用パターンの決定

高速日本語印書装置用パターン決定にさいしては，表8のモジュール当たり，3ドット，3.5ドット，4.5ドットの3種類を検討した．

はじめに，4.5ドットのパターンを検討した．このパターンはバーコードのサイズが大きく（倍率1.47），印字精度に余裕があるが，次の理由により採用に至らなかった．

1) 使用予定のバーコードリーダでは，バーコードの大きさが倍率1.3以上となると2分割読取りとなる．
2) 帳票設計上，バーコードが大きすぎる（倍率1.0以下）．
3) バーの幅が広くなり（最大19ドット分），黒ぬけが発生しやすく，印字が不安定となりやすい．

次に，3ドット（倍率0.96）と3.5ドット（倍率1.1）のパターンを比較・検討した．その結果，バーコードの大きさは倍率1.0以下ではないが，次の理由により3.5ドットのパターンを採用することにした．

1) 作成するパターンが少なくてすむ(モジュール当たり3.5ドットは33種，3ドットは116種)．
2) 文字サイズの種類が少なく，文字サイズの切替えが少なくてすむ（モジュール当たり3.5ドットの文字サイズは1または2種類，3ドットの文字サイズは3種類)．
3) 実際の印字にて黒バーの太り・細り等の発生やバーコードリーダとの相性による問題に対し，パターンを変更できる余地がある．

3.5ドット（倍率1.1）のパターンは，①バーコードの黒バーを細めに印字するパターンと，②黒バーを太めに印字するパターンの2種類を作成し，バーコードリーダに

よる読取りテストの結果，前者①のパターンを採用した．

また，3.5ドットのパターンを表現するには，①10 CPI（キャラクタ部）と15 CPI（センタバー部）とを用いる方法と，②10 CPIのみによる方法とがあるが，バーコード直下に印字する数字との位置関係が適切となる前者の方法を用いた．

4. 高速日本語印書装置によるJANコード印書の品質

高速日本語印書装置にてJANコード（倍率1.1）を印書し，バーコードの品質検査のため次のテストを行った．

1) バーコード・リーダでの読取りテスト
2) バーコード検証機によるテスト
3) 長期間印書のテスト

その結果，印字濃度制御および現像剤の使用状態を管理することで，市販のバーコード・リーダにて読取るのに十分な品質が得られたことが確認できた．各テスト結果の概要を次に述べる．

なお，品質確認の作業用に，次のテスト用印書パターンを作成し使用した．

1) バーパターン・チェック用（付録1参照）……バーパターンおよびバーコード・リーダの特性を検査するために50種類のバーコードを作成した．これらのバーコードには黒バーを細めに印書するパターンと，太めに印書するパターンとの2種類のパターンがある．
2) 品質管理・確認用（付録2参照）……稼働中の印書装置の状態を定期的に確認テストするために1ページに8種のバーコードと黒角とを配置し，バーコードの読取りと黒角による印字濃度と黒ぬけの状況が確認できるように作成した．
3) 業務テスト用パターン（付録3参照）

4.1 バーコード・リーダでの読取りテスト

バーコード・リーダの読取りテストは，使用予定の3機種により行った．

バーコードのテストパターンには前項1)2)3)のパターンの印書サンプルを用いてテストを行い，問題がないことを確認した．

読取りテストの結果は以下の通りである（付録4，5参照）．

1) バーコード・リーダの機種によって読取りやすいパターンがある．
2) 印書を濃くするほど読取り率は高くなる．
3) 読取り率はバーコード・リーダの機種やスキャナの当て方により影響される（スキャナを傾斜させると読みやすい傾向を示した）．

4.2 バーコード検証機によるテスト

バーコード検証機としてCodaScan 3600を使用した．この検証機は読取ったJANコードを自動的に解読し，測定・検証を行い[4]その結果を出力する（図5）．

高速日本語印書装置で印書したバーコードのサンプルをバーコード検証機にて読取らせ，JIS規格との差異確認をした．以下にその結果を示す．

1) JIS規格内となる率は印書濃度に影響する．印書濃度が濃いほど規格内となる率が大きくなる．
2) JIS規格に合致する（許容誤差範囲内となる）率は，印書濃度が濃いとキャラク

```
DIM IN MICROMETERS
UPC-A
NS CODE     6
MFG CODE    56230
COMM CODE   55555 3
MOD CHECK   3
MAG         110
TOLERANCE+-116
AVG A       -025
LIGHT       81 %
DARK         2 %
PCS         0.98
MIN PCS     0.65

   RJS CODASCAN
31-Jul-87  3:23 PM

     A    B    C    D
 G-017 010 000
 6-083-045-058-015
 5 000 045-030 022
 6-060 000-045-017
 2 038 043 012 000
 3-007 025-048 010
 0-022 010-048-002
 G 033-027 015-022
 G 053
 5-033-005-048 000
 5-043-027-063-005
 5-038-020-040 000
 5-035-020-055 000
 5-022-007-033 000
 3-005-015-022 000
 G-007 010 005
SYMBOL IN SPEC
```

```
DIM IN MICROMETERS
UPC-A
NS CODE     7
MFG CODE    77777
COMM CODE   22222 4
MOD CHECK   4
MAG         109
TOLERANCE+-114
AVG A       -065
LIGHT       74 %
DARK         3 %
PCS         0.96
MIN PCS     0.69

   RJS CODASCAN
31-Jul-87  3:26 PM

     A    B    C    D
 G-015 015-005
*7-121-027-086-002
 7-104 000-106 010
*7-124-005-078 012
*7-121 000-109 012
*7-139-020-119 000
*7-132-027-093 025
 G 066-035 002-012
 G 060
 2-015 096-017 020
 2-005 083-050 035
 2-043 071-063 000
 2-007 099-033 020
 2 000 078-010 007
 4-025 022 000 007
 G-005 022 020
SYMBOL OUT OF SPEC
```

CodaScan3600の出力の見方

①NS CODE, MFG CODE, COMM CODE, MOD CHECK:
：これらは読取ったバーコードのデータを示す。

②MAG　：バーコードの倍率を示す。(%)

③TOLERANCE　：規格によるバー/スペースの許容値

④AVG A　：バー幅(A)の平均偏差

⑤LIGHT　：下地(スペース)の反射率

⑥DARK　：バーの反射率

⑦PCS　：プリントコントラストシグナル；下地とバーとのコントラストを表す量。

⑧MIN PCS　：規格による最小のPCS値

⑨A, B, C, D　：下図の部分における規格値(幅)からの偏差である。規格値よりも細い場合は"--"で表示され、許容値を越えた部分は"*"と下線で表示される。

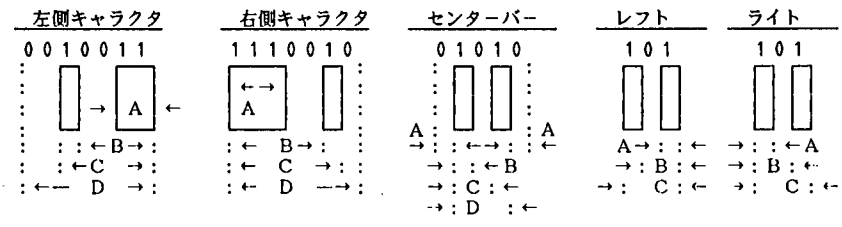


図 5　バーコード検証機 (Coda Scan 3600) の検査結果

Fig. 5　Sample of the result of check with Coda Scan 3600

タ当たり 95 %以上である.

3) JIS 規格外となりやすいキャラクタと,バーコード・リーダで読取りにくいキャラクタとは必ずしも一致していない.

4.3 長期間印書のテスト

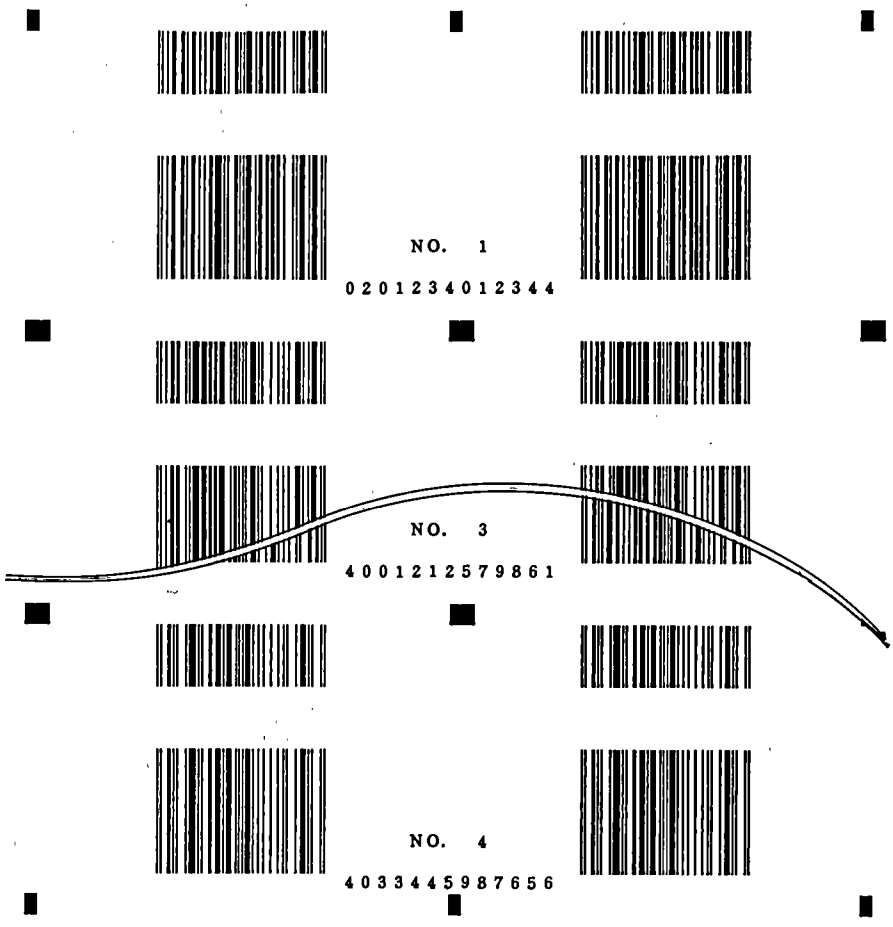
長期間印書による印書の品質変化と印書機器の違いによる品質の差を確認するため,3台の機器で5万ページごとに印字サンプルを採取し,その印書濃度を確認した.その結果,各機器に差は見られなかった.印書の品質を大きく変えるのは現像剤の劣化による印書濃度の変化である.現像剤の劣化が進むと印字が薄くなり,JAN コード印書には適さなくなるので現像剤の使用状態を管理しておく必要がある(付録6参照).

5. おわりに

高速日本語印書装置での JAN コード印書は満足できる結果が得られ,実運用が開始されている.この結果は近年のバーコード・リーダの性能向上におうところも大きく,すべてのバーコード・リーダにて同等の結果が得られるものではないが,今回採用したモジュール当たり,3.5ドット(24ドット用)のパターンは各種の印書装置に応用ができるものと考える.その試行として,ワイヤドット・プリンタに該パターンを外字登録し印字した結果,十分実用性のある印字であった(付録7参照).

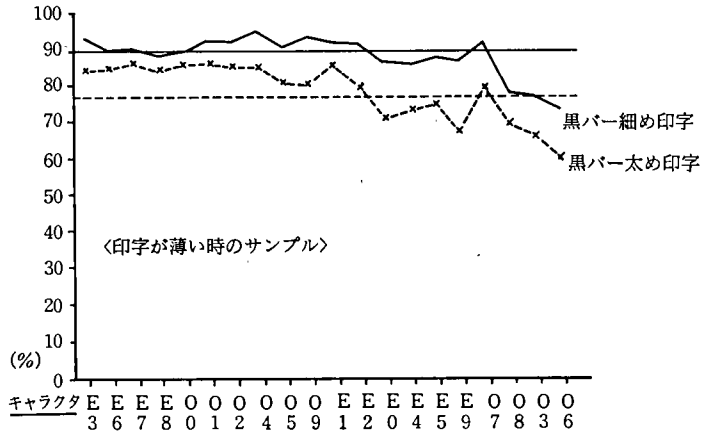
今後も,各種の印書装置で該パターンによる JAN コード印書を行い,品質確認する予定である.

付録 1 バーパターン・チェック用

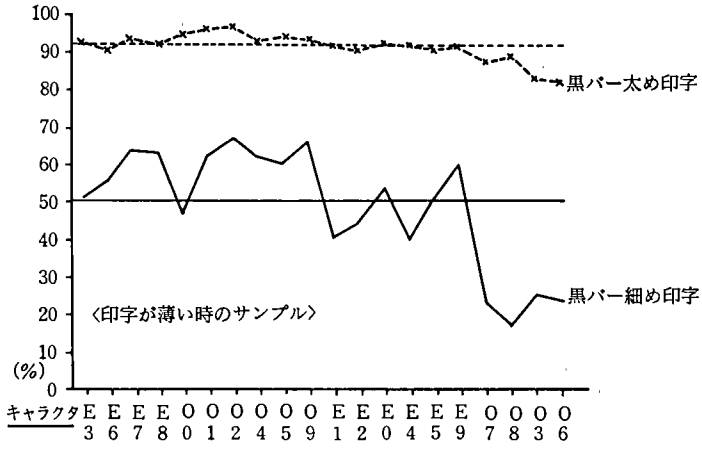


付録 2　品質管理・確認用

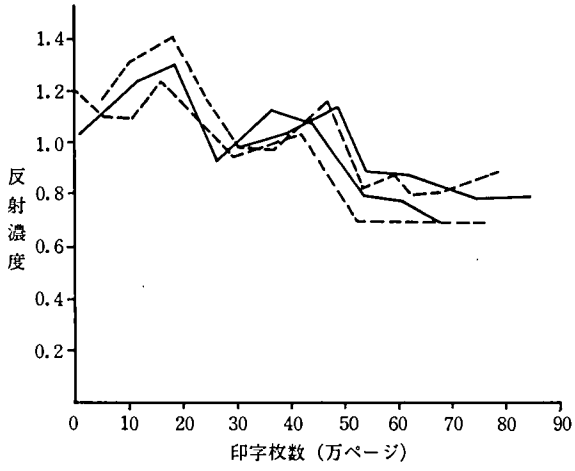
付録 3　業務テスト用パターン



付録 4　バーコードリーダ#1, #2での読取結果（直角）例



付録 5　バーコードリーダ#3での読取結果（傾斜）例



付録 6　現像剤の劣化による印字濃度の推移（小黒角濃度）

(a)

```
DIM IN MICROMETERS
UPC-A
NS CODE     6
MFG CODE    78901
COMM CODE   56789 4
MOD CHECK   4
MAG         148
TOLERANCE+-175
AVG A       -015
LIGHT       75 %
DARK        11 %
PCS         0.85
MIN PCS     0.69

   RJS CODASCAN
28-Apr-88 11:20 AM

    A   B   C   D
 G-086-010-038
 6-045 000-086 017
 7-015-015-060 030
 8-033 000-099-045
 9 060 000-066-030
 0-020-020-088-027
 1 010-063 035-005
 G 050-017 000 000
 G 030
 5-043-055-038-025
 6-007 007 007-007
 7 005-002-002-063
 8 010-040-043-025
 9-002 005 000-012
 4-027 000 020 015
 G-015 000-005
SYMBOL IN SPEC
```

(b)

```
DIM IN MICROMETERS
UPC-A
NS CODE     6
MFG CODE    78901
COMM CODE   56789 4
MOD CHECK   4
MAG         147
TOLERANCE+-172
AVG A       070
LIGHT       75 %
DARK         4 %
PCS         0.95
MIN PCS     0.69

   RJS CODASCAN
28-Apr-88 11:22 AM

    A   B   C   D
 G 050 043 109
 6 030 000 048 015
 7 045-081-015-027
 8 055 027 015 015
 9 101-048 000 000
 0 038 005 000-010
 1 093-035 088-043
 G-020 053 002 088
 G-055
 5 053-015 068 015
 6 053 000 086 000
 7 104 040 121-030
 8 127 000 055 033
 9 078 012 091-033
 4 088 053 157 063
 G 040 000 066
SYMBOL IN SPEC
```

(a)

```
DIM IN MICROMETERS
UPC-A
NS CODE     0
MFG CODE    12345
COMM CODE   01234 1
MOD CHECK   1
MAG         147
TOLERANCE+-172
AVG A       003
LIGHT       75 %
DARK        14 %
PCS         0.81
MIN PCS     0.69

   RJS CODASCAN
28-Apr-88 11:23 AM

    A   B   C   D
 G 012 007-007
 0 000 022-027-005
 1 022-063 027-043
 2 010 033 000-007
 3 000 045 022 071
 4 040-015-048-020
 5-012 025-134-030
 G 022-005 005 002
 G 020
 0 007-027-020 012
 1 005 030-091-022
 2 000 109-015 048
 3-027 000-045-002
 4-010 000 000 002
 1-002 045-083 002
 G-048-010-015
SYMBOL IN SPEC
```

(b)

```
DIM IN MICROMETERS
UPC-A
NS CODE     0
MFG CODE    12345
COMM CODE   01234 1
MOD CHECK   1
MAG         146
TOLERANCE+-172
AVG A       083
LIGHT       76 %
DARK         3 %
PCS         0.96
MIN PCS     0.68

   RJS CODASCAN
28-Apr-88 11:21 AM

    A   B   C   D
 G 093 025 129
 0 081-005 055 000
 1 066-040 109-012
 2 134 010 129 005
 3 055 017 063 038
 4 119 000 045 012
 5 076 040 053 012
 G-025 040 022 129
 G-076
 0 093 007 053 017
 1 091 043 010 027
 2 104 106 053 012
 3 022-015 000 010
 4 027 027 093 000
 1 124 081 066 025
 G 091 073 124
SYMBOL IN SPEC
```

(a)：プリントリボンの寿命直前（印字が薄い）

(b)：プリントリボンが新しい時（印字が濃い）

付録 7　ワイヤー・ドットプリンタによる印書例および検証機による検査結果

参考文献

［1］ 事務機情報新聞社, 第 9 回 JAN・POS 導入実態調査, 1982.3.10.

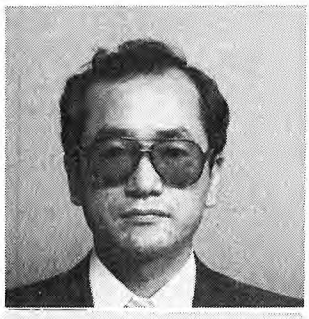
［2］ 森　宗正, “各種のバーコードのパーソナル・コンピュータによる印書実験”, 技報 No. 15, 1987, pp.108～123.

［3］ 森　宗正, 漢字プリンタ用バーコードパターンの理論的考察, テクニカルシンポジウム, 82 (S-7308), 1982.

［4］ シグノード株式会社, コーダースキャン 3600 マニュアル.

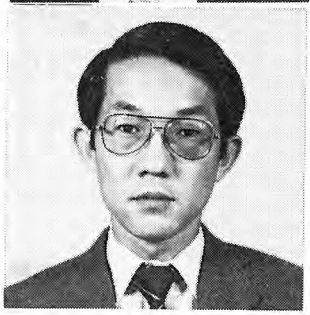
執筆者紹介　浅　井　伸　之 (Nobuyuki Asai)

昭和 47 年富山県立砺波工業高等学校電子科卒業. 同年東京電力(株)入社. 現在情報システム部システム技術課に所属. 主として, OS など基本ソフトウェアの管理業務に従事.



長　谷　明　拓 (Akihira Hase)

昭和 45 年都立本所工業高等学校電気科卒業, 同年日本ユニシス(株)入社. 現在ハードウェアプロダクト四部ペリフェラル一課に所属. 主として, 印書装置導入時の評価と設置後のリエゾン業務に従事.



知識工学ワークステーション KS-301 の 日本語 Common Lisp 環境(NCL)

Using Japanese Character in Common Lisp; Nippongo Common Lisp Environment on Knowledge Engineering Workstation

大 田 一 久

要 約 知識工学ワークステーション KS-301 で，日本語の使用を可能にすべく日本語 Common Lisp 環境(NCL)を開発した．KS-301 は，Common Lisp が稼働する Lisp 専用マシンである．NCL では Common Lisp で日本語文字が使用できるのみならず，KS-301 のプログラミング環境全体を通して日本語文字の使用を可能にした．

Common Lisp で日本語文字を英数字とできる限り同等に取り扱おうとする方法はすでに提案されている．これは，文字コードの値の上限を大きくすることによって日本語文字を表現し，また文字列の表現を2種類用意することにより日本語文字を含まない場合に記憶領域の消費が増大することを避けようとしている．

筆者らの実現法もほぼこれと同等であり，日本語文字を Common Lisp の一つの文字型オブジェクトとして扱い，文字列，シンボルにも日本語文字を使用できる．日本語文字の文字集合に含まれる英数字などの取り扱いにも考慮した．ウィンドウ・システム，ファイル・システム，ネットワークでも一部を除いて日本語の使用を可能にした．

Abstract Japanese language capability of computer systems have great significance in Japan. It is much more important in case of knowledge engineering products in which the user interface play important roles.

We have built such a capability in the Common Lisp language and the programming environment on the Lisp machine KS-301. This facility is called Nippongo Common Lisp Environment (NCL).

This paper describes the basic idea of the implementation and features added to the Common Lisp language. And also mentions the input method of Japanese characters and topics related to Lisp machine specific features.

1. はじめに

日本ユニシスの知識システム・ワークステーション KS-301 は，米国 TI 社の Lisp 専用マシン Explorer* を OEM 販売しているものである[1][2]．当然のことながら，当初は日本語に対する考慮はまったくなされていなかった．このワークステーションは Common Lisp を完全に包含する TI Common Lisp が稼働し，エキスパート・システム構築ツール KEE** を使用することができる．筆者らは KEE およびそのアプリケーションを含めて日本語の使用を可能にすべく，日本語 Common Lisp 環境(NCL)の開発を行ってきた．NCL では Common Lisp で日本語が使用できるのみならず，KS-301 のプログラミング環境全体を通して日本語の使用を可能にした．

* Explorer は米国 Texas Instrument Incorporated 社の商標である．
** KEE は米国 IntelliCorp 社の商標である．

プログラミング言語での日本語の使用について，日本ユニシスでは以下のようなレベルを考えている[3]．

レベル1：日本語データの定義と処理
レベル2：日本語リテラルが使用できる．
レベル3：コメントに日本語が使用できる．
レベル4：日本語でメッセージが出力される．
レベル5：ユーザの定義する名前に日本語が使用できる．
レベル6：予約語，キーワードが日本語である．

このレベル付けによると，今回の開発目標はレベル5に相当する(ただし，レベル4の日本語によるメッセージは一部を除いて今回の開発では達成されていない)．レベル1，レベル2については，Common Lispでの文字の取り扱いとの整合性をとくに考慮し，日本語データを従来の文字データの自然な拡張として使用できるようにした．

Common Lispで日本語文字を英数字とできる限り同等に取り扱おうとする方法は文献[4][5]で提案されているが，筆者らの実現法もほぼこれと同等であり，その実現方法については3章で述べる(この実現法は日本語文字に限らず，文字コードとして複数バイトを必要とする他の言語の文字セットの場合にも応用可能である)．その前にCommon Lispにおける文字，文字列の取り扱いについて2章で述べる．さらに文字セットに含まれる文字に関しての問題，およびその対処について4章で述べる．

また，KS-301のウィンドウ・システム，ファイル・システム，ネットワークでも一部を除いて日本語の使用を可能にしたが，この点については5章で述べる．6章では日本語文字の入力法について述べ，最後に各種のユーティリティおよびKEEでの日本語の使用について7章で述べる．

2. Common Lisp

Lispは，プログラミング言語としてはFORTRANに次ぐ歴史を持っている．しかし，その目的とされたのは記号処理，あるいはリスト処理といったFORTRANが苦手とする分野であった．当初は定理の自動証明システムの研究を目的として開発され，その後，数式処理，あるいは人工知能に関連した分野で主に使われていた．Lispでは言語の取扱うモデルと汎用計算機のアーキテクチャの間にギャップがあるため，また初期の頃は計算機の能力も低かったこともあり，実験的なシステムでの利用にとどまっていた．

しかし，計算機ハードウェアの進歩および処理系技術の発達に伴い，機能の拡張が行われ汎用のプログラミング言語と同等，あるいはそれ以上の機能を持つようになった．また，Lisp専用機の登場で，大規模なシステムの記述にも使われるようになった．汎用機の上でも最適化コンパイラにより，かなりのパフォーマンスが得られるようになってきた．Lispと一口に言っても長年の歴史の中で微妙に仕様の異なる多くの方言が現れてきていたが，近年，Common Lispが事実上の標準として普及しつつあり，ANSI，JISおよびISOにおいて標準化の作業が進行中である．

2.1 Common Lispのデータ型

もともとLispでは，一般のプログラミング言語の意味での型の概念は希薄であっ

た．多くのプログラミング言語では変数に対してデータ型が定義され，この型の情報に従って変数に含まれるデータの解釈が決定される．これに対してLispでは，計算機上のビット列そのものに対して型がそれぞれ与えられ，変数に対してはデータ型は定義されない．言い換えればデータの表現の一部として型の情報が含まれており，ビット列の意味はLispの処理系によって決定されるので，プログラムがそれを自分の都合で解釈することは許されない(これは実行時に型の検査が行われることを意味している)．このようにそれ自身に型などの情報まで含むデータを，とくにオブジェクトと呼ぶことがある．

Common Lispでは数多くのデータ型があらかじめ定義されており，それに加えて使用者が独自のデータ型を定義することもできる．その中から代表的なもののいくつかについて簡単に解説する．Lispでは，シンボルと呼ばれるデータが重要な役割を持っている．これはsymbolというデータ型で，名前を持ち，数値などの他のデータ型と共にリスト構造の基底をなす．リスト構造はlistというデータ型で表現され，リストそのものの入れ子を含む，数値，シンボルなどの列である．数値としては整数，浮動小数点数，有理数，複素数を扱うことができ，それぞれinteger，real，ratio，complexと呼ばれる型を持つ．これらは，すべてnumber型の副型(subtype)として定義されている．たとえばinteger型のオブジェクトはnumber型の特殊な場合で，number型の性質を持ち，それに加えてinteger型特有の性質を持っている．Common Lispでは，このようにデータ型の間に階層関係が定義されている．また，文字はcharacter型のオブジェクトで表される．

Common Lispではこれらのオブジェクトを要素とする配列を用いることができ，それはarray型のオブジェクトとなる．このとき，とくに一次元の配列をベクタと呼び，これをvector型というarray型の副型と定義する．さらに，リストとベクタに共通する一次元の列という性質を取り出してsequence型としており，一次元の列に関する一般的な操作が関数として用意されている．一方，配列には要素の型を限定したものを考えることができ，その場合の型もarray型の副型となる．たとえば要素を整数に制限した配列を用いることができる．とくに文字列は要素を文字に限定したベクタで表され，string型と呼ばれる．

実際の計算機上では，Lispのオブジェクトは，メモリ上のある大きさを占めるデータの内部表現に対する参照ポインタで表される．この参照ポインタには，メモリ上のデータの内部表現のアドレスおよびテータ型を表すタグが含まれている．Lispではこの参照ポインタが実際に変数の値となり，配列の要素となる．また，これを見るだけでオブジェクトのデータ型が判定される．

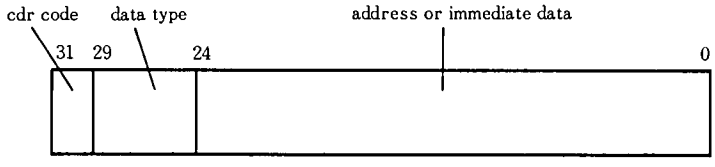


図1　参照ポインタの形式

Fig. 1　Format of object reference

KS-301では，参照ポインタは図1のような形式を持っている．list, array型などのオブジェクトはすべてこの形式をとるが，fixnum型と呼ばれる固定長整数などのようにその内部表現が十分小さい場合は，参照ポインタの代りに，データ型を示すタグとデータの内部表現そのものが用いられることもある．このような表現をとくに即値形式と呼ぶ．要素を特定のデータ型に制限した配列では要素として参照ポインタではなく，データの内部表現そのものを用いる場合がある．

Lispの一つの大きな特徴として，プログラムの表現形式を挙げることができる．すなわち，Lispのプログラムは関数定義の集まりとして構成されるが，これらの関数定義はLispのリストで表現される．関数の定義はいくつかの式の集まりであり，それぞれの式は一つのリストである．式に現れる関数名，変数名はシンボルであり，数値，文字，文字列などは定数として扱われる．また，特殊な表記を用いてリストそのものを定数とすることができる．このようなデータおよびプログラムはいくつかの例外を除いて，テキストとして目に見える形で表現することができる．この表現を印字表現と呼び，データ，プログラムの入出力のために用いられる．この印字表現の構文規則が，一般のプログラミング言語の構文規則に相当する．

2.2 Common Lispでの文字の使用

最初に述べた日本語使用のレベル5を達成するためには，文字および文字列の要素として日本語文字を使用できること，およびシンボルの名前として日本語文字を含むことが許されればよい．そこでcharacter, stringおよびsymbolの各型についてCommon Lispの仕様の概略をKS-301の場合に即して述べる．より詳しい仕様については文献[6]を参照されたい．

Common Lispではcharacter型のオブジェクトはコード，フォントおよびビットの三つの属性を持っている．使用する文字コードの体系については，とくに規定はないが標準文字としてアルファベット，数字，特殊記号および空白の95文字を含むことが要求されている．character型オブジェクトのコード属性は，その処理系で使用している文字コード体系で表される文字のコードである．フォント属性は，その文字を表示するときに使用されるフォントの種類を示す非負整数である．

character型のオブジェクトの印字表現は次のようである．

#\〈文字〉

たとえば，小文字の“a”は次のように表される．

#\a

与えられたデータがcharacter型かどうかを判定するにはcharacterpという関数を用いる．また，文字が同じものかどうかを判定するにはchar=という関数を用いる．これは比較の際にフォント属性および大文字・小文字の区別を考慮する．これに対して，大文字小文字の区別を考慮しないchar-equalという関数もある．また，文字の大小関係の判定のための関数がいくつか用意されている．さらに，文字の種類などを判定する関数として次のようなものがある．

(alpha-char-p 〈文字〉)	アルファベットかどうか
(upper-case-p 〈文字〉)	大文字か
(lower-case-p 〈文字〉)	小文字か

(digit-char-p 〈文字〉)　　数字か

大文字小文字の変換を char-upcase, char-downcase を用いて行うことができる．コード，フォントおよびビットの各属性はそれぞれ，char-code, char-font および char-bits 関数で取り出すことができる．逆に三つの属性が与えられたときに，それらを持つ文字オブジェクトを得る関数が make-char である．KS-301 では，character 型のオブジェクトの内部表現は図 2 に示すような即値形式である．すなわち，コード，フォント属性にそれぞれ 8 ビット，ビット属性に 5 ビット使用している．

string 型はベクタの特殊な場合で，要素を string-char 型と呼ばれる character 型の副型に限定した配列である．どのような文字が string-char 型となるかは処理系に任されており，一般の意味での文字列が効率よく表現できるように工夫の余地が残されている．KS-301 ではフォント，ビット属性を持たない文字オブジェクトが string-char 型であり，string 型のベクタの内部表現は各要素に 8 ビットのコード属性のみを含む．

文字列オブジェクトの印字表現は“ ”の間に要素の文字を並べたものである．たとえば，

"abcdefg"

は七つの文字を要素とする長さ 7 の文字列である．与えられた長さの文字列を作り出す関数として make-string がある．文字列中の要素を取り出すには char 関数を用いる．たとえば，x という変数の値が“abcdefg”という文字列であるとすると，

(char x 0)　　=> **#\a**

である．文字列に限らず配列の先頭の要素の添字は 0 である．文字列の長さは length 関数で得ることができる．これは文字列以外のベクタ，あるいはリストに対しても用いることができる．文字列の要素に文字を代入するためには次のようにする．

(setf (char x 0) #\@)

これにより，x に含まれる文字列の先頭の要素を#\@という文字にする．文字列の比較は string＝あるいは string-equal で行う．これらは，それぞれ char＝あるいは char-equal を用いて各要素を比較する．また，辞書式順序で文字列の大小を判定する関数がいくつか用意されている．文字列の要素である文字を大文字にしたり小文字にするための関数として，string-upcase および string-downcase がある．この他に，sequence 型のオブジェクトに対して用意されている一次元の列を操作する関数は，すべて string 型のオブジェクトに対しても用いることができる．

symbol 型のオブジェクトは名前を持ち，変数としての値，関数の定義，パッケージ

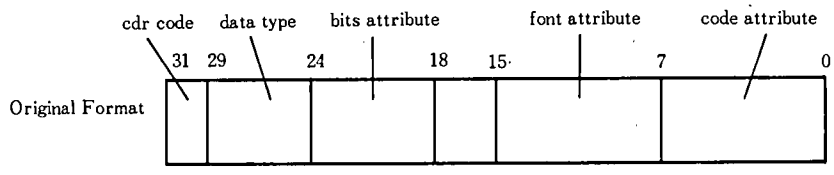


図 2　character 型の表現

Fig. 2　Representation of character type data

および属性リストを持っている．シンボルの名前は文字列であり，シンボルの印字表現として用いられる．これは symbol-name という関数で取り出すことができる．あとの属性はそれぞれ symbol-value, symbol-function, symbol-package, symbol-plist という関数でそれぞれ参照される．変数としての値，および関数の定義はプログラムの中でシンボルが変数あるいは関数として用いられたときに，これらの属性が参照される．パッケージはそのシンボルが登録されている名前表である．

Lisp では同じ名前を持つシンボルは，常に同一のオブジェクトになるように名前表に登録される．Common Lisp ではこの名前表が複数存在することができ，シンボルを指示する際に名前表を指定することができる．逆に各シンボルは自分が登録されている名前表をパッケージ属性として持っている．

シンボルの印字表現は，その名前にアルファベットなど決められた文字しか含まれていない場合，名前そのものである．すなわち，“ABC”という文字列を印字名としてもつシンボルの印字表現は，

ABC

である．より正確には，シンボルの印字表現としてそのまま用いることのできる文字および構文規則が決められており，それらの文字を構成文字と呼ぶ．それ以外の文字をシンボルの名前の一部として使用する場合は，特殊な表記を用いる必要がある．また，パッケージを印字表現の一部として指示することもできる．

3. NCL の日本語処理機能

3.1 Common Lisp での日本語文字の取扱い

本節では，Common Lisp での日本語文字の取扱いを前節で述べたもとの仕様に基づいて述べる．このとき，Common Lisp の仕様とできるだけ矛盾しないようにすることを考えた．

まず，日本語文字を character 型のオブジェクトとして使用することを考える．たとえば，次に示すように漢字を文字オブジェクトとすることができればよい．

(characterp #\漢) =>*t*

このためには，フォント，ビット属性はさておき，コード属性として日本語文字のコードが保持できるような内部表現が必要である．Common Lisp では特定の文字セットを規定していないため，日本語文字を含むような文字セットを使用することは言語仕様としてはなんら問題なく，文字コードの上限を示す定数の値を大きくすることにより，文字種の多い文字セットを Common Lisp で扱うことができる．実現法としては文字オブジェクトの内部表現のうち，コード属性のために使用する部分を拡張することにより可能となる．

次に，このような日本語文字オブジェクトがそのまま文字列の要素として使用できることが望ましい．すなわち，文字列に関して次のような結果が得られるべきである．

(char “日本語 Common Lisp 環境” 16) => #\環

(length “日本語 Common Lisp 環境”) => 18

これは，他のプログラミング言語の日本語機能で見られるように，日本語文字コードの二つのバイトをそれぞれ文字列の要素として詰めるような方法では，日本語文字

列の処理を見通しよく記述することができないからである．また，このような方法では文字列が文字の配列であるという定義と矛盾することになる．実際に文字列の要素として，日本語文字オブジェクトを使用可能にするには string-char 型に日本語文字が含まれるようにすればよい．これも，言語仕様上はとくに問題なく，実現上の問題である．

さらにシンボルの名前としての日本語文字の使用であるが，シンボルの名前は文字列であると規定されているので，文字列の要素として日本語文字が使用できればこれも問題ない．これにより，日本語の名前を持つシンボルをリスト構造の要素として使用することができ，さらにプログラムの変数名，関数名として日本語の名前を使用することができる．

```
(defun 関数 (引数)          ; 日本語の関数名，変数名
  (cons 引数 nil))
```

ただし，印字表現の構文規則が問題として残るが，これについてはあとで文字セットに含まれる文字について考えるときに取り上げる．

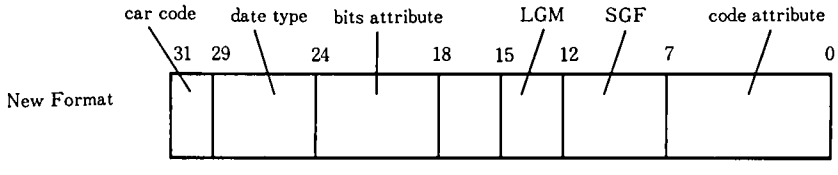
3.2 日本語文字処理機能の実現

前節で述べたような日本語文字の取扱いを KS-301 で実現する方法について簡単に解説する．

KS-301 の文字オブジェクトの内部表現では，フォント属性に 8 ビット使用しており 128 種類の異なったフォントを使用できるが，これはやや非現実的である．実際標準で提供されるフォントの種類はこれよりはるかに少ない．フォント属性が実際に使われるのはウィンドウ・システムと Zmacs エディタであるが，これらのシステムで通常同時に使用できるフォントの数の上限は 26 である．

このことを利用すると，適当な下駄を履かせることによってフォント属性を文字コードの一部として用いることができる．文字コード体系が JIS X 0202(ISO 2022)[7]に準拠しているとすると，2 バイトの文字コードを 1 バイトずつに分割すると両方とも 32 より必ず大きい値を持つ．すなわち，フォント属性が 32 より大きい場合，日本語文字の上位バイトと見なすことができる(図 3)．

このような文字を long-char 型と定義する．この形式では，long-char 型の文字はフォント属性を持つことができない．そこで，long-char 型の文字のフォント属性は常に 0 であることにする．これに対して，従来の文字コードが 1 バイトで表現できる文字を short-char 型とする．long-char 型と short-char 型は，character 型の副型であり，



LGM　Long Character Modifier
SGF　Short Character Font

図3　character 型の新しい表現

Fig. 3　New representation of character type data

共通部分を持たない．

文字列に関しては，ビット属性およびフォント属性を持たない2バイトのコードのみからなる文字をstring-char型とし，これを要素とするベクタをstring型と規定すればよい．これは，英数字と日本語文字の混在する文字列を表現することが可能である．また，シンボル印字名は文字列であるので，読み込み時の構文解析の問題を除けば日本語文字を名前に含むシンボルを使うことができる．

ところが，この方法では文字列はすべて要素当たり2バイトのベクタとなるため，日本語文字を含まない文字列については従来の2倍の記憶領域が必要になる．既存のプログラム，とくにKS-301のシステム・プログラムを考えると，用いられている文字列はシンボルの印字名を含めてまったく日本語文字を含んでいないため，この分の記憶領域は正に2倍になってしまう．この問題に対する一つの解決は，文字列の内部表現を1バイトのベクタと2バイトのベクタの2種類用意することである．すなわち，英数字のみからなる文字列は1バイトのベクタを使用し，日本語文字を含む文字列は2バイトのベクタを使用する．このとき，それぞれを現在のstring型の副型として定義する．そして，現在の文字および文字列操作を新しい型と従来のstring型の両方に適用可能な総称的(generic)な関数として定義する．

この方法では，日本語文字を含まない文字列の記憶領域の消費を現在と同じに押さえることができる．別な見方をすると，日本語文字を含まない文字列はその内部表現をある意味で最適化することができる．

KS-301では，各要素ごとにフォント属性を持つことのできる特殊な文字列が使用されている．この文字列はfat-stringと呼ばれ，ウィンドウ・システムやZmacsエディタでマルチ・フォントのテキストを表現するために用いられている．この文字列は，各要素当たり2バイトの幅を持ち，8ビットのコード属性とフォント属性を同時に格納できる．これを使用すると，long-char型の文字のコード属性を一つの要素に格納することができる．short-char型の文字は，上位バイトを0とすることで表現できるので，両方の混在する文字列を使用することができる．また，2バイト文字コードの性質から上位バイトが32より小さい値を持つ場合は，フォント属性のshort-char型であることが判定でき，従来の使用法とも矛盾を起こさない．この文字列をfat-string型と定義する．

fat-string型の文字列は，見かけ上従来の文字列とまったく同様に扱うことができ，fat-string型はstring型の副型と定義するのが自然である．これに対して従来の1バイトの文字列をthin-string型と定義する．それぞれの要素となり得る文字をfat-string-char型とthin-string-char型と定義する．fat-string型とthin-string型は共通部分を持たないが，thin-string-char型はfat-string-char型の副型になる．さらに，KS-301ではthin-string-char型はshort-char型のうちビット属性，フォント属性を持たないものと一致する．これに対して，fat-string-char型はビット属性を持たないlong-char型とshort-char型の和集合になる(図4)．

シンボルの印字名はstring型であればよいので，fat-string型の印字名を持つシンボルを作ることができる．ただし，このようなシンボルを読み込むためにはlong-char型の文字の構文属性を定義しなければならない．Common Lispでは，標準の文字セッ

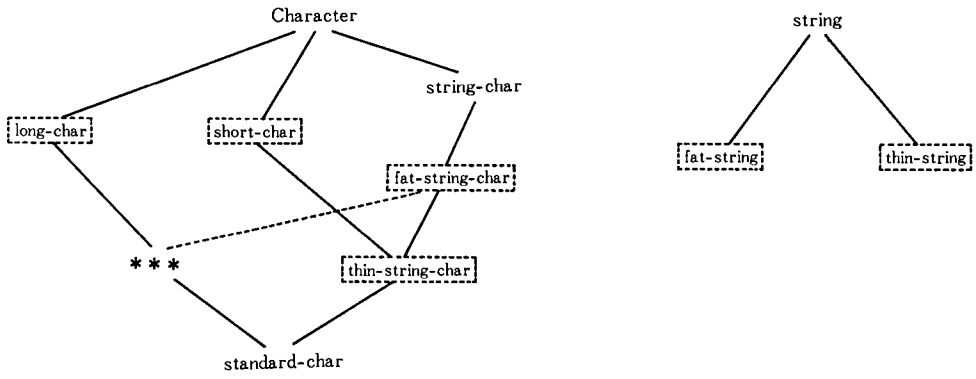


図4　character 型および string 型の階層関係

Fig. 4　Type hierarchy of type character and type string

トを string-char 型の副型である standard-char 型として定義しており，これらの文字を用いて印字表現の構文規則が定義されている．実際には各文字の構文属性がリードテーブルと呼ばれる表に登録されている．標準文字以外の文字についても構文属性を定義しなければならないので，各処理系はこの表に string-char 型の文字をすべて登録するのが普通である．しかし，日本語文字の数は数千個にのぼるため，この方法は現実的ではない．この問題に対する解決法は次章で述べる．

4. 文字セットについて

NCL では，日本語文字の文字セットとして JIS X 0208[7] を用いている．これを JIS 文字セットと呼ぶことにする．これに対して従来 KS-301 で用いられているのは，ASCII 文字セットを拡張した Lisp Machine 文字セットである．これを LISPM 文字セットと呼ぶことにする．この二つの文字セットを同時に使用しようとする場合の問題点と，それに対する対処法を述べる．

JIS 文字セットには ASCII と同じ字形をもつ文字が含まれている．たとえば，アルファベット文字は JIS と ASCII の両方に含まれている．事実，すべての ASCII 文字と同じ字形を持つ文字が JIS の中に含まれている(したがって，Common Lisp の標準文字はすべて JIS の中に含まれていることになる．このことは文字セットが JIS のみであっても，Common Lisp としては問題のないことを意味している)．したがって，JIS と ASCII を同時に使用しようとすると，これらの区別をどうするかという問題がある．

KS-301 の場合では，現在使用している LISPM 文字セットは ASCII を含んでいる．この LISPM 文字セットの ASCII を含む部分集合について，同じ字形を持つ文字が JIS に含まれている(図 5)．この部分集合と ASCI 文字セットとの間に，字形が同じであるという一対一対応を付けることができる．以下，LISPM 文字と JIS 文字の字形の同じ文字の取扱いについて述べるが，ASCII 文字と JIS 文字の場合にも同様の議論が成立する．

構文に関わる場合は，JIS 文字と LISPM 文字の区別は混乱を招く．たとえば，JIS

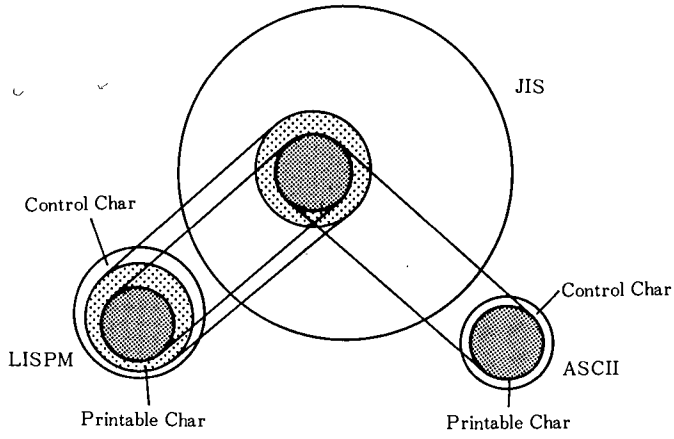


図5　文字セット間の対応関係
Fig. 5　Corespondence among character sets

の括弧は構文上括弧と見なされないとすると，プログラムのテキスト上では括弧の対応はとれているように見えても，実際には読み込めないといったことが起こる．通常，LISPM と JIS のアルファベットは大きさの異なる字体で印書されるが，複数のフォントを用いて印書が行われるような場合には，実際の字体から LISPM か JIS かを判断するのはかなり困難である．また，数字の場合，LISPM であれば数値になるが JIS ではシンボルになるというのでは混乱を招くであろう．したがって，LISPM 文字とそれに対応する JIS 文字は同じ構文属性を持つことが望ましい．

シンボルの場合も，まったく同一の綴の名前を LISPM と JIS の二つの場合を考えることができる．この二つが異なったシンボルになるとすると，定義したはずの関数が未定義であるといったことが起こる．したがって，JIS で書かれた LAMBDA も LISPM で書かれた **LAMBDA** も同一のシンボルになってほしい．Common Lisp ではシンボルの印字名は常に大文字に変換されるが，同様に JIS 文字を対応する LISPM 文字に変換してやることが考えられる．

しかし，この変換は元の区別を復元することが不可能であり，読み込みの際に無条件に変換を行ってしまうと，外部から入ってきたデータが保存されないことになる．これは Lisp 以外の既存のアプリケーションとデータを交換しようとするときの障害となる．ネットワーク，マルチ・プロセッサなどによる分散処理環境を考えるときにこの性質は好ましくない．長期的には，文字コード体系の標準化を検討する過程で同じ字形のコードが 2 種類ある問題は解決されるべきであると考えられるが，現在の段階では無条件の変換を採用するのは時機早尚であろう．

そこで，文字列として読み込まれた場合は，JIS と LISPM の区別を保存するが，そうでない場合は JIS 文字は可能な限り LISPM に変換して読み込むことにした．これを実現するために，JIS と LISPM の対応関係に基づく変換関数を定義する．

```
(char-jis #\a)   =>   #\a
(char-lispm #\a)   =>   #\a
```

(char-lispm #\夏)　=>　#\夏

JIS 文字を印字表現として読み込んだ場合，この変換関数で LISPM に変換してからリードテーブルを参照する．LISPM に変換されない文字は，シンボルの名前を構成する構成文字として扱う．文字列の中ではこの変換を行わない．そうすると，Common Lisp の読み込み機構は JIS 文字で表された S 式を読み込むことができるようになる．このとき，シンボルの名前は JIS で書かれていても印字名は LISPM の大文字に変換される．また，数値の表現は JIS であっても LISPM であっても，数値として読み込まれる．(JIS 文字はリードテーブルに登録されていないため，読み込みマクロとして定義することはできない．)

(eq' lispm 'lispm)　=>　t

(numberp 34)　=>　t

もう一つの問題は，LISPM 文字と JIS の対応する文字を比較することである．先に述べたように Common Lisp では，文字や文字列の比較の際に大文字小文字の区別をするかどうかによって二つの方式がある．これの類似で，アルファベットなどの LISPM と JIS を区別するかしないかの二つの比較方式を導入した．すなわち，従来の比較関数では LISPM と ASCII を区別し，新しく導入した比較関数では区別しない．

(char-equivalent #\% #\%)　=>　t

(char-greaterp #\z #\a)　=>　t

(char-equivalent-greaterp #\z #\a)　=>　nil

Common Lisp では，他にも実際の文字コードに依存することなく文字の大小関係の判定，大文字小文字の変換などの操作ができるように各種の関数を用意している．これらの関数も JIS 文字に対して一貫した結果を得るようにした．すなわち，論理的には JIS 文字に対しては LISPM 文字への変換関数を適用してから，各種の文字関数を適用するようにした．

(char-upcase #\a)　=>　#\A

(char-upcase #\a)　=>　#\A

ひらがなとカタカナの場合にも同様な対応関係と比較の問題がある．日本語ではひらがなは，三つの例外を除いて対応する文字がカタカナに含まれている．この状況はアルファベットの大文字小文字の関係とよく似ているが，外来語はカタカナで表記するというような慣習もあり，完全に同じでよいかは疑問がある．現在のところ，シンボルの印字名を一方に変換するようなことはしておらず，次のような変換関数を用意するにとどめてある．

(char-hiragana #\ア)　=>　#\あ

(char-hiragana #\a)　=>　#\a

ところが，国内で一般に用いられている ASCII を拡張した文字セット(JIS $_{x}$0201)には，半角カタカナと呼ばれる JIS とは別のカタカナが含まれており，これが問題を複雑にする．この文字セットを LISPM 文字セットに加えて使用しようとすると，半角カタカナと JIS のカタカナの対応を考えてやらなければならない．ここでのもっともむずかしい問題は，濁音および半濁音である．というのは，半角カタカナでは濁音は濁点を続けた 2 文字を用いて表されている．これに対して，JIS では 1 文字で濁音を表

すことができる．JIS にも濁点が文字として含まれているので，2 文字の表現が可能であるとはいえ，明らかに 1 文字の表現の方が好ましいだろう．また，LISPM 文字コードでは，半角カタカナに相当する部分に ISO のヨーロッパ仕様の文字を割り当てている．これをどうするかという問題もあり，現在，半角カタカナはサポートしていない．(文字列として読み込まれたデータは保存されるが，カタカナとしては表示されない．)

5. ウィンドウ・システム，ファイル・システム，ネットワーク

5.1 ウィンドウ・システム

KS-301 のコンソールはビットマップ・ディスプレイを用いており，ウィンドウ・システムでマルチ・ウィンドウをサポートしている[8]．文字の表示は各種のフォントを用いてソフトウェアで制御しているため，日本語文字のフォントを用意すれば文字の表示そのものは問題なく行うことができる．しかし，3 章で述べた long-char 型の文字を日本語文字として表示するためには多少の工夫が必要であり，その点について簡単に述べる．

ウィンドウ・システムでは，フォントを字形イメージのベクタとして持っている．LISPM 文字の場合，文字コードを添字としてこのベクタを参照することにより，その文字の字形データをビットマップとして得ることができる．LISPM 文字セットはフォントのベクタを有効に使用するため，通常の ASCII では制御文字として用いられている部分に拡張文字を配置し，制御文字は 128 以降の領域を用いている．これにより，フォントはベクタとしては 128 の長さを持ち，その中を無駄なく用いている(最新版では ISO ヨーロッパ仕様の文字が 160 以降 256 まで詰められている)．

各ウィンドウにはフォント・マップと呼ばれる使用可能なフォントのベクタを持っており，文字オブジェクトのフォント属性はこのベクタに対するインデックスとして用いられ，実際に用いられるフォントとの対応が付けられる．(このベクタの大きさは通常 26 に固定されており，26 を超えるフォントを同時に使用することは通常はない)

このことを用いて，フォント属性付の文字列を用いてマルチ・フォントのテキストをウィンドウに表示することができる．また，フォント・マップ中のフォントを調べることで，ウィンドウの行の高さを決めることができる．

JIS 文字は数千個の文字から構成されているが，JIS x 0202 規格に従って文字コードを 2 バイトに分割したときにそれぞれのバイトが制御文字コードのにならないようにするため，文字コードは連続的に使用されておらず空き領域がかなりある．現在のフォントのデータ構造では，このようなコードに対する字形データを保持しようとすると記憶領域が無駄になる．そこで，このような多字種文字のフォントの概念を表すために，いくつかのフォントを組にしたフォント・グループを導入することにした．フォント・グループはフォントのベクタであり，多字種文字の文字コードの一部を添字としてフォント・グループを参照し，使用するフォントを決定する．さらに，文字コードの残りを用いて得られたフォントを参照し字形データを得る．

JIS 文字は 84 の区に分かれており，それぞれは連続した文字コードを持っている．そこで各区を一つのフォントとし，フォント・グループを区番で参照すればよい．区番点番は 2 バイトの文字コードのそれぞれ上位バイト，下位バイトに相当し，LISPM

文字の場合のフォント属性とコード属性から字形を得るメカニズムとほぼ同様である(このため，日本語テキストを表示する速度はマルチ・フォントのテキストの表示とほぼ同様である)．

現在，long-char型の文字のフォント属性は常に0であるため，各ウィンドウはただ一つのフォント・グループを持つことができるようになっている．日本語文字を表示する場合も，フォント・マップとフォント・グループからウィンドウの行の高さを計算することができる．日本語のフォント・グループとしては12ドット字形，16ドット字形(JIS X 9051)，および24ドット字形(JIS X 9052)の3種類の字体を用意しており，ウィンドウごとにどれを用いるかを指示することができる．

5.2 ファイル・システム，ネットワーク

KS-301のファイル・システムでは，16ビットのデータをそのまま入出力できるファイル形式をサポートしている．これはコンパイルされたプログラムを保存するために用いられているが，文字ファイルとして用いることもできる．この属性はディレクトリに書かれており，このようなファイルをオープンすると要素が16ビットのストリームが作られる．日本語文字を含むファイルはそのまま一文字当たり16ビットで，この形式のファイルに格納することができる．新しくファイルを作る際は，16ビットのデータを用いることをファイルの属性として指示しなければならない．パス名には日本語文字を用いることはできない．

ネットワークに関しては，KS-301はEthernet*を媒体とするLANに接続することができる．KS-301同士の通信ではChaosnetと呼ばれるプロトコルを用いており，これを用いてファイル転送，電子メイルなどのサービスが提供されている．このプロトコルではコンパイルされたファイルの転送のために，16ビット単位のデータ転送をサポートしている．日本語ファイルを転送する場合は，同じ機構を用いて16ビット単位に行うことができる．メイルおよびリアルタイムの会話であるコンバースでも一部を除いて日本語文字の使用がサポートされている．

他のマシンとネットワークで結合する場合，日本語文字コードの表現が異なるため，文字コードの変換が必要となる．KS-301ではTCP/IPプロトコルをサポートしているが，TCP/IP上での日本語文字のコードの標準がないため，現在のところ日本語を含むファイルの転送はサポートしていない．TCP/IPを用いて結合する相手は主にUnix**マシンであるが，その日本語文字の文字コードは統一されておらず，厄介な問題である．現在，EUCコードあるいはSHIFT-JISコードに変換して転送する実験を行っている．

6. 日本語文字の入力

KS-301のユーザ・インタフェースの最大の特長は，初心者，熟練者双方がもっている使いやすさに対する，相異なる要求を同時に満たしていることである．初心者あるいは，たまにしか使わない利用者には，ポップアップ・メニューをマウスで選択する方式が適している．KS-301では，このためにサジェスチョン・メニューと呼ばれるコ

*Ethernetは米国Xerox社の商標である．
**Unixは米国AT&Tベル研究所が開発し，AT&Tがライセンスしている．

マンドのためのメニューを各ユーティリティごとに用意している．

一方，熟練者用としてはcontrolキー，metaキー，hyperキーといった修飾キーを用いたキーストローク・コマンドによって，キーボードから手を離すことなく，システムと対話することができる．この代表的な例がZmacsエディタである．Zmacsエディタは，エディタ・ソフトウェアの傑作といわれるEMACSをもとに，Lispマシンでのプログラミングのための機能を追加して作られたエディタである．このZmacsの豊富な機能の中から，文字の入力，カーソルの移動，および簡単な編集機能を抜き出したものが入力エディタである．

KS-301のキーボードからの標準的な入力にはすべて，この入力エディタが使われており，使用者プログラムからもこの入力エディタを呼び出すことができる．すなわち，使用者にはコマンドを入力する場面だろうとエディタでの文章入力であろうと，同じインタフェース(メニュー，あるいはキーストローク)で文字を入力してくことができるようになっている．

今回の日本語機能の開発ではハードウェアの変更はしない，という制約があったため，キーボード上にカナ文字はなく，「変換キー」ももちろんない．そこで，入力エディタおよびZmacsエディタにコマンドを追加することで，日本語入力のインタフェースを実現した．これをJIT(Japanese Input Toolkit)と呼んでいる．入力エディタそのものがKS-301の入力フロント・プロセッサとして位置づけられるため，日本語入力機能を各種ユーティリティはもちろんのこと，使用者のプログラムからも自由に利用することができる．

入力の方式は，ローマ字による文節単位の入力／最長一致法によるカナ漢字変換方式を採用した．現在，一部には，二文節最長一致法等による連文節変換あるいは全文変換といった機能をもつパソコンも登場しているが，現時点ではこのワークステーションでの大量の文書入力といった要求は低いと判断したためである．

JITでは，ローマ字変換に関しては二つのモードを設けた．英字変換モードは入力がそのまま表示されるが，ローマ字逐次変換モードでは入力された英字列がローマ字として確定した瞬間，表示がかな文字に変更されるモードである．この逐次モードは，エラーメッセージのような文章を入力するときに便利である．Lispのプログラムを入力する場合は，依然として英数字を入力することが多く英字列(ローマ字列)から直接漢字に変換できると便利である．この英字モードでは，

(defun reidai

と入力した時点で，カーソル位置をそのままにして変換コマンドを実行し，

(defun 例題

とすることができる．

JITでは原則として，Zmacsエディタ，入力エディタでリージョンと呼ばれる領域(文字列)をカナ漢字変換の対象としている．したがって，すでに入力済みの文字列もマウスあるいはキーストローク・コマンドによって，リージョンを指定し変換対象として再変換を行うことができる．また，現在入力中の文字列は，省略時の解釈規則によってリージョンが決定され変換される．このため通常の場合，Zmacsエディタ，入力エディタともに，英文を入力するのとまったく同じ要領でローマ字をキーインし，

変換コマンドを入力することによって日本語に変換していくことができる．日本語入力のための特別なモードやウィンドウは必要ない．

カナ漢字変換でもっとも大きな役割を果たす日本語辞書は，UNISYS DS-7のMicro JUSTYに使用されているものを基にしており，約44,000語が登録されている．KS-301は128 MBの仮想記憶空間を持っており，この辞書をすべて仮想記憶上に木構造状に展開して持っている．これは約2 MBの領域を必要とするので，LAN上に辞書サーバを置いて辞書検索のサービスを提供することにより，辞書の重複を避ける方法を検討している．

このほかに，自然言語処理システムの研究・試作を考慮して，変換の過程で得られる形態解析の結果を他のプログラムから容易に利用できるようにしてある．

7. プログラミング環境などでの日本語の使用

Zmacsエディタでは，前章で述べたカナ漢字変換を他のコマンドと一貫した形式でサポートしている．カーソルの位置づけは日本語の場合も文字単位に行われ，英数字のマルチフォントと同時に日本語を用いることができる．Zmacsでは，ファイルの先頭の行を属性行としてそのファイルに関する各種の情報を指示することができ，表示に用いられるフォントのリストもそこに記録される．日本語文字のフォント・グループも属性行に記録できるようにした．

また，Zmacsでは英文のテキストを取り扱うための各種の機能が用意されている．たとえば，単語あるいは文単位のカーソルの移動や文章の整形などであるが，これは英文が空白を単語の区切りとしていることに依存しているため，日本語のテキストを取り扱うためには不十分である．これらの機能は現在検討中である．

その他，インスペクタ，デバッガなどのツールでは，日本語文字の表示，入力エディタによる日本語入力をサポートしており，英字のみの場合と同様に使用することができる．また，各種のメニューなどでも日本語の表示が可能で，日本語の文字列をメニューの表示項目として与えることにより，従来と同様の方法で使用することができる．

KS-301では，オンラインでシステムの使用法に関して各種の情報を得ることができるが，これらの場面でも日本語文字を使用することができる．たとえば，関数のドキュメンテーションを日本語で与えておけば，入力エディタのコマンドでそれを表示させることができる．また，入力エディタなどのコマンドに関する情報を与えるサジェスチョン・メニューでも日本語が表示可能で，カナ漢字変換に関するコマンドの説明は日本語で与えられる．これは日本語使用のレベル4に相当する．

KEEでも，以上に述べたようなKS-301の日本語文字の取り扱いが基本的には可能になった．これはKEE自身はLispで実現されているため，Lispで日本語文字が使用可能であればKEEで使用することも可能である．ただし，KEEは独自のウィンドウ・システム，グラフィックス・パッケージを持っており，そこで日本語文字を使用するためにはさらに改造が必要である．現在，一部の例外を除いてKEEでも日本語を使用することができ，アプリケーションもいくつか開発されている．KEEのアプリケーションで日本語を使用することは，使いやすさの改善に非常に有効であると考えられ，

またこれは実際のアプリケーションを通して実証されている．

8. おわりに

Lispマシン KS-301 の日本語 Common Lisp 環境(NCL)について，その実現法の概略を含めて紹介した．Lispマシンという特殊な条件ではあるが，プログラム言語および環境を通して一貫した日本語機能をサポートしている．さらにアプリケーションでの日本語の使用も着々と進んでいる．

Lispマシンの環境はもともと非常に強力であるが，英語だけではなく日本語の環境が使用できるようになることで，わが国でもその真価を発揮するだろう．とくに KEE を使用したエキスパート・システムの応用で，日本のユーザに合わせたインタフェースが期待できる．また筆者らもソフトウェアの開発保守の場面で，電子メイル，ドキュメンテーションの作成などで日本語機能の恩恵を受けている．今後さらに，入力法の改善および日本語文書編集のサポートなどを充実する予定である．

参考文献

[1] ユニバック知識システム KS-301 概説書，日本ユニバック，1986.
[2] 大田他，“KS-301(Explore)”，bit 別冊，高機能ワークステーション，共立出版，1987.
[3] 真田，“プログラム言語の日本語化実験—日本語 PL/I の作成—”，日本ユニバック技報，No. 7，1984.
[4] 元吉，Common Lisp における日本語処理方式の提案，情報処理学会記号処理研究会 40-6，1987.
[5] “日本語処理に関する調査”，マイクロ・コンピュータに関する調査報告書—Lisp 技術に関する調査—，日本電子工業振興協会，1987.
[6] G. Steel，“Common Lisp：The language”，Digital Press.（邦訳 井田，“Common Lisp”，共立出版）
[7] JIS ハンドブック情報処理—1987，日本規格協会，1987.
[8] 大田，“Lisp マシンのウィンドウ・システム”，日本ユニバック技報，No. 13，1987.

執筆者紹介 大 田 一 久(Kazuhisa Ohta)

昭和 56 年慶応義塾大学工学部数理工学科卒業．同年，日本ユニシス(株)入社．エンド・ユーザ言語の開発に従事．60 年から Lisp マシン開発の業務を担当．現在システム企画本部 システム統括一部 知識システム部 基本システム課に所属．情報処理学会，日本ソフトウェア科学会会員，kaz@unisys. junet.

プログラム自動生成システム——TSX-PASE

TSX-PASE——The Program Generation System

長谷川　邦夫

要　約　TSX-PASE (プログラム自動生成ツール) は，統合的システム開発環境構築支援システム TSX 1100*に含まれるツールである．

ファイル処理を定型化した言語 PASE 1100**による仕様記述から COBOL ソース・コードを生成する仕組みに，ソフトウェア部品の再利用の技法と，情報資源を管理するデータ・ディクショナリを活用する技法を応用したものである．

部品としては，仕様記述の骨組であるスケルトンと，業務固有の機能モジュールを表現するマクロを利用できる．スケルトンおよびマクロには，データ・ディクショナリの中の情報を参照する機能がある．

データ・ディクショナリを活用できる部品の利用によって，仕様記述のレベルが上がることと，自動生成の効果が増大することが期待できる．

Abstract　TSX-PASE is positioned as one of the software tools included in TSX 1100——a software product supportive constructing environments for systems development.

In developing TSX-PASE, applied were three techniques enabling the generation of COBOL programs from the specifications given in PASE 1100 language, the re-using of software components and the utilization of a data dictionary built for information resource management.

Skeletons and macros are usable as software components. The former is for parameterized descriptions of program specifications, while the latter is to describe the functional modules of applications. Both types of components are capable of reference to the information in the TSX data dictionary.

Availability of those software components has brought about the higher-level description of specifications and improved efficiency in code generation.

1. はじめに

本稿は筆者らが設計・開発してきたソフトウェア，TSX-PASE についての技術報告である．TSX-PASE は，統合的ソフトウェア開発支援システム TSX 1100 を構成するソフトウェア・ツールの一つであり，データ・ディクショナリの活用とソフトウェア部品の再利用によるプログラム自動生成ツールである．

ソフトウェア開発の生産性は，より高きを求めてやまない目標になっているが，作ってはやがて消費されていく物品の生産と違うところに問題が生じてくる．

すなわち，ソフトウェア保守が重要な問題になってくる．大量のマン・パワーの投入で開発したものの，大規模かつ複雑になった情報システムの保守には業務内容とシステム化技術に通じた多くの担当者の力が必要である．輪をかけて，大量の要員を利

* Total Support eXecutive system
** Program Automatic Specification Environment

用した結果，かなりのソフトウェアについて，その内部がブラック・ボックスになってしまっている．これは，ソフトウェアの保守自体を不可能にしてしまう危険をはらんでいることになる．

膨大な量の既存ソフトウェアが，新しいソフトウェア開発のために有効活用されることが少ないのも，大きな問題である．

ソフトウェア保守の問題とソフトウェア資産の活用の問題は，ソフトウェアを開発する段階で考えておかなければならない．既存のシステムの保守効率を上げるためには，せいぜい次の対策があげられる．

1) 文書の整備
2) ツールによるプログラムの整形
3) ツールによるシステム構成物の静的解析

既存システムのソフトウェア資産を新しいシステムのソフトウェア開発に流用することがあるが，既存システムが流用されることを念頭において開発されたものでなければ，この資産利用の方法は，既存システムの持つ問題点を新しいシステムの中に持ちこんでしまう危険性がある．

システムを新たに開発するとき，将来の保守が効率よく行われるように図る手段として，たとえば次の対策が有効である．

1) ソフトウェアの部品化と，その組み合わせによるコード生成
2) データ・ディクショナリによるデータ資源の一元管理

ソフトウェアの部品化とは，ソフトウェアを再利用可能なものとして作成することである．企業や業務のレベルで標準的に使うデータ定義や処理手続きは，部品化することができる．ソフトウェア部品の利用によってソフトウェアの作成量が減少するだけでなく，ソフトウェアが見た目に単純で理解しやすいものになることが期待できる．

部品化することによって，ハードウェアや基本ソフトウェア，あるいはデータベースやファイルの物理的性質に依存する部分，つまり物理的情報を部品の中に隠蔽することができる．この論理的部品を使うことで，高レベルの仕様表現が得られる．

TSX-PASE では「マクロ」と呼ぶ部品と，「スケルトン」と呼ぶプログラム骨格を使うことができる．スケルトンは標準的なプログラム構造を記述したもので，その記述の中でマクロ部品を使うことができる．個々のプログラムの作成においては，そのプログラムに適当なスケルトンを選択し，そのプログラムに固有な記述を追加して完成させる．プログラムに固有な記述の中でもマクロ部品を使うことができる．マクロ部品は，ソース・コードレベルの部品である．

TSX-PASE では，単なるデータ定義や手続きの部品化だけでなく，データ定義を含んだ手続きの部品化ができる．これによって論理的部品の作成が容易になる．

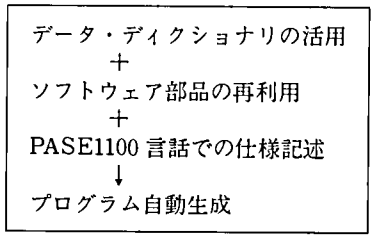
TSX-PASE では，スケルトンやマクロを使ってプログラムの仕様を記述する．仕様記述には PASE 1100 言語を使う．TSX-PASE のソースコード生成機能は，スケルトンやマクロ部品を処理して COBOL によるソース・コードを生成する．

データ・ディクショナリは，システムで扱うデータの仕様定義である．データ仕様を文書ではなく，ソフトウェア・ツールを用いて機械処理のできる形に表現，蓄積し，それを検索・更新・分析することができれば，システム仕様の明確化と確実な伝達が

期待できる．また仕様の変更にも確実かつ容易に対処できる．

TSX におけるデータ・ディクショナリは，データ定義だけでなく，システム開発で作られるすべての成果物に関するディクショナリである．すなわち，システム仕様の表現である．データ・ディクショナリは，それ自体，論理的部品として機能する．事実，TSX-PASE での部品利用では，ディクショナリの内容から自動的にソースコードを生成させることができる．

TSX-PASE の機能概要は，次の図式でとらえることができる．



2. TSX 1100

2.1 TSX 1100 の目的

TSX 1100 はシリーズ 1100／2200 上に，ソフトウェア開発・保守の生産性向上および品質向上を目指した，ソフトウェア開発環境構築の支援を行うことを目的としている．

TSX 1100 が支援する環境とは，ソフトウェア開発のライフサイクル全般にわたって標準化が図られ，ソフトウェア資産や開発に必要となる情報がソフトウェア・データベース（単にディクショナリとも言う）で統合化・共有化され，開発作業がシステム化された環境である．

2.2 TSX 1100 の構成

TSX 1100 は，基本部と標準ツール群から構成される．基本部は単独で機能するが，すべてのツールは基本部の制御の下で機能する．

1) 基本部 (TSX-BASIC)……基本部は，TSX 1100 のかなめであるディクショナリの管理と，開発作業の基本的操作の支援（テキストの編集や翻訳）を機能としている．

2) 標準ツール……標準ツールには，基本部が支援する基本的操作よりも複雑，かつ複合的な機能が用意されている．現在（昭和 63 年 4 月），次に示す七つのツールがあり，ディクショナリに蓄積された情報を通して機能を果たす．ディクショナリの情報がツール間で共有されることによって，ツール間の連動が図られる．

① TSX-DEFN（データ定義支援）

業務システムのデータ定義，およびシステム構成物の定義からディクショナリを構築するためのツールである．

② TSX-PET（データベース効率見積り支援）

業務システムのデータベース設計時に，効率を見積るためのツールである．

③ TSX-GCOB（ソース・コード作成支援）

TSX 1100が用意している五つの基本プログラム・パターンを組み合わせて目的プログラムの骨組みを生成し，さらにオウン・コードを挿入するツールである．

④ TSX-PASE（プログラム自動生成）

⑤ TSX-TEST（テスト支援）

テスト・データの準備，およびモジュール・テストの実行などを支援するツールである．

⑥ TSX-ADMS（プログラム管理支援）

プログラムを一元管理し，保守作業の統制を行うツールである．

⑦ TSX-MANAGE（マネジメント支援）

開発状況の管理，および管理レポートの作成などを支援するツールである．

TSX 1100の機能は基本的にはIPF*の対話セッションの中で実行させることができる．すなわちTSX 1100はディクショナリを持つ，設計者のための作業台，プログラマのための作業台，管理者のための作業台になるものである．

2.3 TSX 1100のディクショナリ

TSX 1100の最も大きな特徴は，そのディクショナリにある．TSX 1100のディクショナリは，開発環境のモデルである．ソフトウェア開発環境を表現するのにふさわしい表現方法として，TSX 1100ではERモデル**による表現を採用している．ERモデルでは実世界のものの性質や，ものとものとの関連が自然に表現できる．ソフトウェア開発環境という実世界をどのようにとらえるか（どんなものが存在するか，どんなものとものとの関連があると考えるか）は一意ではない．

TSX 1100のディクショナリは，任意の環境モデルを表現できる．つまり任意の実体集合，関連集合と実体や関連の属性を定義することができる．ただし図1は標準的な開発環境を想定して，それを表現した標準モデルと呼ぶディクショナリの標準形である．

TSX 1100の標準ツールは，標準モデルのディクショナリを使うようになっている．標準モデルには，データ項目，データ・レコード，データ・ファイルなどのいわゆるデータ・ディクショナリの部分がある．また，モジュールが（別の）モジュールで構成されるという関連や，モジュール（の仕様）はシンボリックというもので実現されるという関連などが含まれている．これらはソフトウェア仕様や，ソフトウェアの開発行為の表現である．

このようにTSX 1100のディクショナリは，単なるデータ・ディクショナリにとどまらず，開発作業全体のディクショナリになり得るものである．本当にそのようなディクショナリになるかどうかは，ディクショナリを活用するツール次第である．

3. TSX-PASEの機能

TSX-PASEの主な機能は次の三つである．

* Interactive Processing Facility，プログラム開発の工程を円滑，かつ効率よく進めるための環境作りを支援するプログラム．

** Entity-Relationship Model，実体・関連モデル

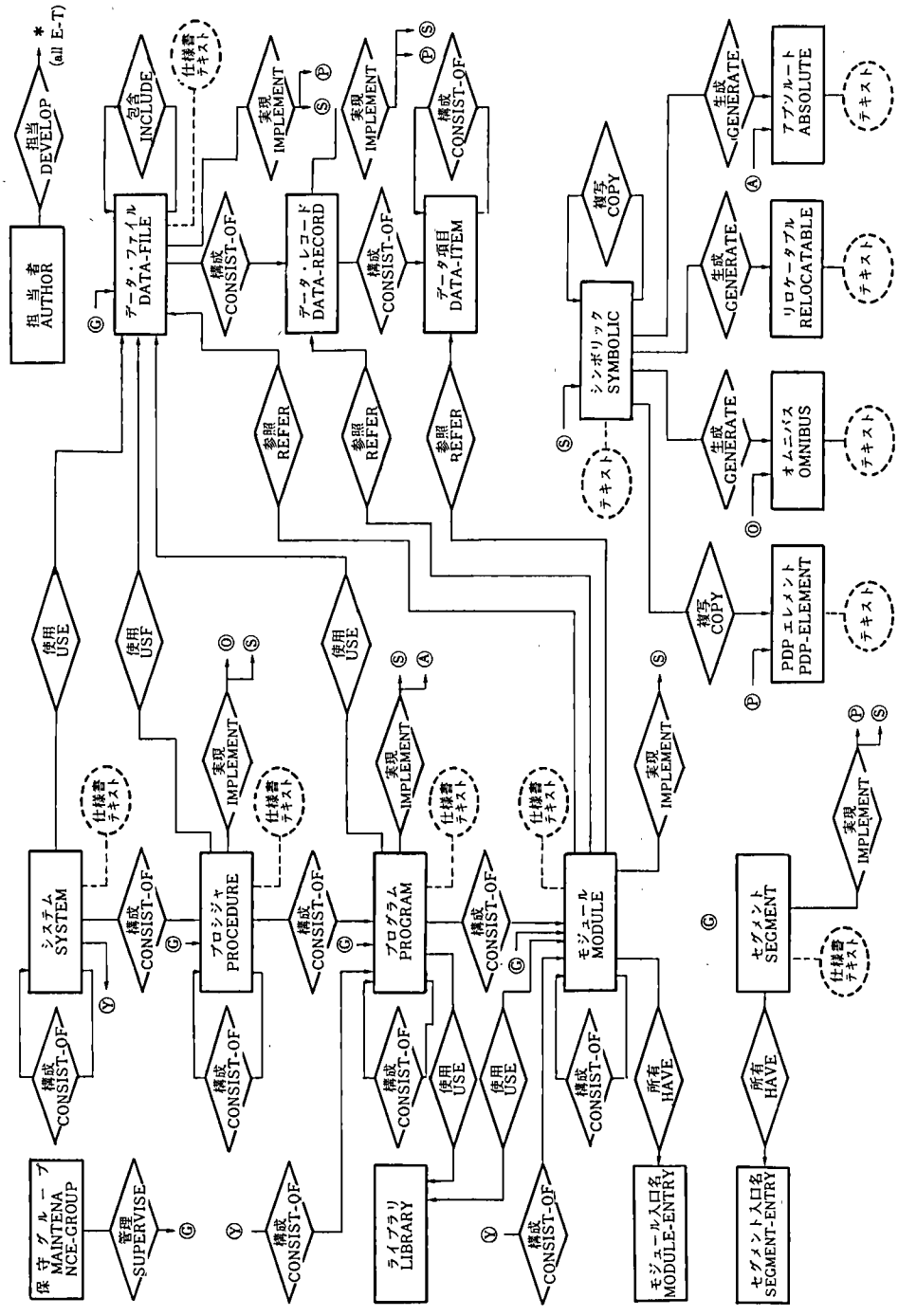


図1 ディクショナリの標準モデル[1]（長方形は実体を表し，菱形は実体と実体の間の関連を表す）

Fig. 1 A diagram of the standard modol of TSX-dictionary in entity-relationship expression

1) モジュール仕様の定義
2) 部品仕様の定義
3) ソース・コードの生成

TSX 1100の標準モデル・ディクショナリにおいては，翻訳の単位をモジュールと呼ぶので，TSX-PASEで仕様を定義する単位もモジュールとなる．

モジュール仕様は，次の三つの部分により定義される．

1) 属性……モジュール実体の属性として表現できる内容．たとえば，メインとサブの区分，リアルとパッチの区分やモジュールの機能説明などである．テキストの記述に使うスケルトンの名前も属性の一つである．扱う属性の種類は任意である．
2) 関連……ディクショナリ上に関連として表現できる内容．たとえば，どのデータ・ファイルをどんな形で読み書きするかという仕様である．
3) テキスト……属性や関連としては表現できない事柄．つまり処理のアルゴリズムや計算，加工の方法などである．このテキストの記述にスケルトンを利用することができる．その場合には，スケルトンに挿入する目的モジュールの固有処理に関する記述（利用者テキスト段落と言う）のみを用意することになる．またスケルトンには，その原文を変化させて使うためのスケルトン・パラメタという変数を持たせることができる．そのようなパラメタがある場合には，パラメタに特定の値を与えることも，テキストの定義に含まれる．

部品仕様の定義は，属性とテキストとで行う．部品のテキスト記述にはスケルトンはないが，部品定義の簡単な言語が存在する．部品はディクショナリ上，セグメント実体で表現する．

ソース・コードの生成機能は，モジュール仕様から，その実現としてのシンボリック実体のテキストを生成するものである．生成の過程でスケルトンと利用者テキスト段落が一体化し，その中で引用されているマクロ，さらにマクロの中で引用されているマクロが展開される．またスケルトンやマクロの中でディクショナリへの参照があれば，その参照が行われて生成コードに反映されることになる．

図2は，モジュール仕様定義の例を文書化したモジュール仕様定義リストである．

モジュール仕様定義リストには属性の定義，関連の定義およびテキストの定義の内容が出力されている．すなわち名称，説明，機能，モジュール型，メイン・サブ区分などが属性として定義されている．

作成者，モジュール入口名，構成モジュール，使用ライブラリ，参照ファイル，参照レコードなどは，モジュールとそれらの間の関連が存在する，という表現で定義されている．ファイルがどのような外部ファイル名と識別子で参照されるか，および入力，出力のどちらのために参照されるかということは，モジュールとそのファイルとの関連として定義されている．

このように，モジュール仕様の内容は，単に文書として記録，管理されるのではなく，属性や関連という情報に細分されている．TSX 1100のディクショナリは，その単位で記録されている．テキストの定義は，あるスケルトンをベースとして，それに与えるパラメタの値およびオウンコード（利用者テキスト段落）で構成される．

```
定義リスト(型名:MODULE        )          ﾓﾃﾞﾙ名:PASE

実体名:MOD001            ﾊﾞｰｼﾞｮﾝ名:0                                    ﾍﾟｰｼﾞ:  01
--------------------------------------------------------------------------------
属性定義
--------------------------------------------------------------------------------
作成者:AKASAKA-TAROO
名称   :商品売上げﾏｽﾀ更新
説明   :
  商品売上げﾏｽﾀを 売上げﾌｧｲﾙで更新する.
  売上げﾌｧｲﾙの金額を ﾏｽﾀに累積加算する.
  ﾏｽﾀにない商品の売上げﾃﾞｰﾀは そのままﾏｽﾀ･ﾃﾞｰﾀとする.


--------------------------------------------------------------------------------
機能          :SEQUENTIAL MATCHING
ﾓｼﾞｭｰﾙ型      :BATCH
ﾒｲﾝ･ｻﾌﾞ区分   :MAIN
ﾓｼﾞｭｰﾙ入口名:
構成ﾓｼﾞｭｰﾙ    :
使用ﾗｲﾌﾞﾗﾘ    :MATER-R, MASTER-R2, TRANS-R
    .
    .
    .
    .
    .
参照ﾌｧｲﾙ      :
----------------------:--------------:--------------:----:--------------------
ﾌｧｲﾙ名                :外部ﾌｧｲﾙ名    :識別子        :入出:参照ﾚｺｰﾄﾞ名
----------------------:--------------:--------------:----:--------------------
MASTER                :OLD-MASTER    :'S-MASTER'    : IN :MASTER-R
TRANS                 :TRANS         :'S-TRANS'     : IN :TRANS-R
OUT1                  :NEW-MASTER    :'S-MASTER'    : OUT:MASTER-R2
----------------------:--------------:--------------:----:--------------------
```

```
定義リスト(型名:MODULE        )          ﾓﾃﾞﾙ名:PASE

実体名:MOD001            ﾊﾞｰｼﾞｮﾝ名:0                                    ﾍﾟｰｼﾞ:  02
--------------------------------------------------------------------------------
ﾃｷｽﾄ定義      ﾃｷｽﾄ名:PASE$TEXT.MOD001
              ｽｹﾙﾄﾝ名:PASE$SKEL.SKELETON001
--------------------------------------------------------------------------------
<ｽｹﾙﾄﾝ･ﾊﾟﾗﾒﾀの値 >
%P1:出力ﾌｧｲﾙの数(1,2):1
%P2:ﾏｽﾀｰﾌｧｲﾙ名          :商品ﾏｽﾀ
%P3:ﾄﾗﾝｻﾞｸｼｮﾝ名         :売上げﾃﾞｰﾀ
%P4:出力 1 ﾌｧｲﾙ名       :OUT1
%P5:出力 2 ﾌｧｲﾙ名       :

--------------------------------------------------------------------------------
<利用者ﾃｷｽﾄ段落>
##TEXT1:WORKING-STORAGE の記述:

##TEXT2:MATCHING-KEY     の記述:
          KEY1 / KEY1
##TEXT3:ﾏｽﾀｰだけの場合の処理  :
          MASTER-R2 を作成する , 但し対応移送は MASTER-R
##TEXT4:ﾄﾗﾝｻﾞｸｼｮﾝだけの場合の処理:
          MASTER-R2 を作成する , 但し対応移送は TRANS-R
##TEXT5:ﾏｯﾁしている場合の処理:
          MASTER-R2 を作成する , 但し対応移送は MASTER-R
                              , 編集方法は ｻﾝﾀﾞｶｺｳｼﾝ
##TEXT6:PASE編集手続きの記述:
        編集方法 ｻﾝﾀﾞｶｺｳｼﾝ
           代入 : 金額 OF MASTER-R2 = 金額 OF MASTER-R
                                    + 金額 OF TRANS-R;
##TEXT7:COBOL手続きの記述   :
        入力時処理 売上げﾃﾞｰﾀ :
        ﾚｺｰﾄﾞ条件は       商品ｺｰﾄﾞ OF TRANS-R NOT = ZEROS
```

図2 モジュール仕様定義の文書化

Fig. 2 A documentation of module specification

図2の例では，パラメタの値およびオウンコードの内容が日本語化されているが，実際のテキストはPASE 1100の文法で書かれている．たとえば## TEXT 4(これは段落名と呼び，その後の：と：の間は段落の内容を示す簡単な注釈である）の部分の

MASTER-R2を作成する，但し対応移送はTRANS-R

とあるのは，実際のPASE 1100での記述では，

EDIT MASTER-R2 FROM TRANS-R

となっているのである．

スケルトンをベースとして，そのオウンコードだけを用意する方法は，モジュールの制御構造を隠してしまうことができる．

さらに，記述のためにPASE 1100を使うというのは，オウンコードの単位をファイル処理の論理レベルで設定することができるという良さを持つ．

4. 部品定義言語

部品にはスケルトンとマクロがある．どちらもテキストの形で引用される（図3）．部品のテキストは，それを使うモジュールのテキストと同一の言語で書かれていなければならない．つまりPASE 1100言語である．スケルトンやマクロのテキスト文は生成コードのテキスト文になるものであるが，いくつかの特別な文，あるいは語が生成コードとしての書き出しを制御するために使われる．以下，概略を述べる．

1) パラメタ参照……%で始まる語はパラメタである．テキストの中のパラメタは，その値で置き換えられる．パラメタには次の種類がある．

① スケルトン・パラメタ

スケルトンの中で使うパラメタで，値はモジュールのテキスト定義時に利用者が指定する．

② マクロ・パラメタ

マクロ引用で引数として値が渡される．

③ ディクショナリ参照パラメタ

ディクショナリの中の情報を値とする．通常，実体の名前や，実体あるいは関連の属性値が値になる．

④ 予約パラメタ

目的モジュールの実体名，バージョン名を値に持つ．パラメタ名は，あらかじめ約束されている．

⑤ 変数パラメタ

テキストの解釈の過程で，値を任意に変えることのできるパラメタ．値を変えるために代入文と加算，減算文がある．

また，別の分類でパラメタはローカルなものとグローバルなものとに分けられる．ローカルなパラメタは，そのマクロあるいはスケルトンの中だけで有効であるが，グローバルなパラメタは，他のマクロの中でも同一のパラメタとして使うことができる．

2) マクロ引用……&で始まる語はマクロ引用である．生成コードには，そのマクロのテキストが展開される．マクロ引用では，引数の並びで引用されるマクロのマ

```
1%P1 出力ファイル の数(1,2):
1%P2:マスターファイル名        :
1%P3:トランザクション名        :
1%P4:出力 1 ファイル名         :
1%P5:出力 2 ファイル名         :
2##TEXT1:WORKING-STORAGE の記述
2##TEXT2:MATCHING-KEY    の記述
2##TEXT3:マスターだけの場合の処理  :
2##TEXT4:トランザクションだけの場合の処理:
2##TEXT5:マッチしている場合の処理:
2##TEXT6:PASE編集手続きの記述:
2##TEXT7:COBOL手続きの記述    :
5&ID-DATA
5$FUNC FUNC-OF IN1 : %P2 ;
5      FUNC-OF IN2 : %P3 ;
5      FUNC-OF OT1 : %P4 ;
5 *IF %P1 = 2  *THEN
5      FUNC-OF OT2 : %P5 ; *ENDIF
5$WORK
5##TEXT1
5
5$PROCESS-FLOW
5 MATCH-P DEFINE OT1 *IF %P1 = 2 *THEN
5             AND OT2 *ENDIF
5              ON IN1 AND IN2
5         MATCHING-KEY ##TEXT2;
5
5$PROCESS MATCH-P
5$OUTPUT  OT1 *IF %P1 = 2 *THEN OT2 *ENDIF
5
5$WHERE RECORD-OF IN1 AND NO-RECORD-OF IN2
5##TEXT3
5
5$WHERE NO-RECORD-OF IN1 AND RECORD-OF IN2
5##TEXT4
5
5$WHERE RECORD-OF IN1 AND RECORD-OF IN2
5##TEXT5
5
5##TEXT6
5
5##TEXT7
5
5&IMP
5/* END OF SKELETON001 */
```

左に1とある部分がスケルトン・パラメタ目録

左に2とある部分が利用者記述テキスト目録

左に5とある部分がスケルトン・テキストの本文

図3 スケルトンの例

Fig. 3 Example of a text-skeleton

クロ・パラメタに値を渡すことができる.

3) 条件文……

*IF 条件式 *THEN テキスト文 [*ELSE テキスト文] *ENDIF

の条件文で, 生成コードに含めるテキスト文を選択できる. 条件式は一つの比較条件, あるいは二つ以上の比較条件を AND または OR で連結したものである. 比較条件の項目は, パラメタあるいは定数である. 演算子は=, >およびくで, 各々に否定記号 NOT を付けることができる.

4) 繰り返し文……*FORALL %パラメタ名 テキスト文 *ENDFOR

の繰り返し文で生成コードに含めるテキスト文を繰り返すことができる. パラメタ名は, この場合ディクショナリ参照パラメタである. このパラメタの値がディクショナリに複数個存在する場合, その値ごとにテキスト文の生成コードの中への書き出しが行われる.

5) 代入文, 加算文, 減算文

*SET パラメタ名 値またはパラメタ名-2

*ADD 1 パラメタ名

＊SUB 1　パラメタ名

上から順に代入，加算，減算を命ずる文である．加算，減算は数値１の加算，減算である．

6)　文字連結子……記号へは，その前後の文字列あるいはパラメタの値を連結して，生成コードの中に書き出すことを示す．

図３および図４は，それぞれスケルトンとマクロの例である．

スケルトン・テキストの中の## TEXT 1のように##で始まる語は，そこにオウンコード（利用者テキスト）を挿入できることを示すものである．それがどういう内容のテキストであるべきかが，利用者記述テキスト目録に示されている．

また，スケルトンの原文に変化を与えるためのパラメタにはどういうものがあるかが，スケルトン・パラメタ目録に示されている．利用者が見るものは，この二つの目録だけであり，スケルトン・テキストの本文を見ることはない．

マクロの中で使われるパラメタにはどういうものがあるかが，それぞれのマクロ定義の中に，マクロ・パラメタ目録として示されている．マクロ-その２（図４）の例では，%P２と%P３はこのマクロを引用するときに与えられる引数であり，%D１は，

マクロ-その１（ID-DATA）

```
6&ID-DATA
3%D1 :FILE-NAME:MODULE,%OBJ,REFER,DATA-FILE
5 $ID %OBJ
5 $DATA-FLOW INPUT *FORALL %%D1
5                  &IO-FILE  %D1,I       *ENDFOR
5            OUTPUT *FORALL %%D1
5                  &IO-FILE  %D1,O       *ENDFOR
5               I-O *FORALL %%D1
5                  &IO-FILE  %D1,IO      *ENDFOR
5 $DATA
5 *FORALL %%D1
5      &FILE-DEF %D1
5 *ENDFOR
```

左に３とある部分がマクロ・パラメタ目録，

左に５とある部分がマクロ・テキスト本文．

このマクロの中では，別のマクロ（IO-FILE と FILE-DEF）を

繰り返し引用している．

マクロ-その２（IO-FILE）

```
6&IO-FILE     %P2,%P3
4%P2 :FILE-NAME  :
4%P3 :I-O        :
3%D1 :IN-OUT     :MODULE,%OBJ,REFER,DATA-FILE,%P2,PHYSICAL-FIL-INF;(1,3)
5 *IF %D1 = %P3 *THEN
5 %P2           *ENDIF
```

左に４，あるいは３とある部分がマクロ・パラメタ目録．

ここでの %D1 は，モジュール %OBJ とデータ・ファイル %P2 の間の REFER という

関連の属性 PHYSICAL-FIL-INF の頭の３桁を指している．

図４　マクロの例

Fig. 4　Examples of Macro definition

```
$ID MODULE001
$DATA-FLOW INPUT
                    MASTER
                    TRANS
           OUTPUT
                    OUT1
              I-O
$DATA
       $FILE MASTER
       $RECORD MASTER-R
                COPY MASTER-R
                     .
       $FILE TRANS
       $RECORD TRANS-R
                COPY TRANS-R
                     .
       $FILE OUT1
       $RECORD MASTER-R2
                COPY MASTER-R2
                     .
$FUNC FUNC-OF IN1 :
                    MASTER
                        ;
      FUNC-OF IN2 :
                    TRANS
                        ;
      FUNC-OF OT1 :
                    OUT1;
$WORK

$PROCESS-FLOW
 MATCH-P DEFINE OT1
            ON IN1 AND IN2
         MATCHING-KEY
                                KEY1 / KEY1
                               ;
$PROCESS MATCH-P
$OUTPUT  OT1

$WHERE RECORD-OF IN1 AND NO-RECORD-OF IN2
        EDIT MASTER-R2 FROM MASTER-R

$WHERE NO-RECORD-OF IN1 AND RECORD-OF IN2
        EDIT MASTER-R2 FROM TRANS-R

$WHERE RECORD-OF IN1 AND RECORD-OF IN2
        EDIT MASTER-R2 FROM MASTER-R
                            BY ｻﾞﾝﾀﾞｶｺｳｼﾝ

       $EDIT ｻﾞﾝﾀﾞｶｺｳｼﾝ
          SET AMOUNT OF MASTER-R2 = AMOUNT OF MASTER-R
                                  + AMOUNT OF TRANS-R;

       $INPUT TRANS :
       ACCEPT-RECORD ON KEY1 OF TRANS-R NOT = ZEROS

$IMPLEMENT
 $PROG
  $FILE MASTER
       :DISC,ST ,10 ,OLD-MASTER
       FILE-ID:'MASTER'
  $FILE TRANS
       :DISC,ST ,10 ,TRANS
       FILE-ID:'TRANS'
  $FILE OUT1
       :DISC,OM ,50 ,NEW-MASTER
       FILE-ID:'MASTER'
```

この部分は，マクロ ID-DATA の引用によって生成された．

この部分は，マクロ IMP の引用によって生成された．

図5 生成されたコード（PASE 1100 言語）の例

Fig. 5 A text in PASE1100 language generated from module definitions

```
PASE SOURCE IMAGE IN JAPANESE    PAGE   1

 1    プログラム名 MODULE001
 2    データフロー 入力
 3                    商品ﾏｽﾀ
 4                    売上げﾃﾞｰﾀ
 5                出力
 6                    OUT1
 7                入出力
 8    データ定義
 9          ファイル定義 商品ﾏｽﾀ
10          レコード定義 MASTER-R
11                COPY MASTER-R
12
13          ファイル定義 売上げﾃﾞｰﾀ
14          レコード定義 TRANS-R
15                COPY TRANS-R
16
17          ファイル定義 OUT1
18          レコード定義 MASTER-R2
19                COPY MASTER-R2
20
21    使用者関数定義 関数名 IN1 :
22                    商品ﾏｽﾀ
23                    ;
24                関数名 IN2 :
25                    売上げﾃﾞｰﾀ
26                    ;
27                関数名 OT1 :
28                    OUT1;
29    作業領域
30
31    プロセスフロー
32     MATCH-P の出力は OT1
33                入力は IN1 および IN2
34            照合キイは
35                                  商品ｺｰﾄﾞ / 商品ｺｰﾄﾞ
36                                ;
37
38    プロセス定義 MATCH-P
39    出力定義  OT1
40
41      【場： IN1   のレコード   と IN2   のレコード無し   で】
42           MASTER-R2 を作成する , 但し対応移送は MASTER-R
43
44      【場： IN1   のレコード無し   と IN2   のレコード   で】
45           MASTER-R2 を作成する , 但し対応移送は TRANS-R
46
47      【場： IN1   のレコード   と IN2   のレコード   で】
48           MASTER-R2 を作成する , 但し対応移送は MASTER-R
49                                , 編集方法は ｻﾞﾝﾀﾞｶｺｳｼﾝ
50
51           編集方法 ｻﾞﾝﾀﾞｶｺｳｼﾝ
52              代入： 金額 OF MASTER-R2 = 金額 OF MASTER-R
53                                       + 金額 OF TRANS-R;
54
55           入力時処理 売上げﾃﾞｰﾀ :
56           レコード条件は      商品ｺｰﾄﾞ OF TRANS-R NOT = ZEROS
57
58    ファイル物理定義
59    プログラム情報
60     ファイル情報 商品ﾏｽﾀ
61          :DISC,ST ,10 ,OLD-MASTER
62          FILE-ID:'S-MASTER'
63     ファイル情報 売上げﾃﾞｰﾀ
64          :DISC,ST ,10 ,売上げﾃﾞｰﾀ
65          FILE-ID:'S-TRANS'
66     ファイル情報 OUT1
67          :DISC,OM ,50 ,NEW-MASTER
68          FILE-ID:'S-MASTER'
```

図6　生成されたコードをPASE 1100で日本語化したリスト

Fig. 6　A text translated into Japanese by PASE1100

ディクショナリの定義内容を値とするディクショナリ参照パラメタである.

ディクショナリ参照パラメタの値は,もちろん,このマクロが展開される時点のディクショナリで決まることになる.このようなパラメタによって,TSX-PASE のマクロは固定的なものでなく,テキストから独立した存在となっている.

図5は,例として示した図2のモジュール定義から生成されたコードである.ただしコードは PASE 1100 言語で書かれているが,この後 PASE 1100 プロセッサを通して COBOL コードが得られる.

図6は,PASE 1100 プロセッサが作る日本語化リストである.

5. おわりに

TSX-PASE は実業務システムで試行されているが,その効果を数字で示すことができるまでには至っていない.PASE 1100 言語による仕様化技法とスケルトンを組み合わせることによって,個々のプログラムの仕様記述はまさに段落単位の独立した記述になり,これは記述の容易さ,テストの容易さをもたらしている.

ディクショナリは作るだけでなく,使う手だて,しかも機械的な手だてがあって初めて生かされてくる.そのような手だてがないディクショナリは正しく維持し続ける魅力を欠いたものになる.

ディクショナリを介在させた開発環境も,ソフトウェアの部品化と再利用も,至極当然のことのように思う.TSX 1100 も TSX-PASE もまだまだ十分ではないが,新しいソフトウェア開発のパラダイムへの出発と思っている.いま TSX 1100 はディクショナリをどう使うのが良いか,どう使うことができるのかという点が弱い.これは,TSX 1100 がソフトウェア開発の方法,技法のどれかに片寄ることを避けてきたためであるが,ディクショナリが真価を発揮するのは,何らかの良い技法にそったものになった時であろう.今後の TSX 1100 に期待するところである.

最後に,本ソフトウェア開発の機会を与え,さらに試行していただいた共栄火災海上相互保険会社システム部の方々,および開発に当たって多大の援助をいただいた小池・山下両氏をはじめ,関係者の方々に感謝の意を表したい.

参考文献 [1] UNISYS シリーズ 2200・1100 TSX-PASE 解説書,資料コード 481265408-0,日本ユニシス㈱.

執筆者紹介 長谷川 邦夫 (Kunio Hasegawa)

昭和47年日本ユニシス㈱入社.事務処理分野での受託システム開発,近年は PASE 1100, TSX 1100 の開発に従事.現在,システム企画本部ソフトウェア生産技術一部に所属.

住宅CADシステムにおける部材決定のためのデータ構造

Data Structure for Selecting Component Parts in the Housing-CAD System

本間一郎

要　約　当社の住宅CADは，一般住宅の設計・製図・積算および躯体/部材展開を行う一貫システムである．このシステムでは，データベース中に，家の持つ各種情報から，3次元表現した家モデルを創成し，その情報をもとに図面の出力や見積りなどを行う．

家モデルを構成するデータおよび処理手順の種類は多く，追加・変更もある．したがって，当システムの開発では，まずこれらのデータの抽象化を行い，この抽象化されたデータ構造を扱えるようにプログラム作成を行った．その抽象レベル以上のデータ構造の追加や変更がないかぎり，データの変更がプログラムの修正や，新規開発にはおよばないからである．

住宅CADシステムの一部である部材展開は，家モデルをもとに出荷する部材を決定する．

本稿では，この部材決定の際に影響のある各種条件と，それによって決まる部材との関係を定義したデータ構造について紹介する．部材展開では，これによりデータ抽象化が実現されている．

Abstract　The Housing-CAD is an integrated total system which supports housing designs, provides drawings and estimates housing costs in addition to selecting materials.

The system generates three-dimensional housing models in its database, produces drawings and estimates building expenses based on those models.

There are enormous varieties of data consisting of a housing model, and so are there varieties of processes. Also, additions and modifications to the data and processes occur in a frequent manner. So, there was our first attempt at data abstraction, in our efforts to develop the system. This is because, once we provide such a program as can trast the abstract data structure, data modifications do not involve any changes in application programs, unless there are changes in data themselves above the level of the abstraction.

The material system, which is a part of the Housing-CAD system, picks out component parts through the use of housing models.

This report describes the data structure which assists in selecting parts. In one word, the data structure is so designed as to define relationships between requirements, which influence parts selection, and the parts selected. It realizes the data abstraction.

1. はじめに

一般に，プログラムは手続きとデータとから成る．したがって，手続きの構造と別にデータの構造が存在する．大雑把にいって，データ構造がある抽象レベルで変わらなければ，手続きの構造も変わらないといえる．本システムでは，データ部分を各種パラメータに至るまでなるべくプログラムの外に出すことにより，データの変更がただちにプログラムの手続きの変更にならないように考慮してきた．したがって，これ

らのデータ構造の中には各種のデータ項目の単なるよせ集めといったものもあり，すべてを紹介することはあまり意味がない．ここでは，住宅CADの部材展開(部材拾い)における部材決定のためのデータ構造について報告する．それは，このデータ構造が部材展開システムの中核であるばかりでなく，手続きとデータが良好な形で分離できた例と考えるからである．

2. システム概要

当社の住宅CADシステムは，一般住宅の設計・製図・積算および部材展開（部材拾い）を行うシステムである（図1）．本システムは，単に製図作業の省力化をするだけではない．顧客に対し，その場で図面や正確な見積りを提出したり，入力されている情報をもとに自動的に部材の展開を行って，工場や部材メーカへの発注までを行う一貫システムである[1]．

これを実現するために，データベース中に家のもつ情報を忠実に表現した家モデルを3次元で持つ．製図・積算および部材展開は，すべてこの家モデルをもとに処理する．したがって，設計の変更に対応した変更を簡単に行うことができる．なお，家モデルの構造について詳しくは参考文献[1]を，またその構造のより抽象的側面については，参考文献[2],[3]を参照されたい．以下では，本システムの設計・製図・積算および部材展開の各機能をサブシステムとよぶことにする．

家モデルを作る際に用いる家構成データは，これらのサブシステムとは別系列の標準品サブシステムで，創成・変更・削除を行う．標準品ファイルは，家構成データの

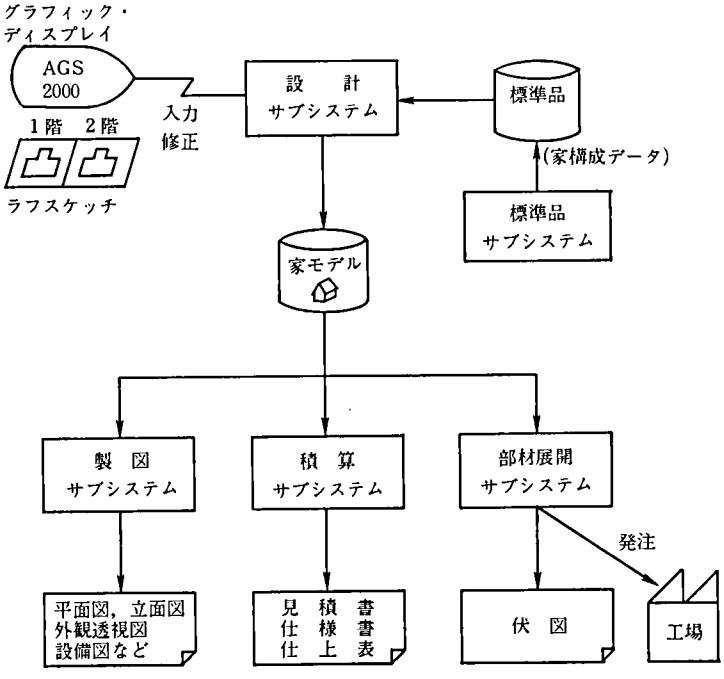


図1　システムの概要

Fig. 1　Outline of HCAD system

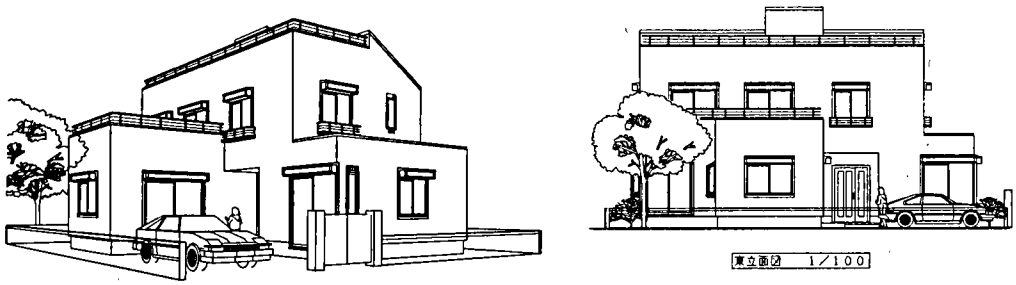


図 2 図面の出力例（その1）
Fig. 2 Example of output drawing (1)

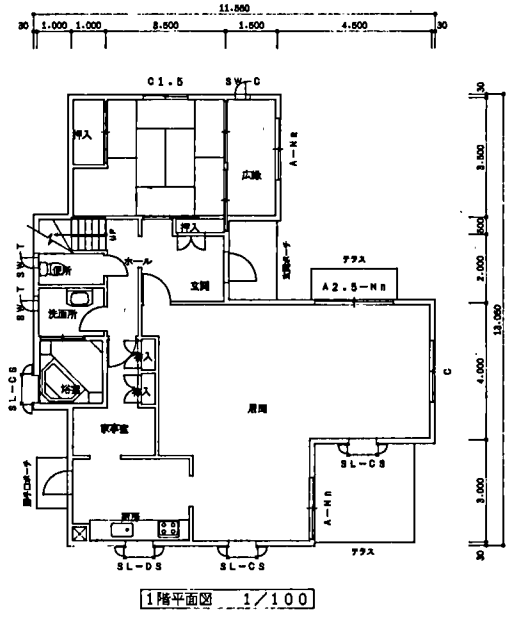


図 3 図面の出力例（その2）
Fig. 3 Example of output drawing (2)

データ構造を表現したデータベースであり，設計サブシステムをはじめ，製図・積算・部材展開の各サブシステムからも検索される．標準品のデータは，非形状（属性）と形状（図形）とに分かれる．非形状は，図形以外のデータすべてを含む．形状は，製図サブシステムで出力する図面や，部材展開サブシステムで出力する伏図で用いる図形を定義する．

これと別に，製図・積算・部材展開の各サブシステム固有のデータを含む各種テーブルがある．後述する部材展開サブシステムのコード決定テーブルもそのうちの一つである．これらのテーブルは標準品と別のデータベースとして持つが，標準品とまったく同じ方法で創成・変更・削除の操作ができる．

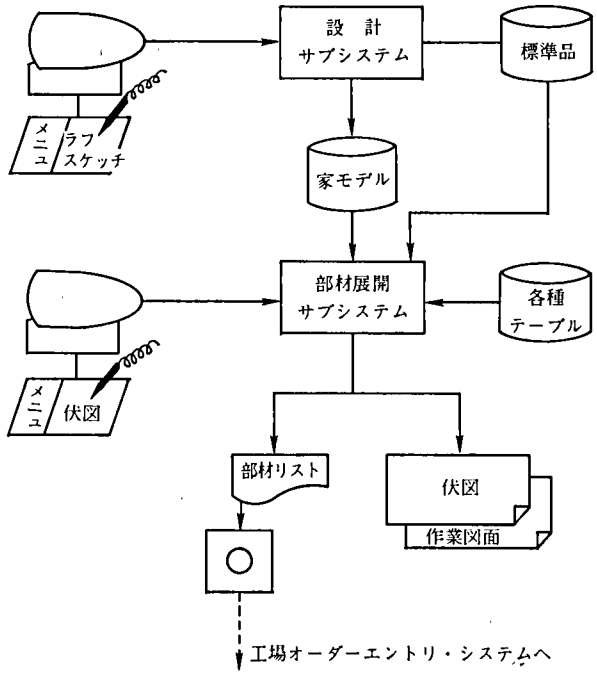


図 4　部材展開サブシステム概要
Fig. 4　Outline of the parts system

3. 部材展開サブシステム

3.1 処 理 概 要

部材展開サブシステムは，設計サブシステムによってつくられた家モデルをもとに，最少限の入力で自動的に工場出荷材を拾い出すシステムである（図4）．

ラフスケッチをもとに，設計サブシステムによってつくられた家モデルより，必要な情報を取り出し，伏図より入力された構造材をもとに，すべての工場出荷材を拾い出す．工場出荷材は，部材リストとして「工場オーダーエントリ・システム」（発注を行うシステム）に送られ，同時に伏図も工場の製図機に出力される．

現在扱っている部材の種類は，数百種に及び，鉄部材（軸・梁などの構造材，および金具類），外部付帯（ドア・窓・バルコニーなど），外装材（パネル・母屋・水切類など）に分類できる．

3.2 処 理 の 流 れ

部材展開サブシステムの処理の流れは，次のとおりである．

1) 部材の創成場所を求める．これには2種類ある．一つは，家モデルやすでに創成されている他の部材からは自動的に決められない，つまり創成場所の求め方が規則化しにくい部材で，これらの部材は操作により入力する．これらの部材が存在することが，部材展開サブシステム自体がCADシステムになっている理由である．このような部材は，このサブシステムが扱う部材の種類の中のごく一部で，軸・梁などの構造材を中心に存在している．

もう一つは，家モデルやすでに創成されている他の部材から自動的に決められ

る，つまり創成場所の求め方が規則化しやすい部材であって，大部分の部材はこの方法で創成場所を決めることができる．前者を創成，後者を自動創成とよんでいる．部材の種類ごとに，創成場所は異なる規則で求められる．

2) 創成される部材を決める．一つの種類の部材にもさまざまなものがあり，品番でそれを区別している．この品番を決めなければならない．品番と同時に品名が決まる．この過程については4章で述べる．

3) 部材の創成場所・品番・品名などの情報を部材ファイルに書き込む．
部材ファイルの構造は，次に示す理由から単純な木構造としている．

① 扱っている部材のうち，大部分は家モデルおよび入力された構造材から自動的に決めることができ，部材展開サブシステム独自の入力操作を必要とする部材はわずかであること．

② 相互に参照することが少ないこと．

③ 出力は部材リストおよび伏図であり，ほとんど逐次に処理すること．

4) 部材リスト，伏図の出力を行う．部材リスト，および伏図の出力は後工程であるということができる．部材ファイルに書き込まれた創成場所・品名・品番などの情報をもとに出力する．部材リストは，印書と同時にディスケットに出力され，

```
W787  A04 [    1] (200L) 0.5NF2200ドアU1                F A04
W395  A04 [    1] (200L) 0.5NFG                         F A04
W302  A04 [    1] (200L) 0.5NFG0.5                      F A04
W538  A04 [    2] (200L) r 0.5NFG0.5                    F A04
W264  A04 [    1] <4W> 1ND1                             F A04
W903  A04 [    2] <4W> 2NE1                             F A04
```

図 5 部材リストの出力例

Fig. 5 Example of output parts list

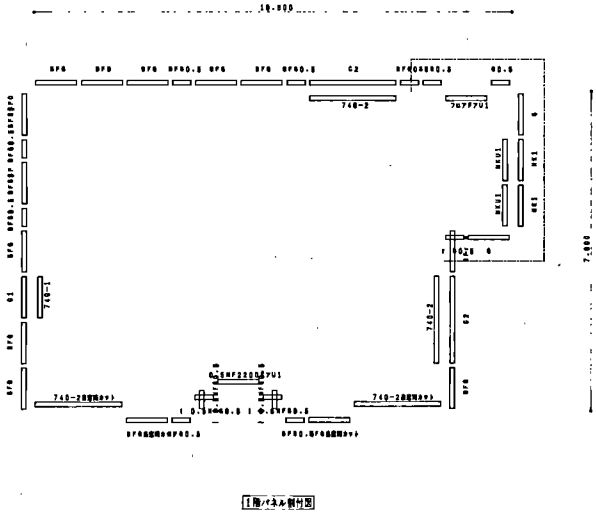


図 6 伏図の出力例（その1）

Fig. 6 Example of output framing plan (1)

オーダーエントリ・システムへの入力となる（図5，図6，図7）．

サブシステムとしての基本処理は以上のとおりであるが，本番での運用時には通常次に示す操作が行われる．

① 部材操作で入力する部材について入力を行い，その部材に関する伏図を出力する（図6，図7）．

② 伏図により入力に誤りがないことを確認し，その後，一括（バッチ）処理により自動創成の部材を作り出す．その後，最終的な伏図と部材リストを出力する．もし，入力に誤りがあれば再び入力しなおす．

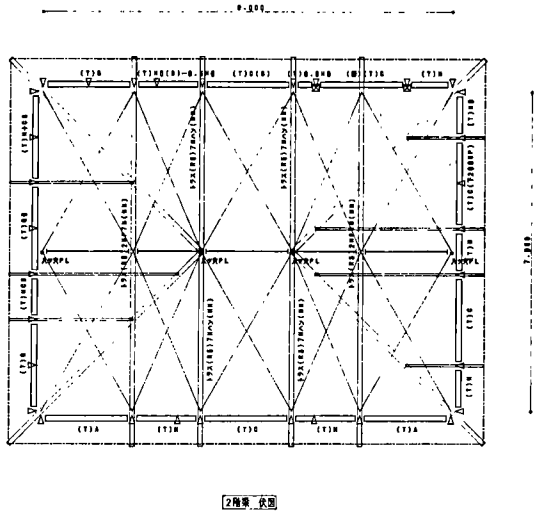


図 7　伏図の出力例（その2）

Fig. 7　Example of output framing plan（2）

4. 部材の決定のしかた

本章では，部材の決定方法について一般論からはじめ，次に本部材展開サブシステムの扱いについて述べる．

1) まず，部材の決定のしかたを個々の家には依存しない一般的な規則として取り出さなければならない．

2) この規則は，一般に部材の創成場所において，

$$P_1 \cap P_2 \cap \cdots \cap P_n \supset Q \qquad (4\text{-}1)$$

の形の命題で表すことができる．ここで，A∩BはAかつBを，A⊃BはAならばBを意味するものとする．（この「AならばB」の記法は，集合論の記法[4]から見るとあべこべのように見えるが，ここではMannaの本[5]などの述語演算の記法に従った．）

また，$P_1, P_2, \cdots, P_n$ は部材を決めるための各種条件が満たされていることを示す命題であり，Qは「部材 q（品名・品番）を創成する」という命題である．以下では，各 P_i を単に条件ということにする．条件には，たとえば「屋根の勾配は4寸である．」とか，「下階の屋根材は瓦である」とかのように専門家以外にもわか

りやすいものからもっと複雑なもの，そのメーカ固有のものなど，さまざまある．

以下では $P_1 \cap P_2 \cap \cdots \cap P_n$ を $\bigcap_{i=1,n} P_i$ と書くことにする．

3) 取り出された規則を計算機で処理するために，これらの命題を計算機システム上に記述が可能な表現で1対1に対応させる．条件 P_i の計算機システム上で表現された値を a_{ij} とし，その変数を x_i で表すと，P_i は $x_i = a_{ij}$ と表現できる．したがって，上記の命題(4-1)は，

$$\bigcap_{i=1,n}(x_i = a_{ij}) \supset (q = q_j) \tag{4-2}$$

の形に書き直すことができる．ここで q_j は，決定された部材の品名・品番を示すものとする．x_i がとる値 a_{ij} を条件値とよぶ．型は，整数・実数・文字（ASCII・漢字コード・漢字セット・混合コード）などである．種類の異なる部材は，一般には異なる条件で決定される．これは，次の形に書ける．

$$\bigcap_{i=1,n'}(x_i' = a_{ij}') \supset (q' = q_j')$$

4) このように表現すると，部材の品番・品名 q_j は，x_i の部材の創成場所における条件値の組

$$(a_{1j}, a_{2j}, \cdots, a_{nj}) \tag{4-3}$$

および条件値と品番・品名の組（対応表）

$$\begin{pmatrix} a_{11}, & a_{21}, & a_{31}, & \cdots, & a_{n1}, & q_1 \\ a_{12}, & a_{22}, & a_{32}, & \cdots, & a_{n2}, & q_2 \\ & & \vdots & & & \\ a_{1m}, & a_{2m}, & a_{3m}, & \cdots, & a_{nm}, & q_m \end{pmatrix} \tag{4-4}$$

を与えれば決めることができる．ここで，対応表(4-4)は個々の家にはよらず，あらかじめ用意できるものである．こうして部材の決定過程は，条件がどのような性質を持つかによらず，形式的に取り扱うことができる．(4-3)の条件値の組に各項目が一致する行を，対応表(4-4)の中から検索すればよいわけである．

ところが，これをこのまま実用にするには少し不都合がある．対応表(4-4)の行数が膨大なものになることである．i 番目の条件 x_i が，とりうる条件値の数を ν_i とするなら，行数 m は，

$$m = \nu_1 \times \nu_2 \times \cdots \times \nu_n$$

となる．さいわい，一般に部材の品番・品名の数は必ずしもこの m の数だけあるわけではない．たとえば，ある品番・品名 q_j の部材については条件 x_i の条件値がどうであれ，q_j に決まる，というようなことがある．そのために，対応表(4-4)の項目中では，対応する条件の(4-3)の中での条件値が何であっても，一致したとみなす書き方を定義する．それを－で表す．これを用いると，今の例は，

$$\begin{pmatrix} a_{11}, & a_{21}, & a_{31}, & \cdots, & a_{i1}, & \cdots, & a_{n1}, & q_1 \\ a_{12}, & a_{22}, & a_{32}, & \cdots, & a_{i2}, & \cdots, & a_{n2}, & q_2 \\ & & & \vdots & & & & \\ a_{1j}, & a_{2j}, & a_{3j}, & \cdots, & — , & \cdots, & a_{nj}, & q_j \\ & & & \vdots & & & & \\ a_{1m}, & a_{2m}, & a_{3m}, & \cdots, & a_{im}, & \cdots, & a_{nm}, & q_m \end{pmatrix}$$

と表せる．また，ある q_j について x_i 以外の条件については同じ場合，$x_i=a_{ij}$ ならば $q=q_j$ に，$x_i \neq a_{ij}$ ならば $q=q_{j+1}$ になるという例でも，この一の書き方を用いることができる．

$$\begin{pmatrix} a_{11}, & a_{21}, & a_{31}, & \cdots, & a_{i1}, & \cdots, & a_{n1}, & q_1 \\ a_{12}, & a_{22}, & a_{32}, & \cdots, & a_{i2}, & \cdots, & a_{n2}, & q_2 \\ & \vdots & & & & & & \\ a_{1j}, & a_{2j}, & a_{3j}, & \cdots, & a_{ij}, & \cdots, & a_{nj}, & q_j \\ a_{1j+1}, & a_{2j+1}, & a_{3j+1}, & \cdots, & — , & \cdots, & a_{nj+1}, & q_{j+1} \\ & \vdots & & & & & & \\ a_{1m}, & a_{2m}, & a_{3m}, & \cdots, & a_{im}, & \cdots, & a_{nm}, & q_m \end{pmatrix}$$

ただし $k \neq i$ のとき，$a_{kj}=a_{kj+1}$ とする．もっともこの場合，条件値の組と対応表の突き合わせは，対応表の上から順ぐりに行うことが前提となる．しかし実用上は，このように書ける方が都合のよいことが多いので，このサブシステムでは上から順に突き合わせている．

このように，部材の決定過程が形式的に行えるということは，部材の種類によらない一般化ができるということを意味する．以下にそれを示す．多くの条件は複数の種類の部材の決定に用いる．すべての条件を重複を省き，順に並べて

$$\xi_1, \xi_2, \cdots, \xi_m$$

とする．このときの条件を表す変数 ξ_i の添字 i を，条件番号とよんでいる．(実際には，条件の性質ごとに分類し，将来の追加のために間をあけて番号をふっているが，ここの説明ではそれは本質的な問題ではない．)

こうして条件番号をふると，(4-1)は次のように書き直せる．

$$\bigcap_{k=1,n}(\xi_{i_k}=a_{kj}) \supset (q=q_j) \tag{4-5}$$

このときコード決定の過程は，次の2段階に分けられる．

① 創成場所を求め，条件値の組（コード決定条件リストとよぶ）

$$(\xi_{i_1}, \xi_{i_2}, \cdots, \xi_{i_n})=(a_{1j}, a_{2j}, \cdots, a_{nj}) \tag{4-6}$$

を求める．これは，条件番号の組

$$(i_1, i_2, \cdots, i_n) \tag{4-7}$$

を与えれば求められるが，条件値を求める際に部材ごとの固有の事情が反映する条件と，部材によらず共通に得られる条件とがある．

② コード決定条件リスト，および条件値と品番・品名の対応表(4-4)を与えると，部材の決定ができる．これは，すでに述べたようにまったく形式的に行えるのであって部材の種類によらない．ただ，部材ごとに条件番号の組(4-7)と，条件値と品番・品名の対応表(4-4)をデータとして用意しておけばよいのである．本サブシステムでは，これをコード決定テーブルとよび，(4-7)を見出し部，(4-4)をレコード部として分けて持つようにしている（5章参照）．

5) このような方法をとることにより，部材を決定する条件の変更に対しては，すでにある条件を使うかぎりプログラムの変更をともなわず，データ（コード決定テーブル）の変更で対処することができる．一方，新たな部材の種類を扱えるよ

うにするための作業は，次のように分けられる．

① その部材の種類の命令プログラム（創成場所を求めることが主要な内容）を作る．

② すでに扱っている部材の決定には，用いられたことのない新規の条件処理プログラムを作る．

作るプログラムそのものも，かなり定型的なものとすることができる．

5. 部材決定のためのデータ構造

この章では，4章のような考え方からつくられたコード決定テーブルのデータ構造を紹介する（図8）．

次に示すデータ構造の中で，構成要素の最小の単位となるのが末端データ型であり属性を持つ．また，その中で用いられている記号の意味は次のとおりとする．

{ } ：非末端データ型

・ ：および

| ：または

＝ ：左辺は末端データ型，右辺が属性

＋ ：1回以上の繰り返し

〔 〕：省略を許されるデータ

$n_1 \| n_2$ ：n_1 回以上，n_2 回以下の繰り返し．

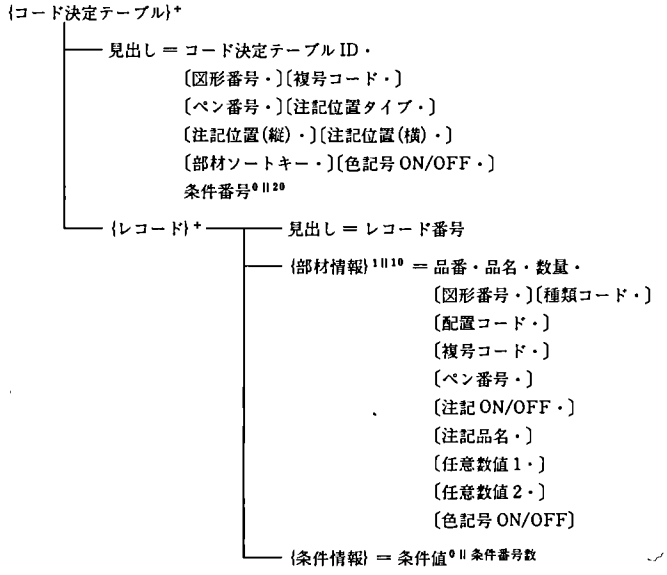


図 8 部材展開コード決定テーブルのデータ構造

Fig. 8 Data structure of the table for selecting parts

6. お わ り に

この報告の中心は，各条件の諸特性を捨象し，形式的な取扱いをすることにより，処理の一般化が可能になったところにある．手続きとデータ構造を抽象化してとらえ

ることの有効性が示された．条件となる命題を条件に対応させるのは，むずかしいことではない．順に並べて番号をふるという奥の手がある．

ここで示した命題の取扱い方法は十分形式的であって，真であるか，偽であるか決めることのできない命題[6][7]さえ，条件として表現できるほどである．もっとも，本サブシステムが現実の部材の決定の忠実なモデル化である限り，そのような条件が現れることはない．ここで報告したようなコード決定の方法は，他の分野にも適用できる．

また，現在の部材展開サブシステムでは，創成場所については各命令ごとに求めているが，創成場所を求める規則を手続きと分けて表現するならば，仕様変更に対する対処は，よりたやすく行える．さらに，本システムの枠組工法への適用では構造の自動創成[8]も試みられており，本サブシステムで現在，創成場所の求め方が規則化しにくく，操作で入力している部材の自動化も今後の課題としてあげられる．これには，ある種の知識ベースシステム[9]の適用が考えられる．

今回紹介した部材の決定そのものについても，知識ベースシステムのために開発された突合せの方法[10]を参考に，効率をよくしていきたい．

最後に本稿をまとめるに当たり，御協力いただいた積水ハウス(株)今井一延氏，藤川博氏，および関係者各位に深く感謝の意を表したい．

参考文献

［1］ 篠田博水，「3次元家モデルに基づく住宅設計一貫システム」，技報，日本ユニバック(株)，第8号，1985，pp. 22～36.

［2］ 柳生孝昭，「CADのための data metamodel」，技報，日本ユニバック(株)，第6号，1984，pp. 1～34.

［3］ 柳生孝昭，「CADにおける databaseの問題(10)——metamodel論を中心として」，PIXEL，No. 31，1985，p. 188.

［4］ 赤攝也，集合論入門，新数学シリーズ，1，培風館，1957，pp. 13～14.

［5］ Z. Manna, *Mathematical Theory of Computation*, McGraw-Hill Inc., 1974, p. 77.

［6］ K. Gödel, "*Über formal unentscheidbare Sätze der Principia Mathematica und verwandter System I.*", *Monatshefte für Mathematik und Physik*, 第38巻，1931，pp. 173～198，邦訳は，ゲーデル，"「プリンキピア・マテマティカ」およびそれに関連する体系の決定不可能な命題について，I"，広瀬健，横田一正訳，広瀬・横田著，ゲーデルの世界，海鳴社，1985，附録.

［7］ D. R. Hochstadter,「ゲーデル，エッシャー，バッハ——あるいは不思議の環」，野崎昭弘，はやしはじめ，柳瀬尚紀訳，白揚社，1985.

［8］ 渡辺寛，「枠組壁工法住宅における構造の自動設計システム」，技報，日本ユニバック(株)，第16号，1988，pp. 43～60.

［9］ 上野晴樹，石塚満共編，知識の表現と利用，知識工学構座，2，オーム社，1987.

［10］ C. L. Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", Artificial Intelligence, 19, 1982, pp. 17～37.

執筆者紹介　本　間　一　郎 (Ichiro Honma)

昭和24年生，49年早稲田大学大学院理工学研究科修士課程修了，原子核物理学理論専攻．同年日本ユニシス(株)入社，言語プロセッサ，グラフィックソフトウェアなどを担当，各種CADシステム開発を経て，現在，ハウジング・システム部所属．日本物理学会会員．

パソコン通信におけるFTAMの実装

Incorporation of FTAM into PC Communications Software Package

広田雅史

要約 JUST*-PC方式（郵政省パーソナル・コンピュータ通信装置推奨通信方式）対応パソコン通信パッケージPCHOSTを開発した．PCHOSTは，UNISYSシリーズ1100/2200をホストとしたパソコン通信ネットワーク構築機能——トランザクション転送・ファイル転送・バッチ起動・電子メール／電子掲示板アクセス機能——を提供するソフトウェア・パッケージである．

このファイル転送機能のプロトコルとしてOSI** FTAM***（DIS8571準拠）を適用した．FTAMはファイル転送，アクセスおよび管理のサービスとプロトコルを定義している．PCHOSTでは，パソコン側の処理能力およびJUST-PC準拠のセッション層の機能を考慮して，FTAMのサポート範囲を以下のように規定し実装した．

1） サービスクラス：転送・管理クラス
2） サービスレベル：利用者訂正型
3） 仮想ファイル　：非構造ファイル
4） チェックポイント・ウィンドウサイズ：1～16（標準：5）

また，付加機能として，共有ファイル・ファイル照会・ホスト起動ファイル転送の各機能を開発した．

本稿では，パッケージ概要・ファイル転送サービス機能・効率および今後の動向について述べている．

Abstract A personal computer communications software package (PCHOST) was developed to meet requirements called for in the JUST-PC method. PCHOST provides functions to transfer transactions and files, kick off batch programs, and gain access to the mailbox/and bulletin board system for the UNISYS Series 1100/2200 computer.

OSI FTAM (DIS 8571-based) was adopted as the file transfer protocol for the package. FTAM defines the service and protocol of file transfer, access and management. In the case of PCHOST, current personal computer capabilities and JUST-PC session layer functions were taken into consideration for the determination of the FTAM support range, that is as follows:

1) service class : transfer and management class
2) service level : user correctable
3) virtual file : unstructured file
4) checkpoint window size : 1 to 16 (standard: 5)

Also, such functions as common file management, file inquiry and host initiated file transfer were newly developed, and have been made available.

This paper describes the overview of PCHOST, its file transfer service, performance and future development

* JUST : Japanese Unified Standard of Telecommunication
** OSI : Open Systems Interconnection, 開放型システム間相互接続
*** FTAM : File Transfer, Access and Management, ファイル転送管理

1. はじめに

新しいコミュニケーション・メディアとして,パソコン通信が注目を集めている．パソコン通信は，マスメディアの放送性と電話の双方向性を同時に兼ね備えたコミュニケーション手段として期待されている．

パソコン通信を高度情報化社会の主要な基盤とすべく郵政省は，1984年に「パーソナル・コンピュータ通信装置推奨通信方式」(JUST-PC方式）を告示した．

これを受けて，NTTでは1985年10月から約1年間の公開実験を行った後，NTTPCコミュニケーションズが事業化し，1986年11月から「NTTPCネットワーク」として商用サービスを開始した．その後アクセス・ポイントの拡大や，サービス機能の充実など着々とネットワークの拡充が行われてきている．

日本ユニシスでは，多機能デスクステーションDS7による実験参加を経て，シリーズ1100/2200による初の情報センタ接続パッケージPCHOSTレベル1を開発し，1987年11月に商品化を行った．さらに,電子メール／電子掲示板接続の機能を追加したPCHOSTレベル2の提供を1988年6月より開始した．

2. プロダクト概要

PCHOSTは，JUST-PC対応パソコンをDCP*通信制御装置経由でシリーズ1100/2200に接続し，①情報センタ接続サービス（トランザクション転送，ファイル転送，バッチ起動)，②電子メールサービス，③電子掲示板サービス，等の業務システム構築を支援するために開発されたソフトウェア・パッケージである．

業務システム構築のためにPCHOST支援ライブラリが提供され，使用者は接続手順，転送プロトコル，出力待行列制御，ネットワーク運用を意識することなく業務システムの構築に専念することができる．

PCHOSTの特徴と機能を次に示す．

1） ネットワーク機能

① JUST-PC対応パソコンを次のネットワーク形態で接続することにより，パソコン情報センタ（ホストを中核としたパソコン・ネットワーク）を構築することができる．

- NTTPCネットワークを経由した「情報センタ接続」
- 標準パソコン手順，第2種パケット交換サービスによる「直接接続」
- NTTPCネットワークを利用した「電子メール／電子掲示板接続」

② JUST-PCに準拠したセッション制御(ログオン／ログオフ／異常切断等)を行うパソコン通信専用のソフトウェア（PCCSU**）をもち，PCHOSTコントロール・スケジュールやパソコン情報センタとしての運用サービス機能（センタ業務開始／終了／強制終了等）を提供する．

③ 最大10万IDの利用者情報の管理機能，および最大1000台同時稼働可能な入出力コントロール機能（待行列制御）をもつ．

2） サービス機能

---

* DCP：Distributed Communication Processor, 通信制御装置

** PCCSU：Personal Computer Communication System User

① 1通信文最大10キロバイトのデータの受け渡しをするトランザクション転送サービス機能を提供する．トランザクション転送には，TPS*インタフェースとMAPPERラン**インタフェースがある．

② パソコン—ホスト間やホスト—ホスト間で，最大2メガバイトのファイルを受け渡しするファイル転送サービス機能を提供する．ファイル集配信，ホスト起動転送，集信ファイル即時引き渡し，共有ファイル配信，ファイル情報検索などのサービス機能がある．

③ パソコンからホストに登録されたバッチランを起動するバッチ起動サービスを提供する．定型バッチラン起動や入力データ付きバッチラン起動などのサービス機能がある．

④ メール書き込み／読み込み，および掲示板書き込み／消去のサービス機能を提供する．メールサービスでは，最大30宛先までの同報出力および最大1か月後までの配送時刻指定が可能である．メール／掲示板サービスを利用してホストシステムの運用時間帯にとらわれない蓄積形のテキスト（ファイル）転送を実現することができる．

⑤ 情報センタ運用に必要なネットワーク管理サービスを提供する．センタ業務開始／終了／強制終了，パソコン強制終了，ログ運用，ネットワーク状況表示，コンソール起動TPSスケジュール，登録TPSスケジュール，課金情報採取，ネットワーク障害管理などの機能がある．

3） プログラム・インタフェース……使用者プログラムからPCHOSTサービス機能を容易に使えるように，トランザクション転送，ファイル転送，運用管理の使用者支援ライブラリが提供される．

4） ユーティリティ……システム運用に必要な利用者情報管理，ログ情報ファイル支援，課金情報支援，ネットワーク運用支援などのユーティリティを提供する．

2.1 プロトコル構成

PCHOSTのインタフェースは，CCITT, ISOの国際標準であるOSIの7階層モデル（図1）のプロトコルにより構成され，表1のようになっている．

6～7層はFTAMおよびJUST-MHS***（郵政省電子メール通信方式）に準じており，1～5層がJUST-PCに準拠している．

プロダクトへのマッピングは，ホスト側では6～7層をPCHOSTコントロールおよびサービスTPS，5層をPCCSU，1～4層をPCHOST　THが受けもち，パソコン側では6～7層をPCHOST-PCおよびMHS-PC，1～5層をパソコン通信アダプタが受けもっている．

2.2 ネットワーク構成

PCHOSTのネットワーク環境は，パソコン—ホスト間では，①NTTPCネットワーク経由，②電話網—DDXパケット網経由（第2種パケット），③構内電話網—PAD経由の接続形態がある．ホスト—ホスト間では，上記3種類の形態に加えて，④専用

---

* TPS：Transaction Programming System，オンライン・システムにおけるトランザクション処理を行う使用者プログラム

** MAPPERラン：UNISYSの利用部門向け第四世代言語MAPPERで組まれた使用者プログラム

*** MHS：Message Handling System，メッセージ通信処理システム

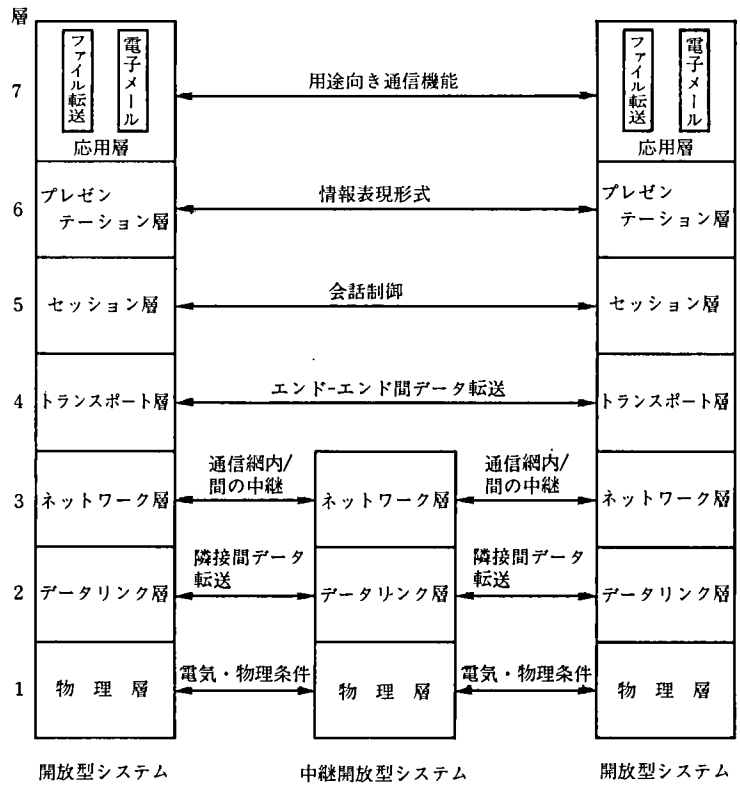


図1　OSI の7階層モデル

Fig. 1　OSI 7 layer model

表1　プロトコル構成

Table 1　Protocol construction

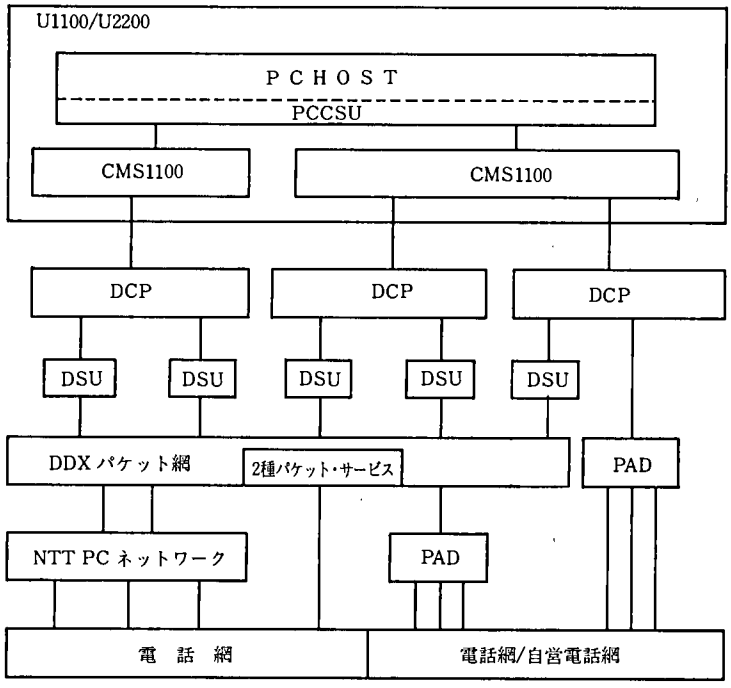
層	CCITT/ISO標準	
7	X. 411, X. 420, DIS 8571	
6	X. 409	
5	X. 225（カーネル＋全2重）	
4	X. 224（クラス0）	
3	T. 70（CSDN）	X. 25（76/80）
2	T. 71（LAPX）	X. 25（LAPB）
1	V. 27 ter	X. 21
網	電話網	パケット網

線経由接続が支援される．一つのシステムにこれらの接続形態を共存させることができ，接続されるパソコンやホストの設置場所，利用形態，通信量に応じて適切なネットワークを選択できる．

PCHOST のネットワーク構成例を図2に示す．

2.3　ソフトウェア構成

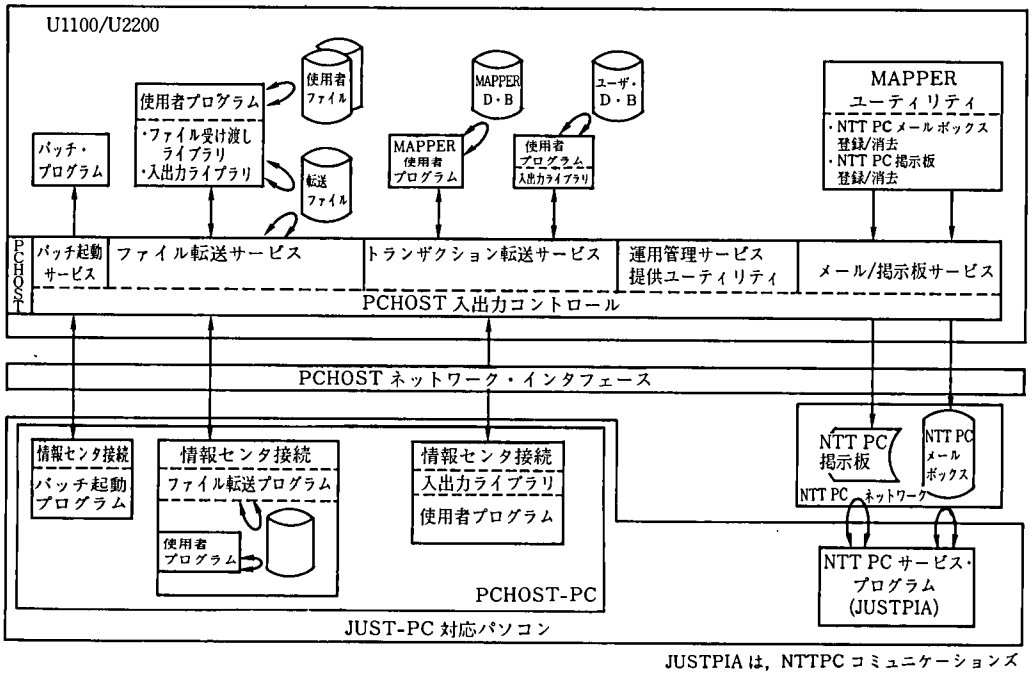
PCHOST のソフトウェアは，OSI 上位層の機能性をもち，セッション制御を行う PCCSU，アソシエーション制御およびネットワーク管理を行う PCHOST コントロール，アプリケーション・サービスを提供する PCHOST サービスに分類され，コモンパ



CMS1100 : Communication Management System for Series 1100
DSU : Digital Service Unit, 回線終端装置
PAD : Packet Assembly Disassembly, パケット組立て・分解装置

図2 ネットワーク構成例

Fig. 2 Network configuration



JUSTPIA は, NTTPC コミュニケーションズの登録商標です.

図3 ソフトウェア構成

Fig. 3 Software configuration

ンク*，TPS，オンラインバッチ**，ライブラリから構成される．ファイル転送，バッチ起動およびトランザクション転送のアプリケーション・サービスの切り換えは，PCHOST入力コントロールが行っている．

PCHOSTのソフトウェア構成を図3に示す．

これまでPCHOSTのパッケージ概要について述べた．次にOSI対応という観点から，PCHOSTの主要機能の一つであるファイル転送システムについてさらに詳しく述べる．

3. ファイル転送システム

PCHOSTのファイル転送機能の開発に当たり，FTAM以外に全銀協プロトコルやJ手順などの一般に公開されているいくつかのファイル転送手順を検討した．検討段階でFTAMは，第2次国際草案（DP）から，国際規格案（DIS）になりつつあり，実装仕様切り出しに十分な状況にあった．

全銀・J手順等は，伝送制御手順はBSC（二進データ同期通信）であり，JUST-PCはOSIの第5層まで規定されたものであるため，既存の上位アプリケーション・パッケージとのインタフェースを変更せずに使用することができないことが判明した．

したがってPCHOSTでは，ファイル転送も下位層同様ISO手順のFTAMの採用を決定した．また採用するに当たっての条件としては，①既存の手順のファイル・フォーマットはそのまま使用できるインタフェースをもつこと，②バッチ起動・トランザクション転送などの他のアプリケーションを同時に稼働させること，③プレゼンテーション/ACSE***/CCR****がなくてFTAMのサブセットをどう切り出すかということに絞られてきた．

本章では，FTAMの概要および実装上のポイントについて記述する．

3.1 モ　デ　ル

FTAMでは異機種間で汎用的な相互接続性をもたせるために，仮想ファイルおよび仮想ファイルストアと呼ばれるモデルを提示している．仮想ファイルは木構造の階層化されたファイル構造をもち，ファイル名・ファイル作成者IDなどの静的なファイル属性と，通信時の起動者IDなどの動的なアクティビティ属性をもつ．仮想ファイルストアは，複数の仮想ファイルを格納し，ファイルの選択・読み出し・書き込みなどの操作を行う．仮想ファイルの構成例を図4に示す．

ここで，DUはデータ単位と呼ばれるファイル中の識別可能な基本要素，FADUはファイルアクセス・データ単位と呼ばれるアクセス可能な単位を示している．

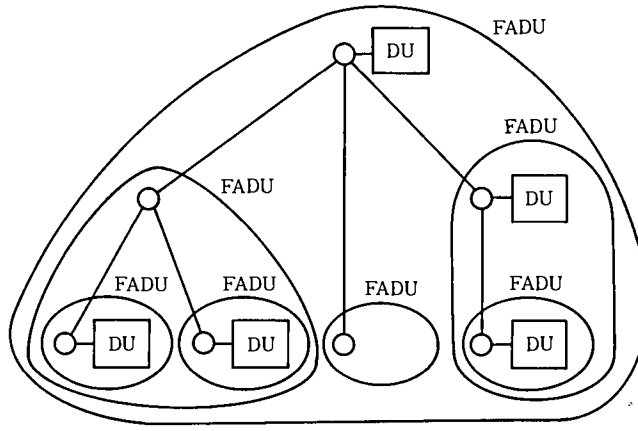
実装では，これらの仮想ファイル，仮想ファイルストアと実際のファイル，データ管理機能とのマッピングが必要となる．

3.2 サービスとプロトコル

FTAMでは前述のモデルを対象にして，サービスとプロトコルとを定義している．

---

* コモンバンク：Common Bank, OS 1100の管理する共有主記憶領域
** オンラインバッチ：Online Batch，オンライン・システムにおけるバッチ処理を行うプログラム
*** ACSE：Association Control Service Element，アソシエーション制御サービス要素
**** CCR：Commitment, Concurrency and Recovery，コミットメント/同時性/回復制御サービス要素



DU　：Data Unit, データ単位
FADU：File Access Data Unit, ファイルアクセス・データ単位

図4　仮想ファイルの構造例
Fig. 4　Virtual file model

サービスは，サービスプリミティブと呼ばれるファイル操作の基本要素から構成され，通信上サービスプリミティブに対応したFPDUと呼ばれるプロトコル・データ単位が規定されている．

FTAMプロトコル制御を司り，ファイルサービスを提供する主体を，FTAMプロトコル機械(FTAM PM)，ファイルサービスを利用する主体をFTAMサービスユーザという．このサービスは，通信路を通して確認や折衝が行われるため，エンドユーザ（端末操作員や使用者プログラム）が直接利用するのは得策ではない．エンドユーザは，ファイル転送を「依頼」という形で行い，FTAMサービスユーザは「依頼」を受けて転送処理を代行するプログラムとして構成されることが望ましい．

FTAMはファイル・サービスを論理的に9個の機能単位に分類し，FTAM PMがどの機能単位を支援するかの組み合わせによって，五つのサービスクラスを定義している．また，障害時の回復をファイルサービスユーザのレベルで行うかどうかによって，利用者訂正型または高信頼型の二つのサービスレベルを定義している．サービスクラスとサービスレベルは，ファイル転送初期化時に折衝され，応答側が起動側の要求レベルより低いレベルにあった場合でもエラーとはせず，自動的に起動側の機能をダウンさせて，ファイル転送を行うことを可能にしている．

3.3　実装仕様の検討

PCHOSTではFTAMの実装に当たり，次のような技術的問題点を検討した．

1）　サービスクラスの選択……転送，管理，転送・管理，アクセス，制限なしの五つのサービスクラスのうち，転送・管理クラスを選択した．ファイルアクセス機能単位の実装は将来の課題とした．

2）　回復および再開機能単位について……回復および再開機能単位の実装は，障害後のファイル途中からの再送を可能にする．しかし，ファイル名，チェックポイントなどの回復／再送に必要な情報（ドケット）を，ホスト側のコモンバンク上

に無制限にもちきれないこと，パソコン側では障害後のファイル途中からの再送が困難なことから実装からはずした．

3） サービスレベルの選択……回復および再開機能単位を実装しないことからサービスクラスは利用者訂正型とし，障害後のファイル再送はエンドユーザが行うものとした．

4） コンカレント処理……ファイル転送処理中に，トランザクション転送や，バッチ起動が同一パソコンからコンカレントに処理できるか否かを検討した．その結果，ホスト側はコンカレント可能な仕様にした．

5） 仮想ファイル……仮想ファイルは，効率を考慮してオンライン・ファイル上にもち，ファイル管理テーブルをコモンバンク上にもつことにした（この仮想ファイルを，「転送ファイル」と呼んでいる）．

6） ユーザ・インタフェース……使用者プログラムから転送ファイルを管理・読み出し・書き込みおよび発呼を行うインタフェースを検討し，支援ライブラリの形態で提供することにした．

7） 付加機能……FTAM には，定義されていないファイル情報検索機能やファイルの共有機能などの付加機能を検討し実装した．ファイル情報検索機能はパソコンからホストの転送ファイルを検索する機能で，共有ファイル機能は任意のパソコンから取り出すことのできるファイル配信機能である．

3.4 システム仕様

前節の技術的検討を踏まえて，実装仕様を決定し，システムとして必要な拡張を行った．こうして決定した PCHOST ファイル転送システムのシステム構成，および主要

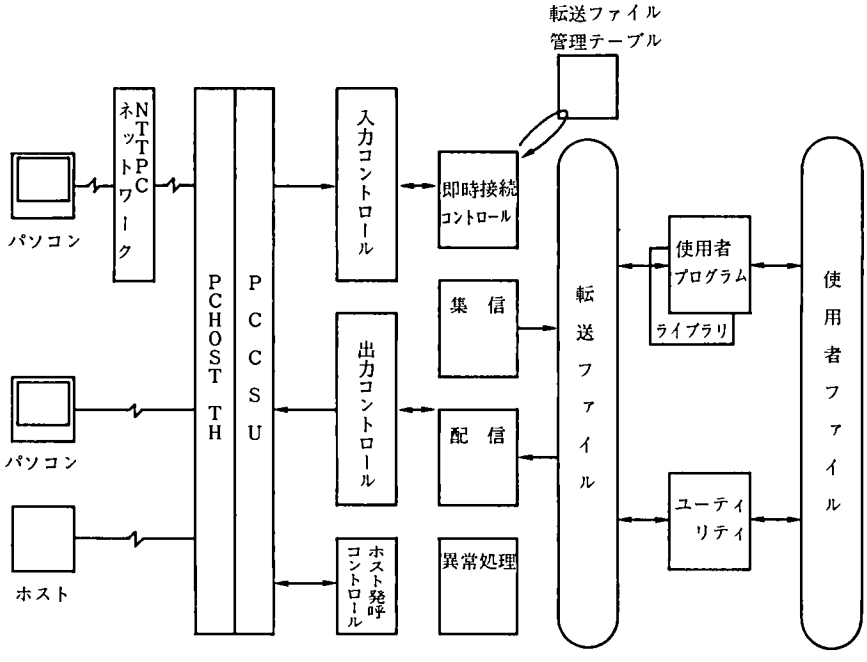


図5　ファイル転送システム構成

Fig. 5　File transfer system

表2 FTAM実装仕様
Table 2 FTAM implemented specification

項　　目	内　　容
FTAM主要仕様	
サービスクラス	転送・管理クラス
サービスレベル	利用者訂正型
転送ファイル	非構造
ルート　ノード	1個/ファイル
転送単位	ファイル単位
機能単位	カーネル＋読み出し＋書き込み＋ 限定付ファイル管理＋グループ化
ウィンドウサイズ	1～16（標準：5）
サービス項目	ファイル集配信サービス 共有ファイル配信サービス ファイル情報検索サービス ホスト起動ファイル集配信サービス 転送ファイル運用管理サービス ログ/課金サービス
起動側	パソコン，ホスト
応答側	ホスト，パソコン
転送ファイル数	2500 ファイル/システム
共有ファイル数	100 ファイル/システム
集配信ファイル数	各 25 ファイル/端末
転送ファイルサイズ	2 Mバイト/ファイル
転送ブロックサイズ	2048 バイト/ブロック

表3 ファイル転送コマンド一覧
Table 3 FTAM service commands

コマンド	内　　容
F-INITIALIZE	ファイル転送 初期化
F-TERMINATE	ファイル転送 終結
F-U-ABORT	ファイル転送 使用者異常終了
F-P-ABORT	ファイル転送 提供者異常終了
“A”	ファイル生成または選択とファイルオープン
“B”	ファイルクローズとファイル解放
“C”	ファイル選択とファイル削除
F-READ	ファイル読み込み
F-WRITE	ファイル書き込み
F-DATA	ファイルデータ
F-CHECK	チェック・ポイント
F-CANCEL	ファイル取り消し
F-DATAEND	ファイルデータ終了
F-TRANSFEREND	ファイル転送 終了

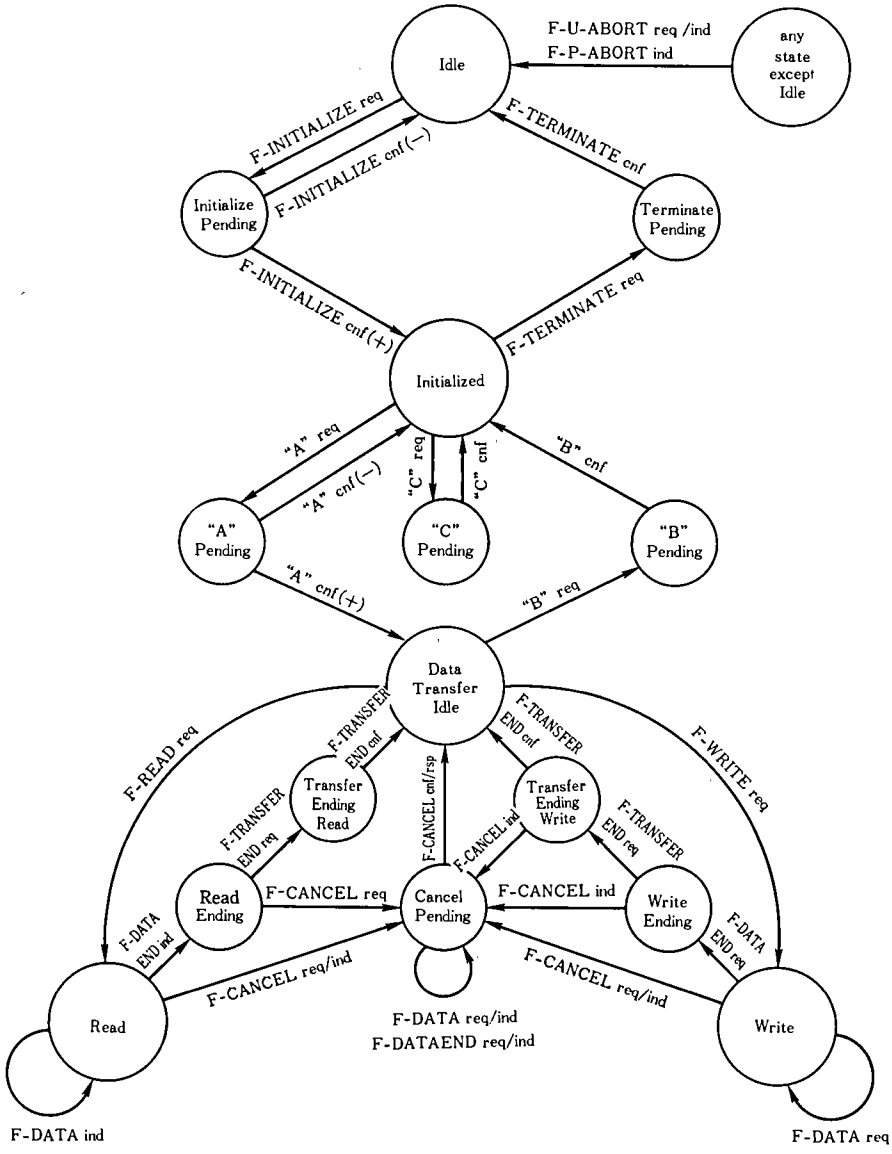


図 6　ファイル転送状態遷移図（起動例）

Fig. 6　File transfer state transition diagram (initiator)

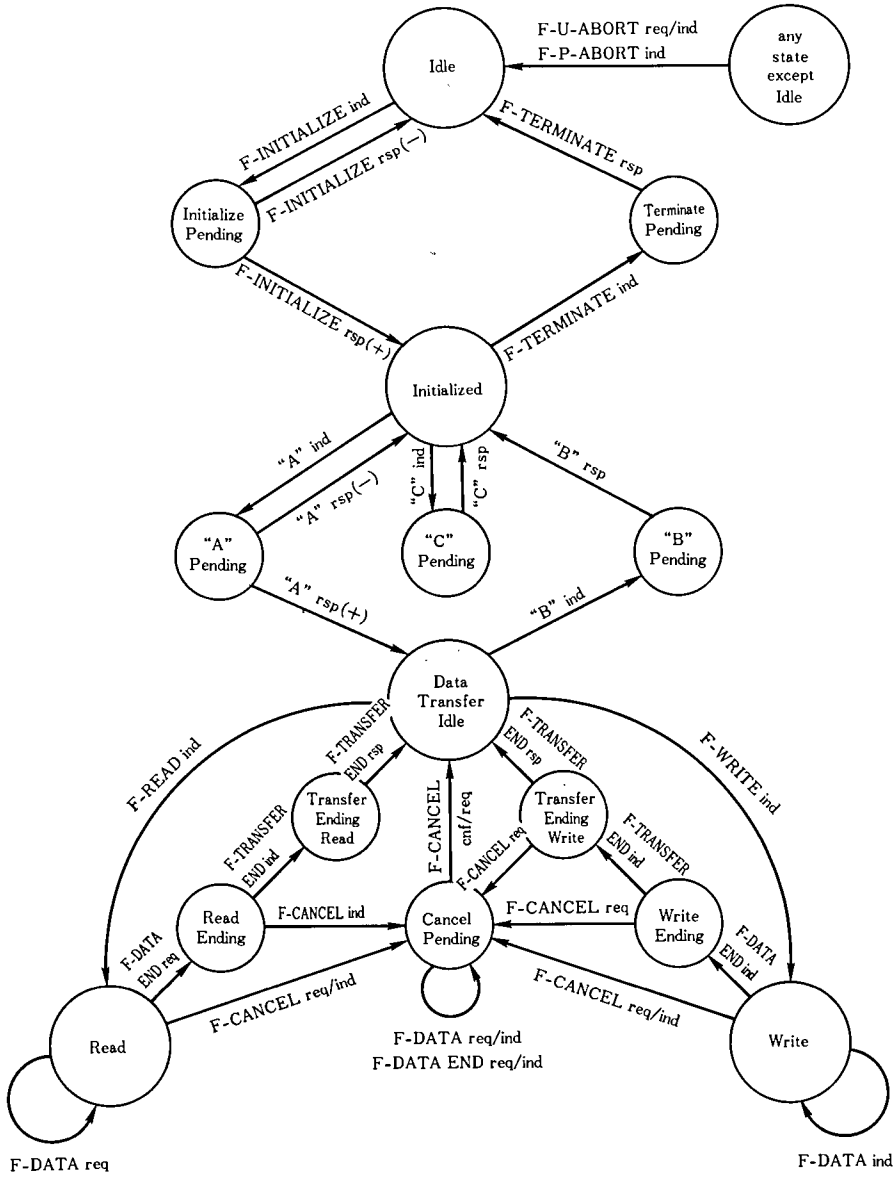


図7　ファイル転送状態遷移図（応答例）

Fig. 7　File transfer state transition diagram (responder)

諸元をそれぞれ図5および表2に示す．また実装した FTAM プロトコル機械の状態遷移図を，図6および図7に，状態遷移図で用いられているコマンドの一覧表を表3に示す．

3.5 効　　率

パソコン—ホスト間およびホスト—ホスト間折り返し（DDX パケット網経由）で，ファイル転送の効率測定を行った．ホスト—ホスト間の測定結果を以下に示す．

ファイル・サイズ	：80000 バイト①
ブロック・サイズ②	： 2000 バイト
ブロック数	： 40 ブロック
転送時間③	：1 分 49 秒
実効転送効率④	： 725 バイト/秒

① ホスト折り返しのため，80000×2 バイトが PCHOST の処理量である．
② 使用者データサイズ．通信上のブロックサイズは 2048 バイトである．
③ ファイル書き込み要求(F-WRITE req)送出から，転送終了確認(F-TRANSFER END cnf)受信までの経過時間．
④ ファイル・サイズ÷転送時間．

効率測定環境図および測定時のファイル転送ログを，それぞれ図8および図9に示す．

ファイル転送の効率としては，既存のファイル転送プログラムとほぼ同等の結果を得た．

4. お わ り に

FTAM は，汎用性・拡張性・包含性のあるシステム体系である．本章では，PCHOST での実装で実現できなかった課題，標準化・製品化の今後の動向について述べる．

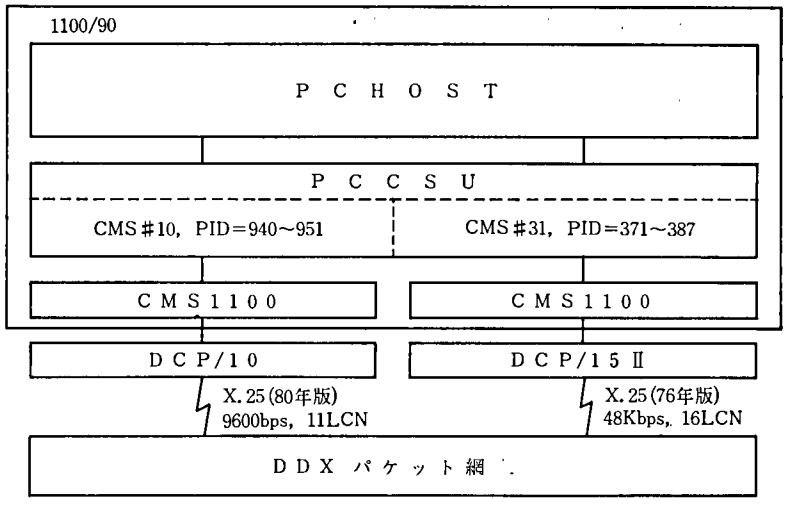


図8　効率測定環境図

Fig. 8　Performance measurement environment

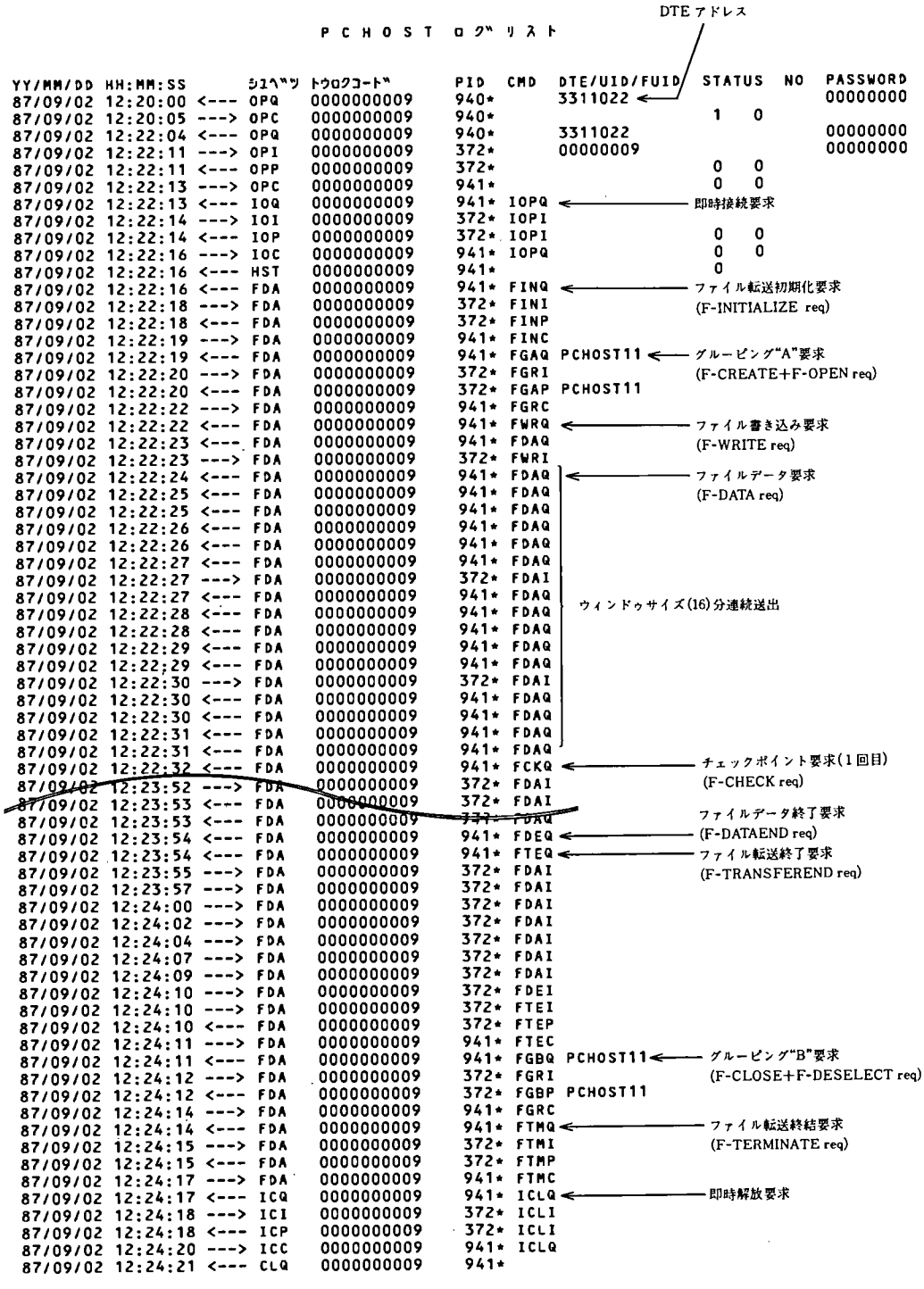


図9 ファイル転送ログ

Fig. 9 File transfer log

4.1 今後の課題

3章の冒頭で述べたように，今回のFTAMの実装は，JUST-PCのセッション機能（カーネル＋全二重），およびNTTPCネットワークの情報センタ接続機能（即時接続プロトコル）の上で展開された．そのためFTAMの下位層とのマッピングにおいて，次のような実装上の決めを行った．

1） FPDUをすべてS-DATAに当たる即時接続の電文転送コマンドにのせた．とくにセッション層に小同期の機能単位がないため，データフロー制御のためのF-CHECKはS-SYNCMINORではなくS-DATAにのせた．

2） アプリケーション層のACSEに相当する機能をPCHOSTコントロールの機能で代替した．

次にPCHOSTの実装仕様の評価だが，異機種ホスト間接続の場合と異なり，パソコン—ホスト間接続の場合には，JUST-PCセッション上での今回の切り出しが効率的にも妥当なものであったと考えている．またパソコン通信における実装の立場からは，セッション層の小同期，半二重（トークン制御を伴う）の機能単位はあまり必要とは思われない．今後，対パソコン向けの実装仕様は今回の切り出しにかなり近いものになると思われる．

4.2 今後の動向

FTAMは現在，国際規格案（DIS）から国際規格（IS）へ標準化が進んできており，各国の標準化機関や業界が実装仕様の切り出しを行っている．

ユニシスでは，DDP*のセッションを用いたFTAMを開発中で，今年11月にはINTAP**のOSI相互接続実験に参加する．

今後は，新規に開発されるファイル転送システムのみならず，既存のシステムも異機種間接続の社会的要請からFTAMの実装が進んでいくであろう．

---

* DDP：Distributed Data Processing

** INTAP：Interoperability Technology Association for Information Processing, (財)情報処理相互運用技術協会

参考文献　[1] パソコン通信ネットワーク　サービス・プロトコル仕様書（第2版），1985年12月，日本電信電話株式会社．

[2] DIS 8571/1～4 Information Processing Systems—OSI File Transfer, Access and Management, 1986年8月．

執筆者紹介　広　田　雅　史（Masafumi Hirota）

昭和26年生．51年北海道大学大学院理学研究科修了．同年，日本ユニシス(株)入社．マーケティング企画，コミュニケーションシステム開発を経て，ニューメディアソフトウェア開発に従事．現在統合OAシステム部，ニューメディアソフトウェア二課に所属．

UNISYS 2200/400 シリーズのハードウェア技術

UNISYS 2200/400 Series Hardware Technology

北村紘次
K. Kitamura

1. はじめに

UNISYS 2200/400 シリーズは，CMOS VLSI ASIC (Application Specific I/C) と高性能バス構造を全面採用した，中/大型のコンピュータ・システムである．その中心となる素子技術は，California 州 Rancho - Bernardo にある Unisys 半導体工場で開発された世界でも最先端の CMOS である．

2200/400 では，27 種(うち IP は 8 種)の VLSI ASIC を新規に開発，IP をはじめ，MS，IOP，入出力チャネルに，全面的に採用することにより，高いコストパフォーマンス，低消費電力，省占有面積，高信頼性を実現している．

本稿では，以下の各項目にそってハードウェアの特長を紹介する．

1) 2200/400 シリーズのシステム構成
2) チップセット IP および MS
3) IOP および入出力チャネル
4) 無人化支援機能
5) パッケージングと素子技術

2. 2200/400 シリーズのシステム構成

図 1 に 2200/400 シリーズのシステム構成概念図を示す．2200/400 シリーズは，2200/200 シリーズで実績のある CMOS VLSI ASIC やバス構造を，さらに発展強化したものである．

ここでは，2200/400 システムを構成する各装置の概要説明とバス構造について述べる．

2.1 装置と概要説明

1) IP：中央処理機構 (Instruction Processor) ……1 ボード上に 8 種 9 個の CMOS VLSI ASIC からなる新 1100 チップ・セットを 2 セット (二重化構成) と，16 K 語 (64 K バイト) のキャッシュ・メモリを搭載している．二重化した新 1100 チップ・セットでデータの完全性を保証している．
2) MS：主記憶機構 (Main Storage) ……1 ボード上に制御部は CMOS VLSI ASIC，記憶部は 1 M ビット DRAM を用いた SIP (Single Inline Package) を搭載して，1 ボードで 4 M 語 (1 語：36 ビット，16 M バイト)，最大の 4 MS 構成で 16 M 語 (64 M バイト) となる．
3) IOP：入出力処理機構 (Input/Output Processor) ……1 ボード上に CMOS VLSI ASIC を用いた I/O プロセッサおよび入出力バス・コントローラを搭載している．搭載するファームウェアにより BMC 用と WDC 用の 2 種類に分けられる．

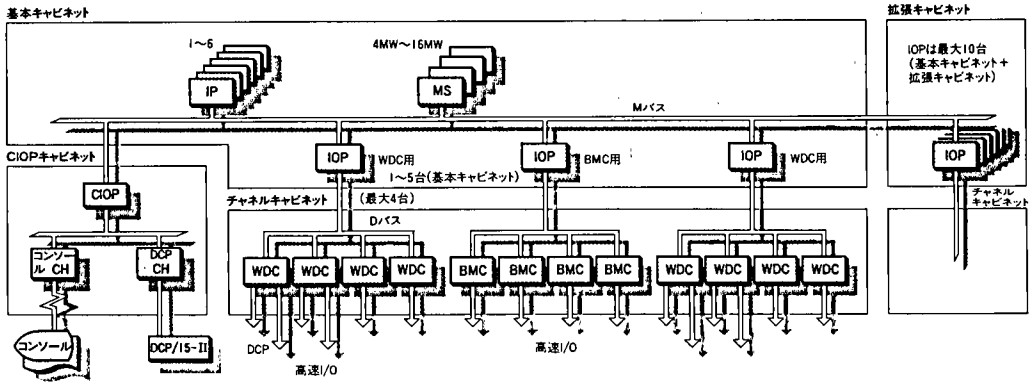


図 1 2200/400 シリーズのシステム構成概念図

Fig. 1 2200/400 Series system configuration concept

4) BMC：ブロック多重チャネル（Block Multiplexer Channel）……一つのIOPで四つのブロック多重チャネルが制御される．各ブロック多重チャネルの最大データ転送速度は3.0Mバイト/秒である．

5) WDC：ワード・チャネル（Word Channel）……一つのIOPで最大八つのワードチャネルが制御される．各チャネルの最大データ転送速度は822K語/秒（ISIチャネル），250Kキャラクタ/秒（ESIチャネル）である．

6) CIOP：コモン入出力処理機構（Common I/O Processor）……業界標準のマルチバスIIとM 68020マイクロ・プロセッサにより中低速周辺装置/通信制御装置とインテリジェント・コンソールを支援する．また，システム制御および故障診断プロセッサの機能も兼ね備えている．

7) 基本キャビネット（Basic Cabinet）……最大6個のIP，4個のMS，5個のIOPを搭載できるキャビネットである．

8) 拡張キャビネット（Expansion Cabinet）……IPが5個か6個のとき，またはIOPが6個以上の構成のとき接続するキャビネットである．全部のIOP（最大10個）がこのキャビネットに収納でき，基本キャビネットとは，Mバス拡張機能（M-BUS Expander）で接続される．

9) チャネルキャビネット（Channel Cabinet）……3個までのWDC，またはBMCDバス・モジュールが内蔵でき，各モジュールごとに最大8チャネルが収納できる．キャビネット当たり最大16個のWDCまたはBMCチャネルが接続でき，システム当たり最大4キャビネットで64チャネルを提供する．

10) CIOPキャビネット（CIOP Cabinet）……CIOPを構成するプロセッサ，4Mバイトのメモリなどと最大10個のDCPチャネル/コンソール・チャネル，を収納するキャビネットである．

一般に，バス構造を採用することはシステム構築の簡素化，ケーブル接続の削減による信頼性の向上，構成拡張の柔軟性などの長所が挙げられる．しかし，その反面いくつかの装置が共通のバスを使用することにより，発生する問題（競合によるシステム効率の低下）がある．

このアクセス競合により発生する問題を解決するため，Mバスでは次のような方策を行っている．

バスに接続する装置のバス・アクセス権獲得方法は，2200/200シリーズのSバスと同様の分散同時評価方式を採用している．これは各装置がバス・サイクルごとにアクセス権が獲得できるかを判断する方式である．この方式は各装置ごとに判断回路が必要になるが，要求を出してからバスのアクセス権を獲得までの時間を短縮できる．

各装置に与える制御権獲得の優先度は，各装置の機能性を考慮して与える固定の優先度方式と，ある装置群（たとえばIOP）ごとに優先順位が変わるローティショナル・プライオリティ（Rotational Priority）方式の2種類を備えている．これにより各装置が均等にバスの使用ができるように保証している．しかし，IOPではバスを優先的に使用したい場合，他の装置のアクセス権を一時的に抑制する機能（Request Inhibit）も備え，一時的に専有することができる．

MSはデータ転送の中心的装置であり，バスの要求が集中するところである．従来のMSはある装置よりの読み出し/書き込み要求を受付けて作動を開始すると，作動が完了するまでは他の装置よりの要求は受付けず，バスの使用効率が低下した（2200/200の場合）．2200/400シリーズのMSは，これを改善して読み出し/書き込み作動中でも，ある一定の要求を先取りで受付けスタックする機能を設けバスの効率を向上させている．

さらにIPは16K語（64Kバイト）のキャッシュ・メモリをIPごとに備えているため，IPの能力を向上させると共に，バスのアクセス競合の機会を少なくしている．

今まで述べた方策により，Mバスに最大6個のIP，4個のMS，10個のIOP，2個のCIOPアダプタが効率よく作動できるのである．

2.2 バス構造

2200/400で採用した階層バスは，Mバス（Mシリーズ・バス）と，Dバス（入出力バス），それにマルチバスII（ローカル・バス）の3種類である．

1) Mバス（Mシリーズ・バス）……Mバスは，2200/200シリーズのSバス（システム・バス）を拡張して約3倍の能力に引き上げ，将来への拡張性も設計に取り込んでいる．1バス・サイクルは80ナノ秒であり，データ幅は2語

(72ビット)である．バス獲得サイクルとデータ/メッセージ転送サイクルは，並行して動作可能である．Mバスの基本転送は8語であり，最高20M語/秒（90Mバイト/秒）のデータ転送ができる．

2) Dバス(入出力バス)……Dバスは，IOPとWDCまたはBMC間のデータ転送専用バスでデータ幅は1語である．データの最大転送速度は37.5Mバイト/秒である．

3) マルチバスII（ローカル・バス）……業界標準のマルチバスIIは，CIOPキャビネット内に存在する．データ幅は32ビット，データの転送速度は1.25～3.0Mバイト/秒のバスである．

3. チップセットIPおよびMS

3.1 IP (Instruction Processor)

2200/400シリーズのIPは，2200/200シリーズのIPで開発された1100チップセット（6種7個）をさらに機能強化した8種9個の新1100チップセットと，3種3個のMバス・インタフェースを制御するマネジメント・チップ，そして16K語（64Kバイト）のキャッシュ・メモリから構成されている．

2200/400シリーズIPの新チップセットで強化された機能は，次のようなものである．

- パイプライン制御
- チップセットの高速化（108ナノ秒→80ナノ秒）
- キャッシュ・メモリの容量拡大（8K語→16K語）

新1100チップセットの各チップの機能を簡単に説明する．図2がIPのブロック図である．

1) DEC(Decode)チップ……シリーズ1100の命令語を解読し，他のチップにマイクロ命令の開始番地，および命令実行に必要な種々のデータを与えて命令の実行を制御する．これ以外に，AGUチップから送られてくる割込み要求の優先制御，あるいはカンタム・タイマの計量などの機能がある．

2) AGU (Address Generation Unit) チップ……AGUチップは，シリーズ1100アーキテ

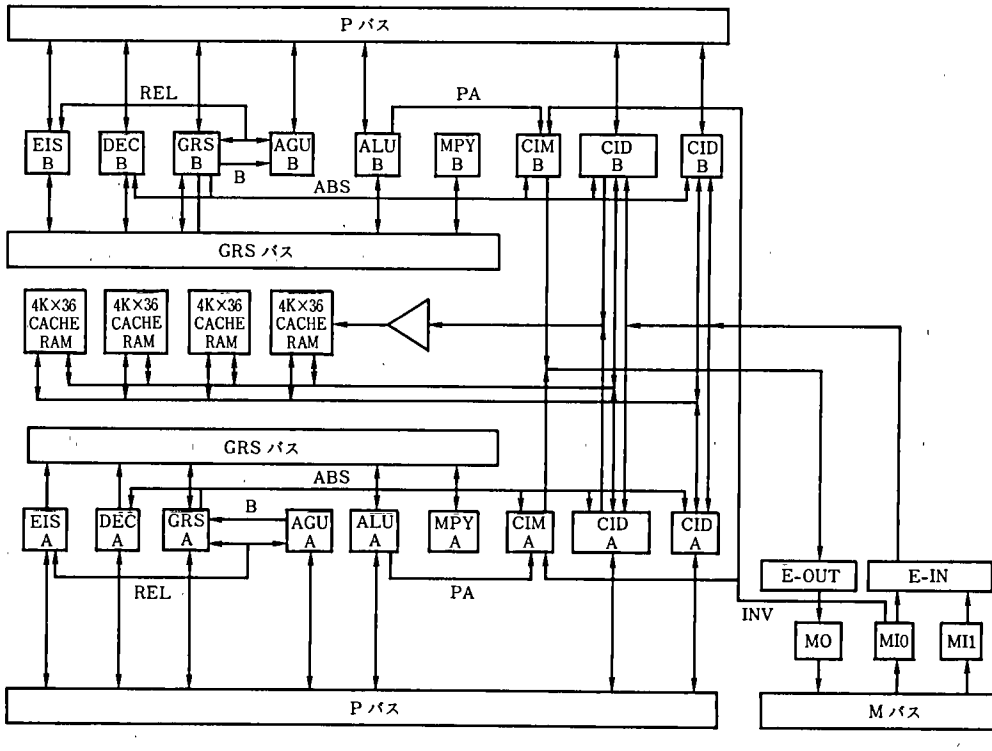


図2 2200/400シリーズIPブロック図

Fig. 2 IP block diagram

クチャにおけるアドレス変換を実行するチップで，変換機構，マイクロエンジン，データパス，レジスタスタックなどで構成されている．アドレス変換の機構は基本モードと拡張モードを支援している．そのための32個のベースレジスタ，リミットレジスタ，セイブファイルがレジスタ・スタックに存在する．CALLやGOTO命令などアドレス変換の複雑な命令は，120ビット672語のマイクロ命令とマイクロ・エンジンにより実行される．

3) ALU (Arithmetic Logic Unit) チップ……ALUチップはシリーズ1100の四則演算，論理演算を実行するチップである．アキュムレータ，マルチプレクサ，ALU等からなるメイン・データパス，浮動小数点の指数部データパス，120ビット256語のマイクロ命令とマイクロエンジン等から構成される．

4) GRS (General Register Stack) チップ……GRSチップは，2200/200用チップセットのALUチップに存在していたGRS（36ビット128語の汎用レジスタ）を独立させたものである．半サイクル（40ナノ秒）で読み出し／書き込みが同時にできる高速性と，パイプライン制御のために発生するコンフリクトを遅延なく解決するテクノロジが採用されている．また，メイン・メモリへの部分書き込みのための回路も内蔵している．

5) CIM (Cache Interface Management) チップ……2200/400シリーズのキャッシュ・メモリは，IPごとに16K語（64Kバイト）の容量を持つ．最大構成の2200/406では，システム当たり96K語（384Kバイト）の大容量となる．

セット・アソシエィティブ方式によるアドレス・マッピングと1100/60シリーズで採用されたPaired LRU (Least Recently Used) を，さらに強化したReplacement by Validityと呼ばれるアルゴリズムによるリプレースメント方法がとられている．

16K語のキャッシュ・メモリは256セットから成り，1セットは4ブロックで，1ブロック16語の構成となっている．MSとのデータ転送は一度に8語が基本のため，1ブロックを二つのハーフ・ブロック（8語）として取り扱うことができる．CIMチップは，これらのキャッシュ・メモリと他のチップとのデータ転送の各種制御を実行する．

6) CID (Cache Interface Data) チップ……CIDチップは，キャッシュ・メモリと他のチップ，そしてMバスへのデータ転送を行う．二つのCIDチップをボードに搭載し，それぞれのチップが各々18ビットを取扱う．

キャッシュ・ミスの場合，MSからデータが転送されるまでのアイドル・サイクルを有効利用する技術が採用されている．これは要求したデータが戻ったことで，次の処理を続行する他のチップとキャッシュ・メモリ間でのコンフリクトを防ぐためのものであり，MSから送られてきたデータ（要求した番地が先頭に送られる）を他のチップへ送ると同時に8語から成るRM (Read Miss) スタッフへ一度保持する．

RMスタックに書き込まれたデータは，次にキャッシュ・ミスとなったアイドル・サイクルにキャッシュ・メモリへ書き込まれる．そしてキャッシュ・メモリへ書き込まれるまでの間は，RMスタック上でのヒットもチェックしている．

7) EIS (Extended Instruction Set) チップ……シリーズ1100のフィールド命令(バイト単位および任意のビット数のデータを取り扱える29種類の命令)を実行する専用チップである．138ビット768語のマイクロ命令とマイクロ・エンジンを持っている．

8) MPY (Multiply/Divide) チップ……2進数，浮動小数点データの演算を高速で行うチップである．2200/200シリーズでは，M/Dチップはオプションであるが，2200/400シリーズは標準装備にし，大幅な性能アップを行った．

以上，説明してきた8種のチップは，Pバス，PAバス，GRSバスなどの内部バスで結合される．

次にパイプライン制御について説明する．図3がパイプライン制御例である．

シリーズ1100の加算命令(演算レジスタとオペランドのデータを加算する）を例にとって，各チップとバスの動きを説明する．1サイクルは10ナノ秒が8フェーズから成る80ナノ秒である．

サイクル'N'のフェーズ0で各チップは，CIDがドライブした命令語（仮に命令番地をPとす

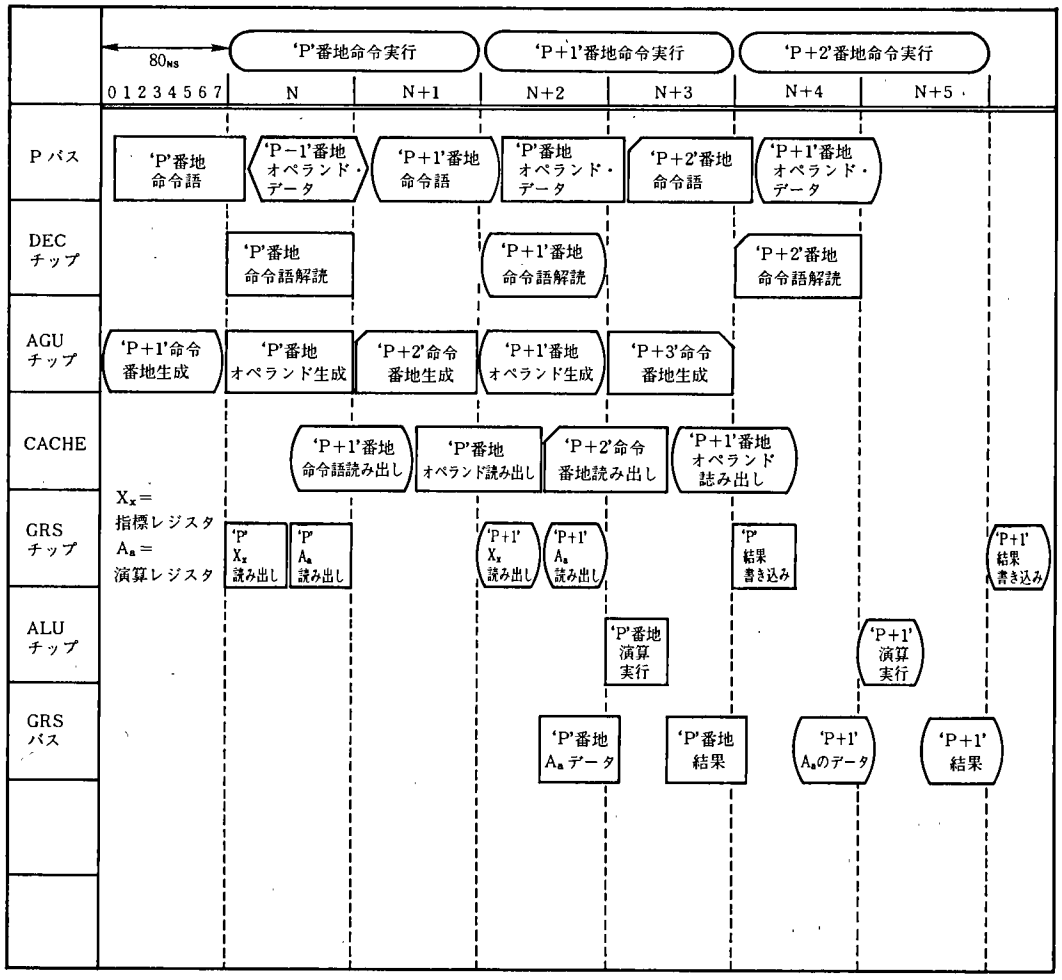
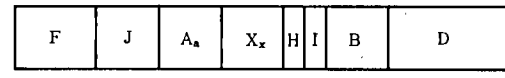


図3 パイプライン制御例

Fig. 3 Sample of pipeline control

る）をラッチする．GRS チップは，命令語の指標レジスタを読み出す（図4）．同時に AGU チップは，命令語のベース・レジスタ指定部分で指定されたベース・レジスタの内容を GRS チップに送って 24 ビットの実アドレス（オペラント・アドレス）を生成し，CIM チップへ要求する（サイクル 'N+1'）．CIM チップは TAG メモリをアスセスし，要求されたアドレスがヒット（キャッシュ上に要求番地がある）かミスかを判定する．

ヒットの場合，CID チップへヒットしたブロックを通知し，CID チップはそのブロックの 16 語のうち要求された番地の 1 語を選択してサイクル 'N+2' に P バスへドライブし，ALU チップへ送る．



F ：命令コード
J ：部分語指定または F/J で命令コード
A_n：演算レジスタの指定
X_x：指標レジスタの指定
H ：指標レジスタの増減指定
I ：間接番地の指定
B ：ベース・レジスタの指定
D ：オペラント・アドレス（ディスプレイスメント部分）

図4 シリーズ 1100 命令語

Fig. 4 Instruction format of Series 1100

キャッシュミスの場合は，CIMチップはマネジメント・チップを経由して，MSから読み出し（要求番地を含む8語転送）を行う．

一方GRSチップは，サイクルNの後半の40ナノ秒で命令語のAa部分で指定された演算レジスタの読み出しを行い，CIDチップからのオペランド・データと同期してサイクル'N+2'にGRSバスへドライブし，ALUチップへ送る．サイクル'N+3'前半の40ナノ秒でALUチップは，演算レジスタの値とオペランド・データを加算し演算結果をGRSバスへドライブする．

サイクル'N+4'前半で，GRSチップは演算結果を演算レジスタに書き込む．パイプ・ライン制御として，サイクル'N+1'では'P+2'の命令語の番地生成が行われ，サイクル'N+2'で'P+1'の命令語が実行される．パイプライン制御で，ロード命令，加減算命令など，基本的な命令は見かけ上2サイクルで実行できることになる．

次にデータ・インテグリティについてであるが，2200/400シリーズも2200/200シリーズと同じく，IPはチップセットを2セット持つ二重化構造を採用している．二つのチップセットは，キャッシュ・メモリを共用する以外独立して動作し，同時に同じことを行う．一方のチップセットをマスタ，他方をチェッカーと呼び，両者の演算結果の比較検査は，チェッカー側のCIMチップが行う．もし異常が検出された場合，MSの書き込みは停止され即時にSCD（System Control Driver：従来のSSPに替わる機構）へ異常検出を報告する．

また，二重化機構以外の部分，すなわち両者のCIDチップとキャッシュ・メモリ間，CIMチップとマネージメント・チップ間については，スルー・パリティによる奇偶検査を行っている．

以上，IPのチップセットの説明を行ってきたが，Mバスを経由して他ユニット（MS，IOP，CIOP等）とのコミニュケーションを行うマネージメント・チップ（MI 0，MI 1，MO 0）にも新しいアーキテクチャが採用されている．それはHIS（Hardware Independent Software）と呼ばれるものである．

従来のIPは，他ユニットとのやり取りで異常が検出されたとき，一部の異常を除いてはハードウェアによる再試行を行っていた．再試行の結果，異常が再び検出されると，オペレーティング・システムに割り込みを発生させ，異常状態の分析，リカバリ・アクションはオペレーティング・システムが行っていた．

2200/400シリーズのIPでは，オペレーティング・システムによるリカバリ・アクションの必要が発生したとき，オペレーティング・システムが行うべきアクションと異常箇所をエラー状況語で知らせる（図5）．このエラー状況語はシリーズ2200に共通のものであり，オペレーティング・システムはプロセッサ・タイプに無関係に，ハードウェアから知らされたエラー状況語に従ってアクションするだけでよいことになる．

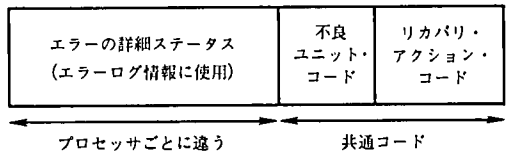


図5　エラー状況語

Fig. 5　Error condition word

3.2　MS (Main Storage)

2200/400シリーズのMSは1Mビット DRAMを採用して，MS当たり4M語（16Mバイト）の容量を持つ．システム当たり4MSまで増設でき，最大16M語（64Mバイト）構成となる．MSは，3種5個のVLSI ASICの開発と1MビットDRAMの3次元実装により，36.6 cm×28.2 cmのボード上に4M語を搭載することができた．次に3種5個のチップの簡単な説明をする．図6がMSのブロック図である．

1)　MAC（Memory Array Controller）チップ……MACチップは，Mバスのインタフェース制御，メモリ・アレイ制御，データパス制御，エラー検出など中心的な役割を行っている．データパス制御は，MACチップから他のチップへ信号を送り制御している．MACチップは，MS当たり1個使用されている．

2)　IDR（Input Data Register）チップ……IDRチップはMバス・インタフェースのデータ部分のインプットとECC（Error Correction Code）生成，メモリ・アレイへの書き込みパス，メモリ・アレイのリフレッシュ，命令語の解析とスタックなどを行っている．IDRチップは奇数番地用と偶数番地用があり，MS当たり2個使用している．

3)　ODR（Output Data Register）チップ……

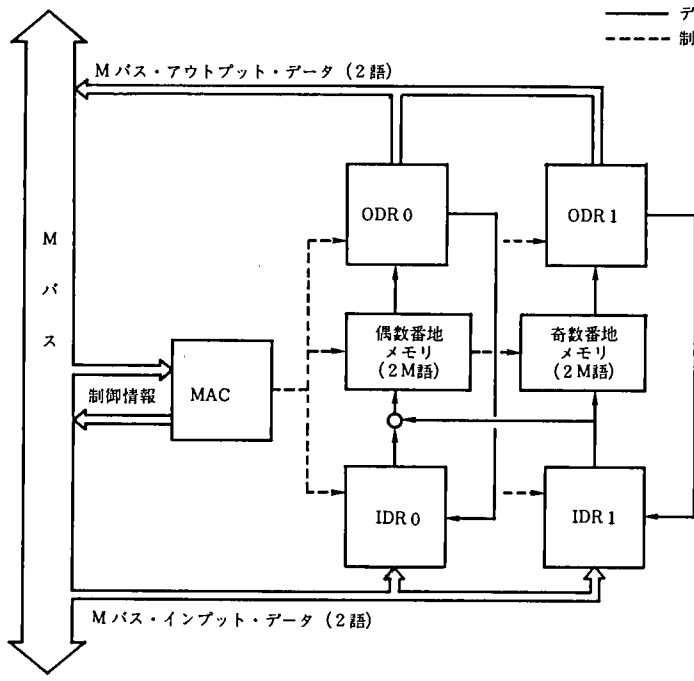


図6　2200/400 シリーズ MS ブロック図
Fig. 6　MS block diagram

ODR チップは，M バス・インタフェースへデータのアウトプットと，メモリ・データの 1 ビットエラー修正，2 ビット以上のエラー検出などを行っている．ODR チップも MS 当たり 2 個使用している．

2200/400 シリーズの MS の大きな特徴は，IP と同じくパイプライン制御ができることである．IP からの要求を処理中に他のユニットからの要求が来た場合，命令語および書き込みデータをスタックすることができ，連続して命令を実行できる．パイプ・ライン制御によって，IP からの基本読み出し単位である 8 語読み出しは，480 ナノ秒という高速サービスが可能となった．

以下に，パイプライン制御による命令ごとのサイクル・タイムを記載する．

命　令	サイクル・タイム
読み出し　2語	240（ナノ秒）
〃　4語	320
〃　8語	480
書き込み　1語	240
〃　2語	320
〃　4語	400
〃　8語	560
部分書き込み	560

4. 入出力チャネル

2200/400 シリーズでは，入出力チャネルとして I/O プロセッサ（以下 IOP と略す）により制御される高速汎用チャネル（最大 64 チャネル）と，CIOP（Common I/O Processor）により制御される専用チャネル（最大 10 チャネル）の 2 種類があり，合計で 74 チャネルを提供している．

以下に，入出力制御機構の特長および各チャネルについて述べる．

4.1 IOP

IOP (Input/Output Processor) は，基本キャビネットに内蔵され，高速・大容量のディスク装置やテープ装置および DCP ファミリ通信制御装置などを接続することができる．IOP には接続されるチャネルのタイプにより，BMC (Block Multiplexer Channel) 系 IOP と WDC (Word Channel) 系 IOP の 2 種類があり，両系ともハードウェアは同一でファームウェアのみが異なる設計となっている点が特長である．また IOP とチャネル間には，D バスと呼ばれるチャネル専用高速バスを新たに開発採用したことにより，IOP 当たり 4 個の BMC，または 8 個の WDC を高速に制御できる設計となっている．

図7にIOPチャネルの構成を示す．

基本キャビネット内に最大5個のIOPを，また拡張キャビネットを付加にすることにより，システム当たり最大10個までIOPを接続できる構成となっている．各々のIOPは，Mバスを介してMSとデータ授受を行う．

また，IPとのインタフェースは，シリーズ1100，2200共通のUPI（Universal Processor Interface）プロトコルが使用される．

IOPハードウェアの特長としてVLSI ASICの採用がある．IOPでは，ハードウェア回路の大部分を5種5個のVLSI ASICに集約したため，従

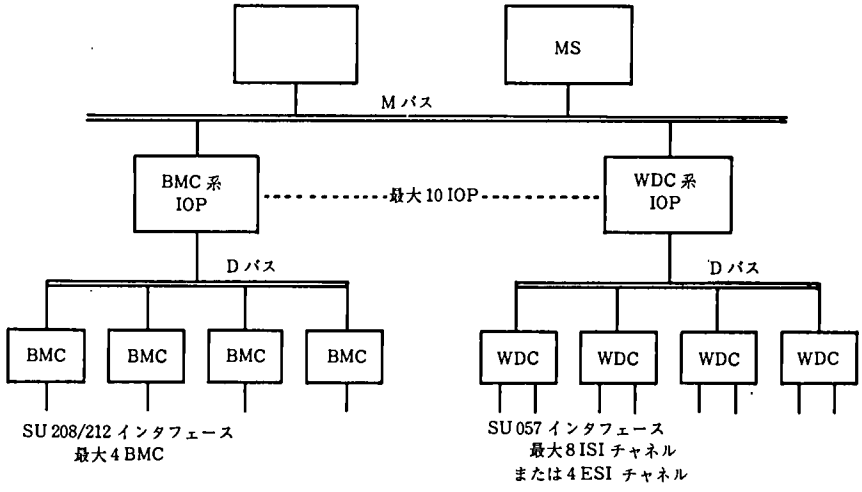


図7　IOPチャネル構成図

Fig. 7　IOP/channel configuration

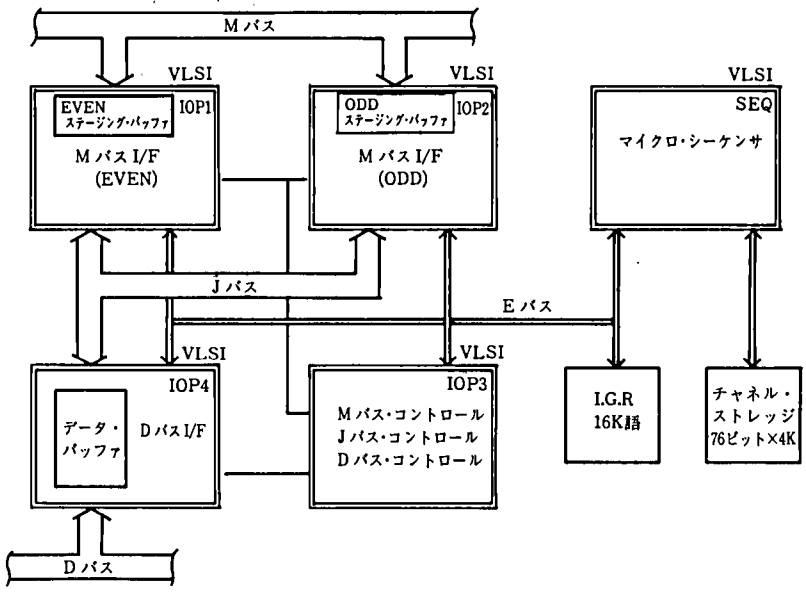


図8　IOPブロック図

Fig. 8　IOP block diagram

来の1100/70シリーズでは約100枚のボードにより構成されていたものが，寸法28.2 cm×36.6 cmの1枚のボードにすべての機能を納めることができ，大幅に信頼性を向上させている．

また，高速データ転送を達成するために，データ・トランスファ・ファシリティ（DTF）を独立させ，ファームウェア（マイクロ・シーケンサ）の介在を少なくしている．図8にIOPハードウェアのブロック図を示す．IOPハードウェアは，機能的に次の三つに分割することができる．

1) Mバス・インタフェース部……IPからのI/O命令の受領およびMSの読み出し/書き込みを実行する部分で，IOP 1，IOP 2と呼ばれる2個のVLSI ASICにより処理される．

ここでは，Mバス・メッセージを解析し，メッセージが自分に対するI/O命令であれば，マイクロ・シーケンサ部へ割り込みにより知らせる．またチャネル単位にMバス・データ転送に必要な転送先メモリ・アドレス，および転送語数を内部レジスタに記憶しているので，一度データ転送が開始されるとその後は，マイクロ・シーケンサの介在なしにデータ転送を続行することができる．

MSへの読み出し／書き込みは，高速性およびバスの負荷軽減の意味から8語単位のブロック転送を基本としている．このため，各チャネル・入出力個別に8語のステージング・バッファを持っており，効率の良い高速データ転送を実現している．

2) データ・トランスファ・ファシリティ（DTF）部……チャネル側とのデータのやり取り，すなわちDバスとのデータ転送はIOP 3，IOP 4と呼ばれるVLSI ASICにより処理され，IOPインターナルなJバスを介して先に述べたMバス・インタフェース部とデータの授受を行う．

この部分のハードウェアは，一度マイクロ・シーケンサによりI/O命令の実行が起動されると，その後のデータ転送はハードウェアが自動的に行い，データ転送が終了するとチャネル側からのインタラプトにより，マイクロ・シーケンサに制御を戻すようになっている．また，チャネル単位に32語のデータ・バッファが用意されており，Dバスとの高速データ転送を可能にしている．

3) マイクロ・シーケンサ部……サイクル・タイム80ナノ秒のカスタム・デザイン・マイクロ・シーケンサで2組の独立したALU（演算回路）を持っている．この部分ではIOPファームウェアを実行し，Mバス・インタフェースおよびの各VLSI ASICに対しては，インターナルなEバスを介してI/O命令の実行を指示する．

IOPファームウェアは，76ビット×4K語のコントロール・ストレージ（COS）上にBMC用，またはWDC用のどちらか一方のファームウェアがシステム・イニシャライゼーション時にロードされている．このファームウェアは，マイクロ・シーケンサにより逐次読み出され実行される．

IOPファームウェアにより実行されるI/O命令は，以下の四つの基本プロセスで処理される．

① CAW（Channel Access Word）プロセス：IPからのI/O命令の実行指示“UPI SEND”メッセージにより起動され，CAWの読み出し・解析・実行を行うプロセスである．

② SIOF（Start I/O Fast Release）プロセス：CAWプロセスにより起動されI/O命令群であるCCW（Channel Command Word）を読み出し変換して，SIOFキューを作成する．チャネルに対しては，I/O命令の実行を開始させる．

③ CCW（Channel Command Word）プロセス：チャネルからのSIOFリクエストにより起動され，該当チャネルのデータ転送を開始させるプロセスである．データ転送に必要な情報はCCWをもとに，このプロセスでDTFの各VLSI ASICにセットされ，データ転送が起動される．

④ CSW（Channel Status Word）プロセス：チャネルからのステータス・リクエストにより，起動されるプロセスでチャネル・ステータス・ワード（CSW）を作成し，MSに書き込み後，IPに対して“UPI SEND”メッセージによりI/O命令の完了を通知する．

IOPファームウェアには，コマンド・チェーン，データ・チェーンを高速処理する手法としてCCWプリフェッチ機能がある．8個のCCWを1

ブロックとして，最大512ブロック（4096 CCW）までプリフェッチすることができる．

4.2 Dバス

Dバスは，チャネル・キャビネットの中にあって最大4個のBMCまたは8個のWDCを接続できる．

Dバスは36データ＋2パリティ・ビットの他，コントロール，クロック，スキャン・セットなど，合計98本のシグナルから成っている．データ転送は，3サイクル（240ナノ秒）で2語転送が可能であり，最大転送能力は37.5Mバイト/秒と十分に高速チャネルを並行処理できる設計となっている．

Dバス・プロトコルでは，以下のメッセージが定義されている．

① SIOF Available（FC=0）メッセージ：チャネルに対して実行すべきI/O命令があることを知らせる．

② Send Command Word（FC=2）メッセージ：実行すべきI/O命令コードをチャネルに送る．

③ Terminate（FC=1）メッセージ：チャネルにデータ転送の終了を知らせる．

④ Selective Reset（FC=4）メッセージ：チャネル：インタフェースのリセットを指示する．

⑥ Clear Channel（FC=5）メッセージ：チャネルのクリアを指示する．

4.3 BMC

BMC（Block Multiplexer Channel）は，チャネル・キャビネット内のDバス・モジュールに内蔵され，高速バイト系入出力装置を接続できる．図9にBMCハードウェアのブロック図を示す．BMCハードウェアでも，VLSI ASICを採用したことにより，1100/70シリーズでは約30枚のボードで構成されていたものが，寸法22.1 cm×23.4 cmの1枚のボードに集約でき，大幅に信頼性の向上がなされた．

さらに，VLSI ASIC内部には16ビットのマイクロ・シーケンサの他，データ・トランスファ・シーケンサおよびデータ・バッファ等を持っており，最大3.0Mバイト／秒の高速転送を可能にしている．

4.4 WDC

WDC(Word Channel)は，従来のシリーズ1100のワード・チャネルに接続されていたI/Oサブシステムを2200/400シリーズに接続するためのチャネルで，ISI（Internally Specified Index）およびESI（Externally by Specified Index）の両インタフェースをサポートする．WDCはチャネル・キャビネット内のDバス・モジュールに内蔵され，ISIインタフェースに高速・大容量のディスク装

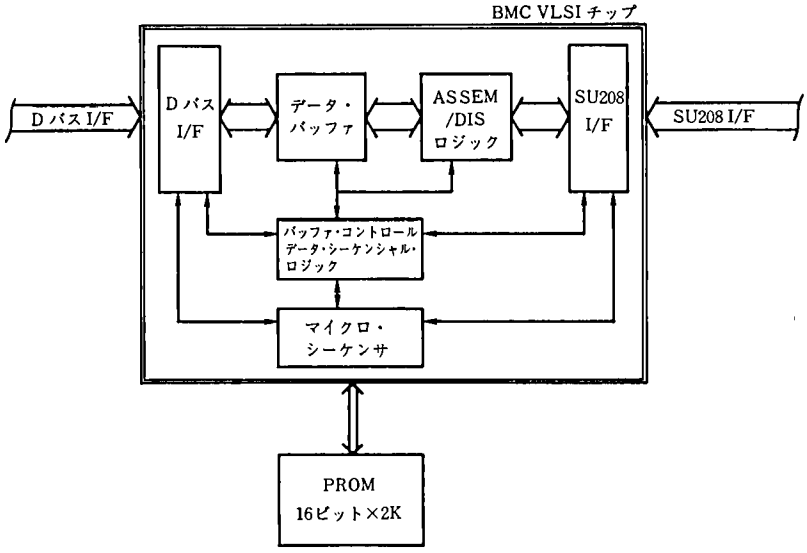


図9　BMCブロック図

Fig. 9　BMC block diagram

置やDCPファミリ通信制御装置を，またESIインタフェースにはGCS通信制御装置を接続することができる．

図10にWDCハードウェア・ブロック図を示す．

3個のVLSI ASICを採用したことにより，寸法22.1 cm×36.6 cmの2枚のボードにすべての機能を集約することができた．

4.5 コモンI/Oプロセッサ（CIOP）

CIOPは，中低速I/Oサブシステムを2200/400シリーズに接続するための専用チャネル（最大10チャネル）を持っており，これらのチャネルは1100オペレーティング・システムからは従来のBMCと同等に扱われる．

また，CIOPには従来のシリーズ1100のSSP（System Support Processor）が行っていたシステム・コントロールの機能を持ち，システム・イニシャライゼーション，システム・リカバリ，スキャン・セットおよび1100オペレーティング・システムとのコミュニケーションとして，ELF（Exec Link Function）機能をサポートしている．

図11にCIOPハードウェアの構成図を示す．基

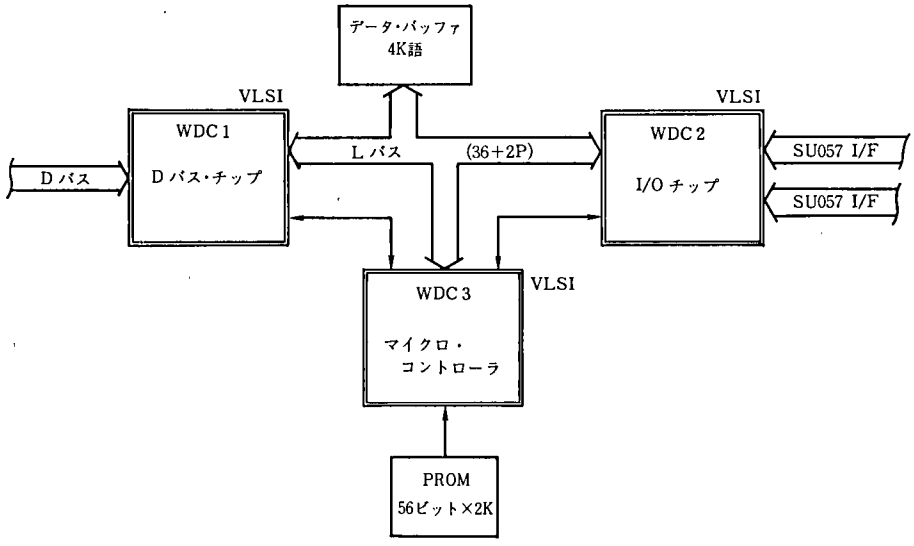


図10 WDCブロック図

Fig. 10 WDC block diagram

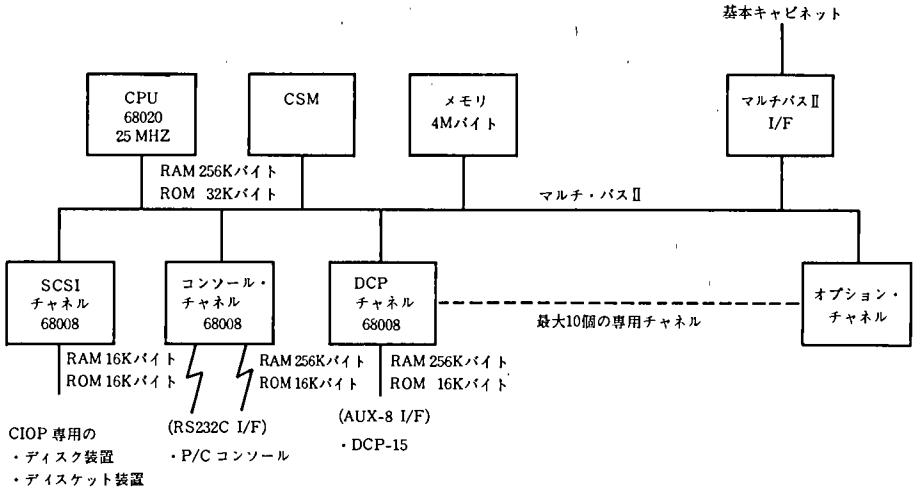


図11 CIOP構成図

Fig. 11 CIOP configuration

本キャビネット内のIP/MSとは，マルチバス-IIインタフェース・ボードにより接続されている．

CIOPハードウェアの特長として，以下が上げられる．

1) CIOPプロセッサ部分に32ビット・プロセッサである68020, 25 MHZ版を採用，さらに4Mバイトのメモリを標準装備し高速処理を実現している．
2) CIOP内部のシステム・バスとして業界標準のマルチバスIIを採用したことにより，将来の柔軟な拡張を可能としている．

CIOPファームウェアは，68020アセンブラ言語で書かれており，機能別にモジュール化されている．さらに，各モジュールはプロセス単位に分割され，CIOPファームウェアの核であるPSOSと呼ばれるオペレーティング・システムにより，管理される階層構造を成している．

CIOPファームウェアの機能として，以下のものがある．

1) システム・イニシャライゼーション
2) システムのフォルト検知およびリカバリ・アクション
3) 各CIOPチャネルに対するI/O命令の実行
4) ELF (Exec Link Function) による1100 OSとのコミュニケーション
5) システム・コントロール

4.6 専用チャネル

専用チャネルには，CIOPファームウェアが専用に使うSCSIチャネルの他，システム・コンソール用としてのコンソール・チャネルおよびDCP 15 II通信制御装置を接続するDCPチャネルがある．これらの各専用チャネルは，プロセッサに68008 10 MHZを使用した1ボードで構成されている．

以下に各チャネルについて簡単に説明する．

1) SCSIチャネル……業界標準のSCSI (Small Computer System Interface) を支援するチャネルで，CIOPファームウェアが専用に使う内蔵ハード・ディスクおよびディスケット装置を接続する．これらの装置は，2200/400シリーズのファームウェアの管理およびシステム・コントロール用ファイルとして使用される．16Kバイトのプログラム・メモリ (RAM) と4Kバイトのデータ・バッファを持ち，最大1.5Mバイト／秒の転送能力を持つ．
2) コンソール・チャネル……コンソール・チャネルは，RS 232-Cインタフェースをサポートするチャネルで，2200/400シリーズのシステム・コンソールを接続することができる．256Kバイトのプログラム・メモリを持ち，最大19,200ボーの転送が可能である．
3) DCPチャネル……DCPチャネルは，AUX-8と呼ばれる8ビットのバイト・インタフェースをサポートするチャネルで，DCP-15 II通信制御装置を接続することができる．

5. インテリジェント・コンソール

従来のシリーズ1100では，システム・オペレーションは専用のオペレータ・コンソールを使用し，システム・コントロールおよびシステムの診断機能はSSP (System Support Processor) により行われていた．

2200/400シリーズでは，インテリジェント・コンソールを採用することにより，これらの機能を一体化した他，新たに自動運転支援機能 (UOF)，および使いやすさを向上させたイーズ・オブ・ユース (Ease of Use) 機能を追加し，インテリジェント・コンソール上で一括制御ができるようになった．

インテリジェント・コンソールのハードウェアには，16ビットCPUを持つパーソナル・コンピュータが使用される．40Mバイトのハードディスク，1.2Mバイトのディスケット装置，および4.5Mバイトのメモリ，EGAカラーモニタ・インタフェース等を内蔵し，CIOPのコンソール・チャネルに接続される．

インテリジェント・コンソールのソフトウェアは，オペレーティング・システムとしてUNIX*系のXENIXを使用し，図12に示すようなソフトウェア構成となっている．

2200/400シリーズで支援されるインテリジェント・コンソールの機能として，次のものがある．

- システム・オペレーション
- システム・コントロール
- システム診断機能

*UNIXは，AT&Tの登録商標である．

・フレンドリ・コンソール
・スマート・コンソール
・自動電源コントロール
・日本語（漢字）対応
・ユティリティ

以下に主な機能について簡単に述べる．

1) システム・オペレーション……従来の 1100 オペレーティング・システムのオペレータ・コンソールとしての基本的な機能で，CMC (Console Message Control) プログラムを介して OS 1100 とコンソール・メッセージの授受が成される．

2) システム・コントロール……従来の SSP (System Support Processor) 機能に代わるもので，M シリーズのシステム・コントロールを 2200/400 シリーズに適用したもので，総称して SCF (System Control Facility) と呼ばれる．

2200/400 シリーズの SCF は，図 13 に示すように PC コンソール側の SCP (System Control Program) と，CIOP 側の SCD (System Control Driver) から構成される．

SCD は，SCF のノード・コントロールとして基本キャビネット内の各ユニットに対し，スキャン・セット・インタフェースを介してイニシャライゼーション，およびフォルト・リカバリ等を実

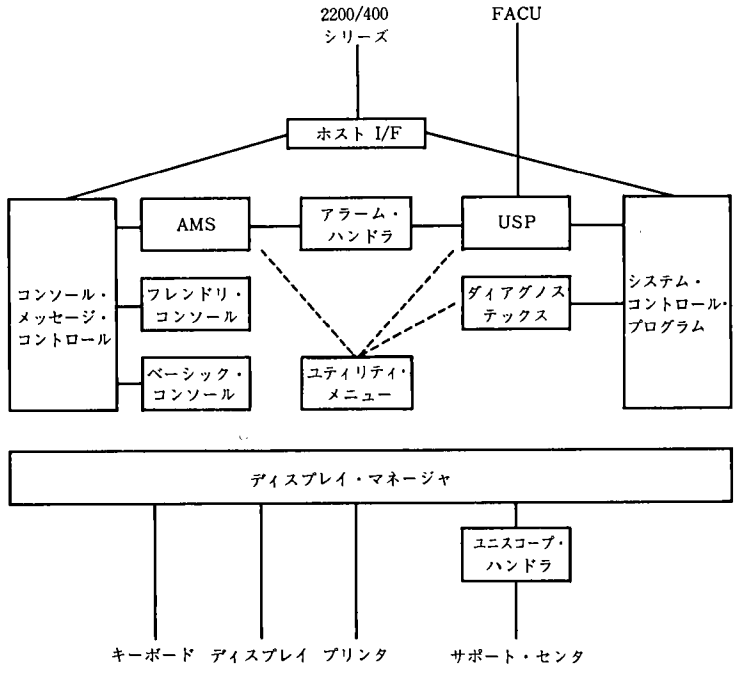


図12 インテリジェント・コンソールソフトウェア構成

Fig. 12 INTELLIGENT console software configuration

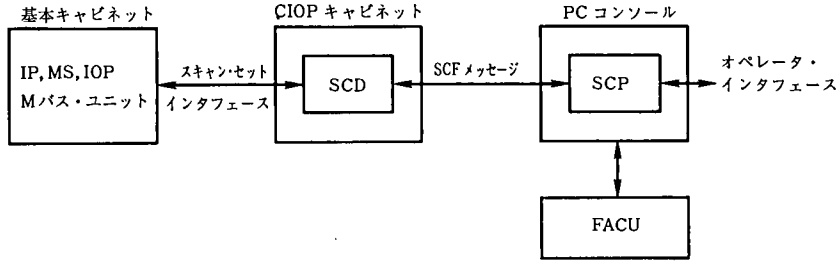


図13 SCF 概念図

Fig. 13 SCF concept

行する．

SCPは，システム・コントロールに関するオペレータ・インタフェースの役目とSCDに対して，システム・コントロールの実行を指示するためのSCFメッセージの授受を行う．

3) フレンドリ・コンソール……2200/400シリーズでは，経験の浅いオペレータや，めったに使わないコンソール操作の実行を補佐するためにイーズ・オブ・ユース機能を提供している．

メニュー選択／穴埋め方式により，簡単にシステム・オペレーションが行え，ヘルプ機能も完備しており，操作性が飛躍的に向上している．

4) 自動運転支援機能（UOF）……無人化／自動運転で要求される機能として，以下のものがある．

① コンピュータ室内の設備機器およびシステム本体の電源を自動制御できること．

② システムの自動立上げ

③ ユーザ・アプリケーション・プログラムの自動起動

④ システム稼働中のモニタと異常発生時の自動対処

⑤ 業務終了後のシリーズの自動停止

2200/400シリーズでは，UOF（Unattended Operation Facility）コンセプトに基づき，これらの機能を完全に支援し自動運転／無人運転を可能にしている．

無人化のためのハードウェアとして，FACU（Facility Access Control Unit）が新たに開発され，インテリジェント・コンソールの制御のもとにコンピュータ室の設備機器の制御・監視および各機器の電源制御・監視等を行う．

インテリジェント・コンソールのソフトウェアでは，このFACUを制御するためのAPC（Automated Power Control）プログラムとスマート・コンソールと呼ばれ，コンソール・メッセージの自動応答を行うAMS（Autoaction Message System）プログラムにより，自動／無人運転を支援している．

1100オペレーティング・システムの機能として，UOFを統合監視する自動運転支援ソフトウェア（UOSS）がある．

2200/400シリーズにおけるUOFは，これらのUOSS，インテリジェント・コンソール，FACU等のハードウェア／ソフトウェアが一体となって無人／自動運転を達成している．

6. パッケージング素子技術

2200/400シリーズでは，2章で述べたバス構造を採用した他，CMOS III技術によるVLSI ASICを全面的に採用したこと，および2200/200シリーズで確立されたNPF（New Power Family）と呼ばれる小型高性能電源システムの採用等により，基本構成で占有面積2 m²にすべての機能を凝

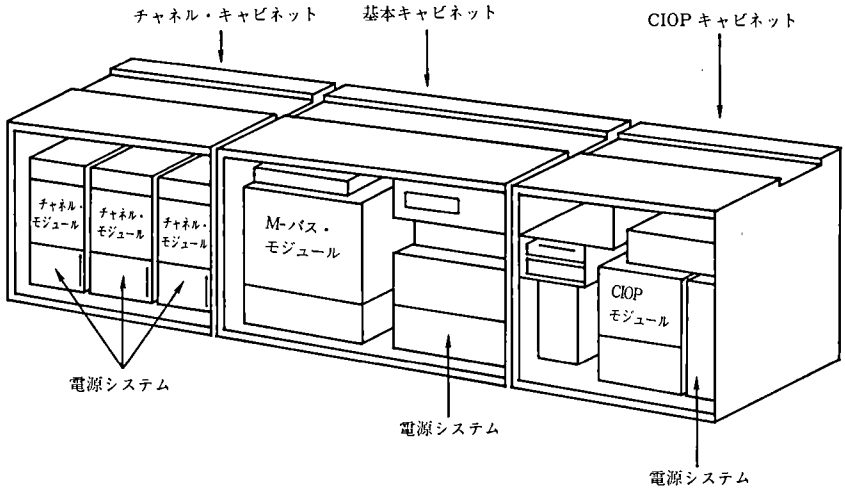


図14　2200/400シリーズ基本構成

Fig. 14　2200/400 Series basic configuration

表1 筐体とバス・モジュール
Table 1 Cabinet and BUS module

筐　　対	筐体のサイズ 幅×高×奥 (mm)	バス・モジュール名	スロット数	モジュールの層　数
基本キャビネット	1,067×953×762	Mバス	15	12
CIOP キャビネット	762×953×762	CIOP モジュール (マルチ・バスII)	16	6
チャネル・キャビネット	762×953×762	チャネル・モジュール (Dバス)	4	6

表2 ボードの特徴
Table 2 Board characteristics

ユニット名	NLSI ASIC 数	ボードのサイズ (cm)	ボードの層　数	入出力ピン数	サーフェース・マウントの有・無
IP	11種　21個	28.2×36.6	12	288	有
MS	3種　5個	28.2×36.6	12	288	有
IOP	5種　5個	28.2×36.6	12	288	有
CCA	4種　4個	28.2×36.6	12	288	有
BMC	1種　1個	22.1×23.4	8	192	無
WDC	3種　3個	22.1×36.6	8	288	無
CIOP	—	17.0×23.0	8	192	無

表3 IP チップセットのゲート数とピン数
Table 3 Number of gate and pins of IP chip set

チップの種類	ゲート数	信号ピン数
ALU	27 K	196
AGU	41 K	145
CID	36 K	196
CIM	40 K	194
DEC	18 K	194
GRS	30 K	195
EIS	36 K	138
MPY	26 K	143
合　計	254 K	—

縮することができた.

これは，今までの 1100/70 シリーズに比べると約 1/3 の床面積で設置可能となり，大幅な省床面積を達成している.

2200/400 シリーズの基本構成は，図 14 に示すように基本キャビネット，チャネル・キャビネット，CIOP キャビネットの 3 筐体から成っている. それぞれの筐体には，バス・モジュールと呼ばれる多層基板のバックパネルすなわちバスがあり，ここに各種のシステム構成ユニット（ボード）を挿入することにより，システムの構築・拡張が設置場所で簡単にできる構造となっている. 表 1 に各筐体とバス・モジュールの関係を示す.

素子技術では，2200/200 シリーズで確立された CMOS III技術をさらに発展させ，2200/400 シリーズでは 27 種の VLSI ASIC が新たに開発され使用されている.

IP 用に開発された新 1100 チップセット 8 種 9 個は，最大 160 K トランジスタを持つカスタム・プロセス設計で作られており，それ以外の 19 種は最大 73 K トランジスタを持つセミ・カスタム・プロセスが用いられている.

表 2 に各ユニット（ボード）における VLSI ASIC の使用状況を，また表 3 には IP 新チップセットのゲート数および信号ピン数を示す.

このように 2200/400 シリーズでは，広範囲にわたり CMOS III VLSI ASIC を採用したことにより，素子の数を大幅に少なくすることができ，システムの信頼性を向上させているばかりでなく低消費電力／低発熱量/省床面積を実現している.

写真 1～3 に IP, MS, IOP 各ボードを示す. また，写真 4 は新チップセットの VLSI 内部の様子である.

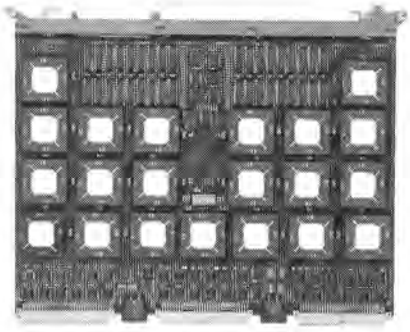


写真1　IP ボード
Photo. 1　IP board

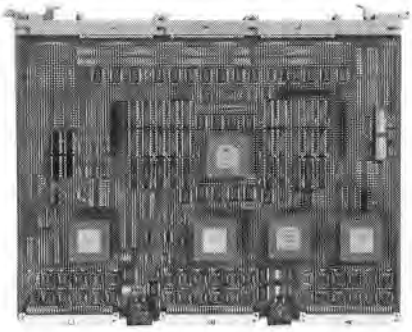


写真2　IOP ボード
Photo. 2　IOP board

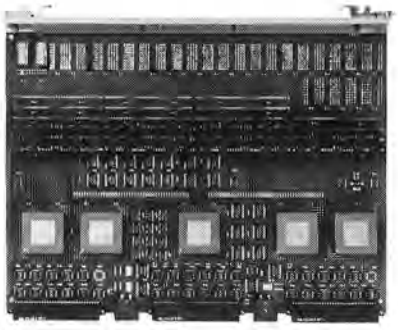


写真3　MS ボード
Photo. 3　MS board

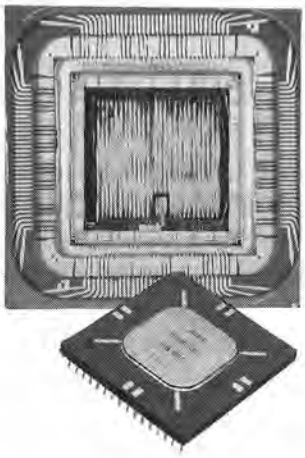


写真4　IP チップセット
Photo. 4　IP chip set

7. おわりに

2200/400 シリーズのシステム構成，チップセット IP，MS，入出力チャネル，自動運転支援機能，実装技術について説明してきた．しかし何と言っても最大の特長は，新 1100 チップセット IP と，その実現を可能とした CMOS-III VLSI ASIC の素子技術にある．

コンピュータを構成する素子に要求される，3 大要素としては，高速スイッチング・タイム，高集積度，そして低消費電力があげられる．微細加工技術の進歩により，スイッチング・タイムの問題を解決した CMOS VLSI はその本来の特長である高集積度と低消費電力という長所を十分に生かし，小／中規模システムの汎用コンピュータ素子技術の中心的位置は，確固たるものとなってきた．

今回，新たに開発された 2200/400 シリーズが，この流れの主要な役割を果たすと共に，将来の更なる発展の礎となるものと確信している．

なお，本稿の執筆に当たっては，ハードウェアプロダクト二部の阿部，大村およびテクニカルサポート部の三橋，各氏の絶大なるご協力を頂き，感謝の意を表したい．

（ハードウェアプロダクト二部）

InfoExec
—SIM を中心として—
—Centering around SIM—

山口裕久
H. Yamaguchi

1. はじめに

InfoExec は，Information Executive（情報の管理者）を意味する当社のセマンティック・データベース統合情報管理システムの商品名で，昨年の9月に発表された．

InfoExec は，InfoExec シリーズと呼ばれる次のようなソフトウェア群から構成される．

1) SIM (Semantic Information Manager, セマンティック情報管理ソフトウェア) ……セマンティック・データベースを作成，管理するデータベース管理システムである．
2) ADDS (Advanced Data Dictionary System, セマンティック・データベース定義/制御ソフトウェア) ……データベースの定義や変更を行い，データベースに関する定義情報の一元管理を行う．
3) OCM (Operation Control Manager) ……データベース全般に関する運用管理，障害回復などを行う．
4) IQF (Information Query Facility) ……端末ディスプレイ装置より，データベースの問い合わせや報告書の作成を行う．

このように InfoExec は，セマンティック・データベースを中心とした統合化された開発・運用・利用環境を提供しており，InfoExec マネジャと呼ばれるソフトウェアにより表示されるメニュー画面から自分のやりたい仕事を選ぶだけで，これらの多くのソフトウェアを，あたかも一つのソフトウェアであるかのように使用することができる．

操作はすべて対話形式で行うことができる．表示されるすべての画面の形式は，人間工学に基づく標準の画面形式に統一されており，はじめての人でも容易に操作を行うことができる．

本稿では，InfoExec の中核をなすデータベース管理システム SIM で実現されているセマンティック・データモデルの概念を通して，新しいデータベースの設計，検索方法の一端を紹介する．

2. データモデル

現実の世界をコンピュータ化するためには，現実の世界をすべての人が理解できるように抽象化する必要がある．データモデルは抽象化するための方法を提供するものであり，各々のデータモデルによって抽象化の方法も異なる．

E. F. Codd は数学による関係をデータモデルに適用し，リレーショナル・モデル[1]を発表したが，これ以降にも多くのデータモデルが発表されている．

代表的なモデルとしては，1976 年 P. P. Chen が発表した E-R モデル[2]，1979 年リレーショナル・モデルの生みの親である Codd が発表した RM/T モデル[3]，そして 1981 年 M. Hammer と D. Mcleod が発表した SDM[4]がある．

このうち，E-R モデルはデータベースの世界よりも，むしろシステム設計の分野で有名である．SDM は UNISYS の SIM の基礎になったモデルである．これらのモデルは，それ以前のモデルでは表現できなかったことが表現できるようになっている．

3. エンティティ

抽象化しようとする興味の対象の最小単位をエンティティと呼び，同じ特徴を持ったエンティティの集まりをエンティティ型と呼ぶ．エンティティは SIM の中で一番重要な概念で，実体を持った対象を指す．たとえば，ある企業のデータベースを考える．

山口という社員はエンティティであり，浅井という社員もエンティティである．さらにこの二人は共にある企業の従業員だという同じ特徴を持っているので，従業員というエンティティ型にまとめられる．つまり，同じ特徴を持つエンティティは一つのエンティティ型にまとめることができるのである（図 1）．

SIM では，エンティティ型をクラスと呼ぶ論理構造体と考える．

4. クラスとサブクラス

一般に乗物と呼ばれるものを考えてみると，空を飛ぶ乗物，水上を進む乗物，水中を潜る乗物，陸を走る乗物がある．つまり，乗物はそれらの総称であると言える．さらに空を飛ぶ乗物には，ジェット機やプロペラ機という乗物がある．

図2はこの関係を階層構造で表したものである．乗物から始まるこの階層構造において，上位構造は下位構造を総称したものになっている．別な考え方をすると，乗物の役割として空を飛ぶ役割，水上を進む役割等があると考えられる．さらに，空を飛ぶ乗物の役割としてはジェット機としての役割やプロペラ機としての役割がある．

図2において下位構造は上位構造の役割であるとも考えられる．一般的にはこの階層構造は汎化階層構造 (Generalization Hierarchy) と言われ，SIM ではサブクラス関係と呼ぶ．また，この関係は“ISA”の関係とも呼ばれている．

図2における円はエンティティ型を表している．SIM ではこの各々の円がクラスに対応する．さらに，階層構造を表現するために，その頂点にある乗物クラスをスーパクラスと呼び，その下位にあるクラスをサブクラスと呼ぶ．したがって，空を飛ぶものは乗物のサブクラスであり，ジェット機は空を飛ぶもののサブクラスである．

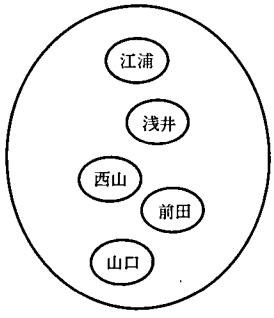


図1　エンティティとエンティティ型
Fig. 1　Entity and entity type

サブクラス関係で重要なことは，X 社のジェット機“X 474”というエンティティは乗物クラス，空を飛ぶもののクラス，ジェット機クラスのすべてに属しているということである．

4.1　サブクラス関係における属性

図2の各々のクラスに属性を割り当てると，名称・全長・全幅・全高・重量・定員等は，すべてのエンティティにあてはまる属性である．しかし，巡航速度や巡航高度は空を飛ぶもののクラスにあてはまるが，陸を走るもののクラスにあてはめると，的外れな属性になる．

このことからもわかるように，属性にはすべてのエンティティにあてはまる属性と，特定のエンティティにあてはまる属性がある．すべてのエンティティにあてはまる属性をすべてのクラスにあてはめると，その属性は冗長性のある属性となってしまい，そこに割り当てられる値は，メモリ，ディスク容量を浪費する結果になる．

そこで SIM では，サブクラス関係における上位のクラスの属性は，その下位のサブクラスに継承されるという考えを持つ．したがって，すべてのエンティティに共通する属性は，その階層構造における最上位のクラスの属性とすればよい．たとえば，名称・全長・全幅・全高・重量・定員等は，乗物クラスの属性とすることによって，下位のすべてのサブクラスに継承されて参照できることになる．

SIM ではクラス固有の属性を直接属性と呼び，上位のクラスから下位のクラスへ継承される属性

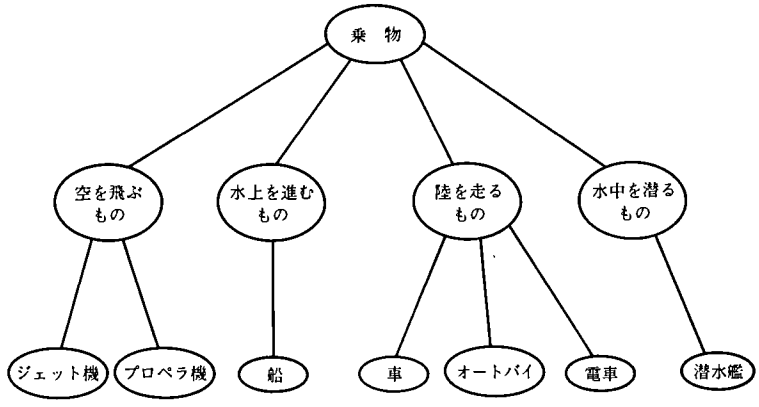


図2　エンティティ型とその総称
Fig. 2　Entity type and generic name

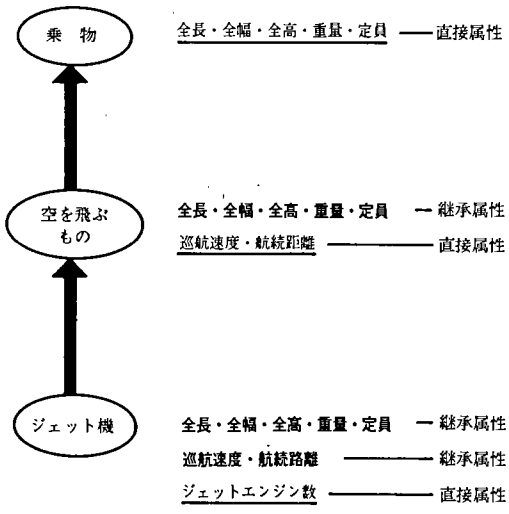


図3 サブクラス関係と属性
Fig. 3 Subclass and attributes

を継承属性と呼ぶ（図3）.

サブクラスからその上位のクラスの属性は参照できるが，上位のクラスから，その下位のサブクラスの属性は参照することはできない．これは自然なことである．乗物としてジェット機のことを議論している場合には，その乗物がジェット機かどうかは問題であるが，その乗物のジェット機としての属性，たとえば，巡航速度，巡航高度等は対象外である．しかしジェット機としてある乗物を議論するならば，その乗物の重量・定員等，すべての乗物にあてはまる属性だけでなく，ジェット機が持つ属性も対象になるはずである．

今までのデータベースでは，汎化階層構造をデータベースで表現するためには，特別な方法を用いて表現していたが[5]，SIMのサブクラス関係を用いれば簡単に表現できる．さらに，これ以降で述べるデータ操作においても，サブクラス関係のエンティティ検索を行うことができる．

5. 属性の種類

空を飛んでいるジェット機を説明するためには，名称・全長・巡航速度・巡航高度を述べて説明する方法もあるが，そのジェット機の写真や絵を見せたり，空港へ行って指でさす方法もある．SIMでは，エンティティを説明するための手段として属性があると考える．今までのデータベースの主キーはレコードや行を説明するためというよりも，やはり他のレコードや行と区別し識別するためであると考えられるので，SIMでは必要ない属性である．

今までのデータベースにおいて“属性”と言えば名称・全長・全幅・全高・重量・定員等の端末に入出力したり，プリンタに出力したりすることができる数字や文字のことであった．あるエンティティだけを説明するのであれば，このような表示可能な値だけで十分である．しかし，現実の世界は非常に複雑で各々のエンティティは密接に関連している．そのため，表示可能な値だけでエンティティ間の関連を説明することはむずかしい．

SIMではエンティティとエンティティの関連を説明するために，属性の値として表示可能な値だけでなく，エンティティそのものを値として持つことが可能である．属性の値としてエンティティそのものを持つことによって，エンティティ間の関係を表現することができるのである．

SIMでは表示可能な値を持つ属性をデータ値属性（DVA）と呼び，エンティティを値として持つ属性をエンティティ値属性（EVA）と呼ぶ．

5.1 エンティティ値属性（EVA）

EVAを持つことによって，あるエンティティは，自分のクラスのエンティティだけではなく，他のクラスのエンティティと関係を付けることができる．

人間の親子関係を考えると，個人個人は一つのエンティティであり，人間というエンティティ型の構成要素である．人には親があり子供がある．人の親も人であり，人の子供も人であるので，これは人間というクラス内の関係である（図4）.

ある企業の社員と部門の関係は，二つのクラスの間での関係である．社員が所属している部門は社員クラスのEVAの値として，その社員が所属している部門クラスのエンティティを持つことによって表現できる．

エンティティ間の関係は，一つのエンティティが必ず一つのエンティティと関係があるとは限らない．一つのエンティティが複数のエンティティと関係がある場合がある．たとえば，ある部門に所属している社員を考えると，部門には複数の社員が所属している．この事実は，一つのエンティティが複数のエンティティに関係していることを示している．

このように，一つの属性として複数の値を同時

に持つ属性を複数値属性と呼び，同時に一つの値しか持たない属性を単数値属性と呼ぶ．複数値属性はEVAと組み合わせることによって，今までのデータモデルではむずかしかった1：Mの関係を簡単に表現できる（図5）．

EVAを通して関係が付けられているエンティティの持つ属性も継承属性と同様に，あたかも自分のクラスの属性のように参照できるので，今までのデータベースのように利用者が明示的に検索する必要がない．図6に社員と部門の関係をSIMの表現で表す．Emp-deptはEMPLOYEEクラスのEVAであり，Dept-empはDEPARTMENTクラスのEVAである．表1に各々のクラスで参照できる属性を示す．

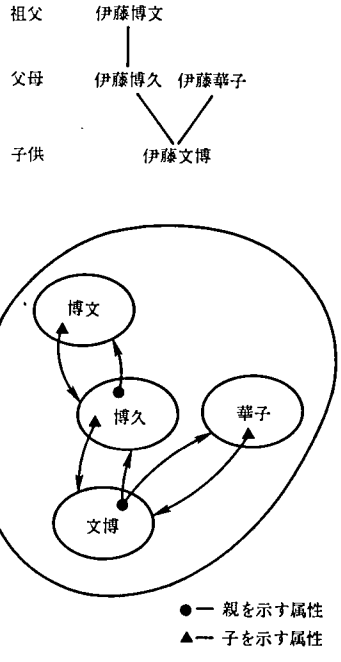


図4　親子関係
Fig. 4　Relationship among family

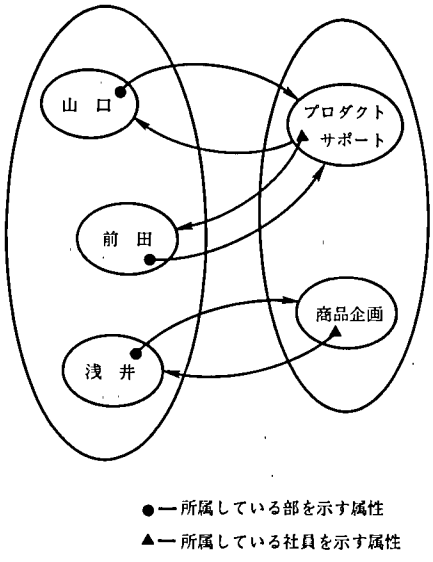


図5　社員と部門の関係
Fig. 5　Association among employee and department

5.2 逆関係

社員と部門の関係において，山口という社員がプロダクトサポート本部に所属しているという事実があるならば，その逆関係は必ず成り立つ．つまり，プロダクトサポート本部には山口という社員が所属していることも事実でなければならない．もしその逆関係がデータベース上で成り立たなければ，データベース内に誤った情報が存在することになる．

SIMではエンティティとエンティティの関係を表現できるため，そこに生じる制約も表現できる．逆関係もその制約の一つであり，データベース管理システムによって保証される．

5.3 制約条件

通常データベースには，格納されるデータが論理的に正しいことを保証するために整合性制約条件が備わっている．表2にSIMにおける整合性制

表1　図6で参照可能な属性
Table 1　Attributes can be referenced in Fig. 6

クラス名	属性 (DVA)
EMPLOYEE	Name, Address Title of Emp-dept
DEPARTMENT	Title Name of Dept-emp Address of Dept-emp

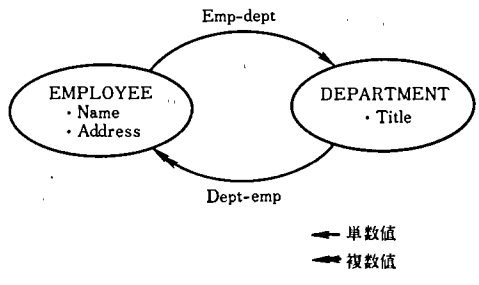


図6　SIMにおける社員と部門の関係
Fig. 6　Association among employee and department by SIM

表2 SIMにおける整合性制約条件
Table 2 Integrity constraint on SIM

UNIQUE	同一クラス内で同一の値は存在できない
REQUIRED	ナル値を持つことが許されない 必ず値が割り当てられなければならない
DISTINCT	一つのエンティティ内で複数値属性のDVAに同一値が存在できない
MAX	複数値属性の値の最大個数を指定する
INITIALVALUE	エンティティ作成時に値が割り当てられない場合システムが自動的に与える値を指定する
INVERSE	EVAで関係付けられているエンティティの逆関係を保証する
部分範囲指定	属性の持ち得る値や値の範囲を指定する

約条件を示す．今までのデータベースにおける整合性制約条件はSIMにおけるDVAに対する条件に対応する．SIMにはDVA以外にもEVAがあるので，これに対する整合性制約条件もある．

たとえば，EVAを用いて親子関係を表すと，そこにはある制約が存在する．一人の人間の親は父親と母親の二人であるので，一人の人間から最大二人の人間へ親の関係が存在する．しかし3人以上に親の関係があれば，それは誤った関係があり，誤った情報がデータベース内に存在することになる．人間の実の親は二人いるので親の関係を示すEVAは複数値属性である．しかし一つのEVAが持ち得る値，つまり関係は2個までである．SIMでは属性が持ち得る値の最大個数を指定することができる．

EVAが単数値属性であれば二つ以上，複数値属性であり最大個数が指定されていると，その値を越える関係を付けようとすると，データベース管理システムによりエラーが返される．

SIMでは整合性制約条件に違反するような情報がデータベースに格納されようとすると，データベース管理システムがどの属性が，どのように誤っているかを利用者に知らせる．その誤った情報はデータベースに格納されない．

今までのデータベースでは，MAX, INVERSE等のEVAの整合性制約として指定できる情報の検査は，データベースにアクセスする利用者プログラムの役割であった．したがって，データベース内に誤った情報が存在する可能性があった．

6. データ操作

SIMのデータ操作はAlgol, Cobol 74の言語を用いて行う方法と，IQF (Interractive Query Facility)を用いる方法がある．ここでは，とくにCobol 74を用いて説明する．データ操作は大きく更新操作と検索操作に分けられるが，SIMではデータ操作の意味が重要であり，どのクラスについて操作するのかが非常に重要な意味を持っている．SIMではデータ操作の中心となるクラスを視点クラスと呼ぶ．

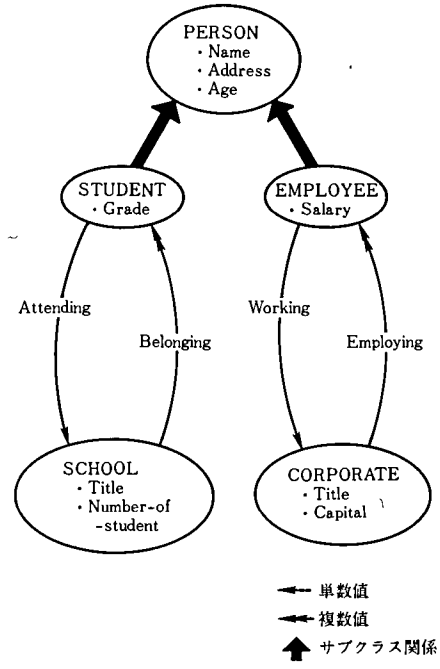


図7 データ操作における例
Fig. 7 Example for manipulation

データ操作を説明するために，一つの例をあげる．学生と学校，会社員と会社の関係をSIMで表現したのが図7である．学生も会社員も一つのク

ラスであり，かつ人間というスーパクラスのサブクラスである．表3に各々のクラスから参照できる属性を示す．

SIMの操作命令は数種類の命令文の組み合わせによって構成される．表4に基本的な命令文を示す．

6.1 エンティティの作成

このデータベースの情報として対象になる人を考えると，学生，会社員，それ以外の人の三つのパターンがある．SIMでは，サブクラス関係を持つエンティティの作成にも継承属性の考えが生きているので，どのサブクラスを視点クラスとして作成するかが重要である．サブクラスでエンティティを作成した場合には，そのサブクラスの上位にあるすべてのクラスのエンティティも同時に作成される．

たとえば，STUDENTのクラスでエンティティを作成すると，同時にPERSONクラスのエンティティも作成される．つまり，学生は学生である前に一人の人間であるということがこれによって表現されているのである．

したがって学生のエンティティを作成するために，まずPERSONクラスのエンティティを作成し，その後STUDENTクラスのエンティティを作成する必要はない．

GradeはSTUDENTクラスの直接属性であるので，PERSONクラスからもEMPLOYEEクラスからも参照することができない．したがってSTUDENTクラスから作成する以外には，値を割り当てることができない．

エンティティをPERSONクラスで作成すると，そのエンティティはSTUDENTクラスにもEMPLOYEEクラスにも属さない．これは，学生でも会社員でもない人のことを表している．図8に作成の例を示す．

サブクラス関係において，学生でも会社員でもない人が，学校に入学したとする．この人は，すでにPERSONクラスのエンティティであり，STUDENTクラスで作成することは，その人のエンティティを重複して作成することになる．そこでSIMでは，すでに存在する上位クラスのエンティティに対して，下位クラスのエンティティを追加することができる．図9に例を示す．

6.2 エンティティの削除

エンティティの削除も作成と同様にどのクラスで削除するかが重要であり，意味を持っている．

表3　図7で参照可能な属性
Table 3　Attributes can be referenced in Fig. 7

クラス名	属性 (DVA)
PERSON	Name, Address, Age
STUDENTS	Name, Address, Age, Grade Title of Attending Number-of-students of Attending
EMPLOYEE	Name, Address, Age, Salary Title of Working Capital of Working
SCHOOL	Title, Number-of-student Name of Belonging Address of Belonging Age of Belonging Grade of Belonging
CORPORATE	Title, Capital Name of Employing Address of Employing Age of Employing Salary of Employing

表4　SIMにおけるデータ操作命令文
Table 4　Statements to manipulate data on SIM

INSERT	エンティティを作成する
MODIFY	エンティティを変更する
DELETE	エンティティを削除する
ASSIGN	エンティティの属性に値を与える INSERT命令文とMODIFY命令文で使用できる
SELECT	エンティティの値を検索するための，視点クラス属性，検索条件を指定する選択命令文
RETRIEVE	SELECT命令文で選択されたエンティティの値を問い合わせ変数に取り込む

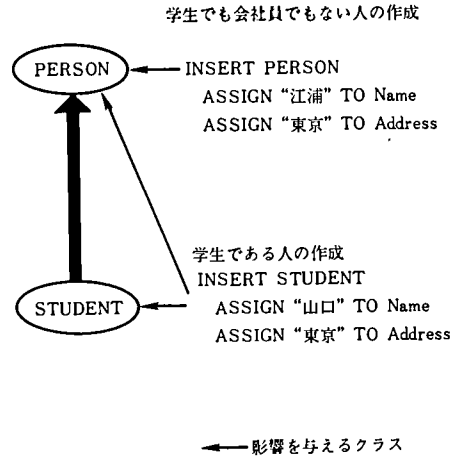


図8　エンティティの作成
Fig. 8　Create an entity

たとえば，ある人が学校を卒業した場合，その人は学生でも会社員でもないただの人になるため，学生としてのその人の情報はもはや必要でない．STUDENT クラスで削除すると，学生としての情報だけが削除され，人としての情報は残る．つまり学生として必要だった学年という情報は必要ないので削除されるが，人としての名前・住所・年令は残るということである．

PERSON クラスでエンティティを削除すると，その人の情報は，このデータベースにおいてもはや必要のないことを示しているので，その人のエンティティはデータベース上から削除される．図 10 に削除の例を示す．

学校を卒業し会社に就職した人の場合，STUDENT クラスでその人のエンティティを削除し，その人のエンティティを EMPLOYEE クラスに追加することによって表現できる．

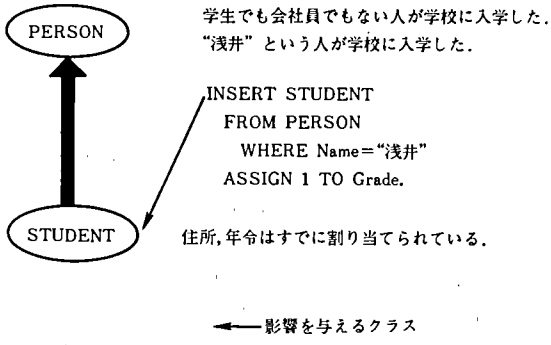


図9 下位クラスから上位クラスを作成する
Fig. 9 Create an entity from an existing entity

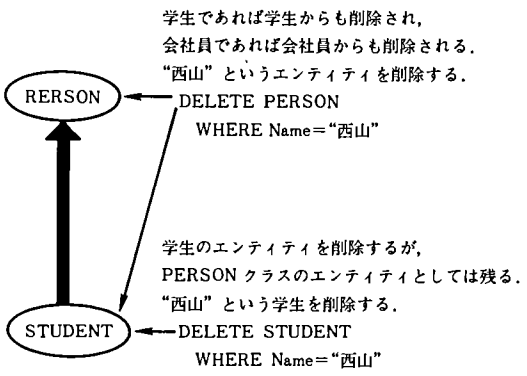


図10 サブクラス関係のエンティティ削除
Fig. 10 Delete an entitiy of subclass

6.3 エンティティの変更

エンティティの属性値を変更する場合にも，継承属性の考えがあてはまる．学生の住所が変更された場合には STUDENT クラスで変更し，スーパクラスの PERSON クラスで変更する必要はない．会社員の住所が変更された場合には，EMPLOYEE クラスで変更する．エンティティの作成と同様に Grade は，PERSON クラスや EMPLOYEE クラスを視点クラスにした変更では値を割り当てることができない．図 11 に変更の例を示す．

6.4 エンティティの関係付け

ある人が会社に就職したとすると，EMPLOYEE と CORPORATE の間に関係を付けなければいけない．この二つのクラスには，Working と Employing という二つの逆関係の EVA が存在するので，一方向の関係を付ければ，その逆方向の関係は SIM によって自動的に保守される．図 12 にこの例を示す．

6.5 一括処理

エンティティの削除，変更は複数のエンティティに対して同時に行うことが可能である．さらに対象となるエンティティの個数を指定したり，実行後，対象となったエンティティの個数を知ることも可能である．図 13 に一括処理の例を示す．

6.6 エンティティの検索

われわれがある情報を得ようとする時には，常に何かを中心に情報を集める．たとえば，学生の

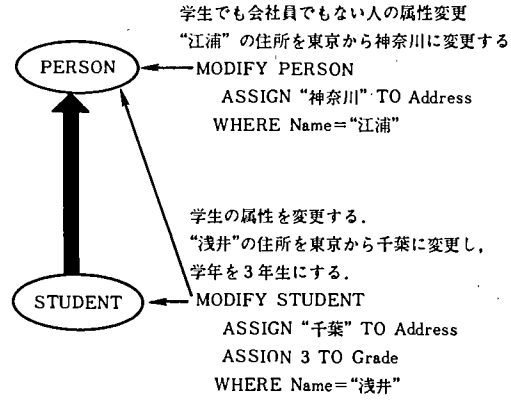
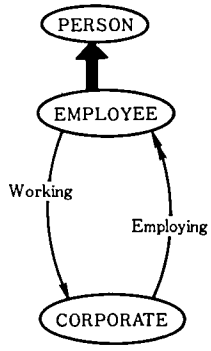


図11 サブクラス関係のエンティティ変更
Fig. 11 Modify an entity of subclass



会社員のエンティティを作成し，
会社との関係を付ける．
“前田”という人がUNISYSに就職した．

```
INSERT EMPLOYEE
 FROM PERSON WHERE Name="前田"
  ASSIGN CORPORATE WITH
          Title="UNISYS"
        TO Working.
```

Employing は Working の逆関係であるので，
SIM によって逆関係は保守される．

図12　エンティティ間の関係付け

Fig. 12　Relate among entities

・年令が65才以上のすべての会社員を削除する．

```
DELETE (NO LIMIT) EMPLOYEE
    WHERE Age >=65.
```

・データベースに格納されている人の年令を一つ加え，変更したエンティティ数を知る．

```
MODIFY (NO LIMIT) PERSON
    ASSIGN Age+1 TO Age.
DISPLAY DMUPDATECOUNT OF DMSTATE.
```

・UNISYS に勤務する年令が30才以下の社員1000人の給与を5万円引き上げる，
対象が1000人を越える場合には，更新は行わない．

```
MODIFY (LIMIT=1000) EMPLOYEE
    ASSIGN Salary+50000 TO Salary
  WHERE Age <=30 AND
        Title OF Working="UNISYS"
```

図13　一括処理の例

Fig. 13　Set operations

名前を知りたい場合には，学生という視点で考えるし，ある会社に勤務する社員の名前を知りたい場合には，会社に視点をおいて，そこに勤務している社員の名前を知ろうとするはずである．

SIMでもこれと同様に検索したい情報によって，その視点を考えることが必要である．しかし，ある情報を別の視点で見た場合にも，必ず同じ情報が集められるように，SIMでも，一つの検索方法で得られた情報は視点クラスを変えて検索しても同じ情報が検索できる．

SIMには直接属性・継承属性・拡張属性があるので，一つのクラスから他のクラスの属性も参照できる．しかし，実際に検索の対象になる属性はその内のいくつかの属性である．そこで検索の対象となる属性の値を返す領域をプログラム内に確保し，余分な属性の値を検索しないようにする．この領域は問い合わせ変数と呼ばれ，SQLのカーソルのような働きをする．カーソルと異ることは，検索のための選択条件が実行時に指定でき，何回も条件を変えて検索できることである．

今までのデータベースでは複雑だと思われる検索も，継承属性，拡張属性という属性を使うことによってSIMでは簡単に検索できる．

SELECT命令文によって検索したい視点クラス・属性・検索条件を指定することによって，いくつかのエンティティが選択される．選択されたエンティティの属性値は，RETRIEVE命令文によってエンティティごとにプログラムの問い合わせ変数に取り込まれる．

図14に学生の名前・学年・在学校名を検索する

例を示す．STUDENT を視点クラスにして，継承属性である名前・直接属性の学年・拡張属性の在学校名を検索する例である．

図 15 には UNISYS に勤務し，給与が 40 万円以上で，住所が東京にある人を検索する例である．例の(A)では EMPLOYEE クラスを視点クラスとした検索例で，例の(B)では CORPORATE クラスを視点クラスとする検索例である．

図 16 と図 17 には，会社別の会社員一覧表を検索する例が示してある．図 16 は EMPLOYEE を

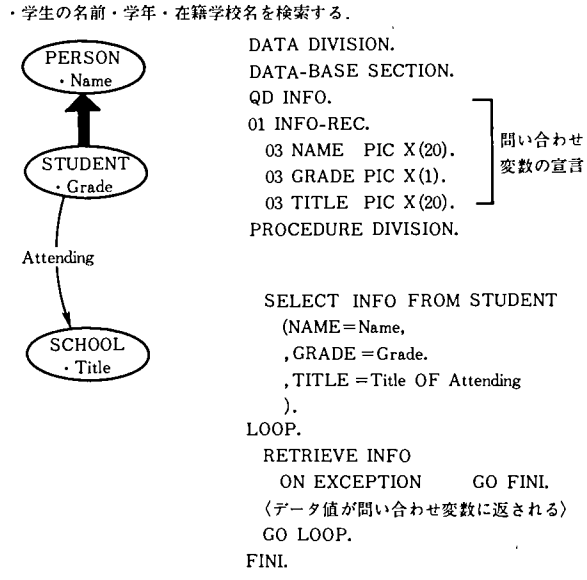


図 14　直接属性と継承属性の検索

Fig. 14　Retrieve direct and inherited attributes

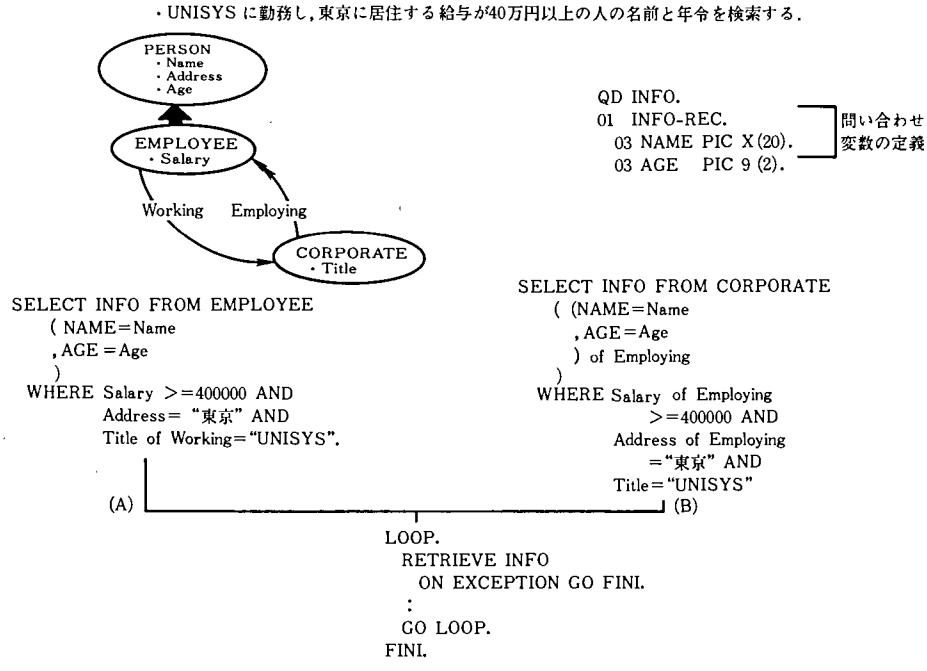
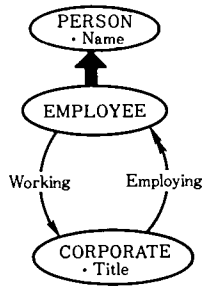


図 15　異なる視点クラスでの検索

Fig. 15　Retrieve through different perspective classes

・会社別，社員一覧表




```
QD INFO.
01  INFO-REC.
  03 TITLE  PIC X(10).
  03 NAME   PIC X(10).
```

```
SELECT INFO FROM EMPLOYEE
  (TITLE=Title of Working
  , NAME=Name
  )  ORDER BY Title of Working.
LOOP.
  RETRIEVE INFO
    ON  EXCEPTION GO FINI.
  :〈会社員のエンティティを一件検索〉
  GO LOOP.
FINI.
```

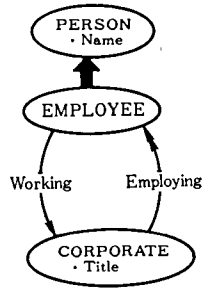
検索結果例

```
UNISYS      山口    (Retrieve INFO)
UNISYS      浅井    (Retrieve INFO)
UNISYS      西山    (Retrieve INFO)
XYZ株式会社  中屋敷  (Retrieve INFO)
XYZ株式会社  岸本    (Retrieve INFO)
XYA株式会社  高井    (Retrieve INFO)
(株) LMN    前田    (Retrieve INFO)
```

図16　SELECT命令文による検索

Fig. 16　Retrieve with select statement

・会社別，社員一覧表．




```
QD INFO.
01  INFO-REC.
  03 TITLE PIC X(10).
QD EMP.
01  EMP-REC.
  03 NAME  PIC X(10).
```

```
SELECT INFO FROM CORPORATE
   ( TITLE=Title
   ,SELECT EMP FROM Employing
      (NAME=Name)
   ).
LOOP.
   RETRIEVE INFO
     ON EXCEPTION GO FINI.
  :〈会社のエンティティを一件検索〉
EMP-LOOP.
   RETRIEVE EMP
     ON EXCEPTION GO LOOP.
  :〈会社に関係する社員のエンティティを検索〉
   GO EMP-LOOP.
FINI.
```

検索結果例

```
UNISYS       (Retrieve INFO)
  山口         (Retrieve EMP)
  浅井         (Retrieve EMP)
  西山         (Retrieve EMP)
XYZ株式会社  (Retrieve INFO)
  中屋敷       (Retrieve EMP)
  岸本         (Retrieve EMP)
  高井         (Retrieve EMP)
(株) LMN     (Retrieve INFO)
  前田         (Retrieve EMP)
```

図17　埋め込みSELECT命令文による検索

Fig. 17　Retrieve with embedded select statement

視点クラスとし，EVA で関係している会社名順に検索している．図 17 では埋め込み SELECT 命令文を用いて，CORPORATE を視点クラスとし，一つの会社に勤務している会社員を検索している．

今までのデータベースでは，拡張属性によって参照できる属性の検索と同様の検索を行うことはむずかしく，誤った情報が検索される可能性もあった．

EVA で関係が付けられていないクラス間を関係付けて検索したいときには，リレーショナル・データベースのように，実行時に動的にクラスとクラスの属性を用いて突き合わせることもできる．

7. おわりに

SIM は，セマンティック・データモデルを実現することによって，より多くの意味，つまり物事の関係や制約を表現できるようになっている．したがって，より忠実に現実の世界をデータベースとして表現できる．データ操作においても，その意味通りのデータ操作ができる．

プログラム開発において，時間がかかり，バグが発生する可能性が高いのがデータ検証のロジックである．SIM にアクセスするプログラムは，その豊富な整合性制約条件によってデータ検証ロジックが大幅に減少する．そのためプログラム開発の時間が短縮され，精度が向上する．データベースには正しい，整合性のとれた情報だけが存在することになる．

最後に SIM と，今までのデータベースと比較した利点を上げる．

1) 自然なデータベースが設計できる．
2) 主キーや外部キーのような人工的な属性がない．
3) データの重複が発生しない．
4) エンティティ間関係が容易に表現・検索できる．
5) プログラムの関係，保守が容易である．

参考文献

[1] E. F. Codd, "A Relational Model of Large Shared Data Banks", Communication of the ACM, Vol.13 No.6, Jun., 1970.

[2] P. P. Chen, "The Entity - Relationship Model——Toward Unified View of Data", ACM Transaction on Database Systems, Vol.1, No.1, Mar., 1976.

[3] E. F. Codd, "Entending The Database Relational Model to Capture More meaning", ACM Transaction on Database Systems, Vol.4, No.4, Dec., 1979.

[4] M. Hammer, D. Mcleod, "Database Description with SDM : A semantic Database Model", ACM Transaction on Database Systems, Vol.6, No.3, Sept., 1981.

[5] J. Smith, D. Smith, "Database Abstractions : Aggregation and Generalization", ACM Transaction on Database Systems, Vol.2, No.2, Jun., 1977.

(システムプロダクト一部)

(財)光産業技術振興協会　監修

光ディスク技術ハンドブック

日経マグロウヒル社，A 4 判，v＋181 pp.，1987，18,000 円

情報処理技術の著しい進歩によって，多種多様な情報処理が高速に行えるようになり，それにともない膨大な情報を蓄えられるメモリが必要になってきた．コンピュータが誕生して 40 年余経過した今日に至るまでメモリ・デバイスは，記憶容量，速度，信頼性，価格，使いやすさ等を追い求め，多くのデバイスが開発されてきた．外部記憶装置としては，磁気ディスク，磁気テープ等，機械的運動を伴う磁気記録が長い年月主流の座を占めている．数年前に，反射膜に $\lambda/4$ の凹凸をつけ，これに絞り込まれた光ビームを当て，その反射率の差によって，データ"1"，"0"に対応させる光ディスクが登場し，CD（コンパクト・ディスク）やビデオ・ディスクへの応用で従来のカセット・テープやレコードをしのぐ普及をもたらした．光ディスクは，ビット当たりの記憶占有面積が $1\mu m^2$ と磁気ディスクの 1/50 にすぎず，現存するメモリ・デバイス中で最も高記録密度性をもち，130 mm ディスク片面に 600 M バイト記録できる可換型媒体であることから，コンピュータのメモリ・デバイスとしての進出も期待されている．

メモリ技術の進歩が著しいこともあって，メモリ・デバイスに関する専門書は非常に少ない．とくに，光ディスクは誕生して，日も浅いこともあって，学会での論文は数多くあるものの，光ディスクのハードウェアをまとめた専門書はなく，現状において本書は貴重な存在である．光ディスクは，光記録材料，半導体レーザ，光ヘッド，フォーカス・サーボ，トラッキング・サーボ，信号処理などの新しい技術分野に広くまたがって構成されている．本書は，これらの分野の第一線の専門家が分担して執筆しており，現時点での最新技術を専門的に掘り下げ，よく書かれているユニークな専門書である．光ディスクに携る技術者にとって，専門分野の周辺技術を理解するのにたいへん役立つ書物となろう．反面，初心者にはむずかしく感じるかもしれない．

第 1 章は光ディスクの発展の経緯を中心に，各種光ディスクの方式を概観しており，光ディスク技術の変遷と各種記録材料についての概要を知ることができる．また，光ディスクの製造工程と光ディスク駆動装置のフォーカス・サーボ技術の概要についても記述されている．この点は，初心者にも光ディスクの概要が理解できるように配慮されたものと思われるが，前述したように光ディスクは広い範囲にわたり新しい分野の技術が取り入れられて構成されているので，初心者が第 2 章以降を理解する手助けとするためには，光ディスクの「読み／書き」の基本原理など光ディスク全般の基礎原理概要の記述があればと思う．

第 2 章は，光ディスクの記録材料に関し，現在実用化されつつある光磁気記録材料，まだ研究中で実用化に至っていない相変化や有機記録材料に至るまで，材料の物理的性質のみにとどまらず信頼性やオーバーライトなどさまざまな物性に言及しており，光ディスクに携る技術者にとって非常に役立つところである．反面，現在実用化されている再生専用型，追記型の記録材料についての記述が少ない．これらは，実用化されてはいるものの速度などの点で今後の改善が必要になると考えられることから，これらの記録材料についての言及が欲しいものである．

第 3 章は，光ディスク装置側での技術，すなわち半導体レーザ，光ヘッド，フォーカス・サーボ，トラッキング・サーボ技術について，各種方式の現状を記述しており，各種方式の長短を知るのに役立つところである．

第 4 章は，光ディスクに信号を記録する方式を現在標準化検討されている CD, CD-ROM, CD-I, 130 mm 追記型について，変調方式，トラック・フォーマット，誤り訂正方式を詳細に記述している．記録方式は一種の取り決めであるが，製品化する上で記録データの互換性上，非常に大切なところである．

第 5 章では，光ディスクのコンピュータ外部記憶装置としての位置付けと用途について，磁気ディスク，磁気テープなど競合記憶媒体との多方面からの比較によって，可能性を論じている．コン

ピュータのメモリ・デバイスとしての普及が低迷している状態から抜出すために改善する必要のあるポイントを指摘している．

第6章では，光ディスクの標準化動向と資料を掲載している．

光ディスクは，前述したように記憶容量が大きくランダム・アクセス可能な小型可換媒体であることからAV分野で急速な普及をもたらした．同時に情報処理分野でも魅力のあるメモリ・デバイスであるはずだが，光ディスクが発表されてから数年経過（文書ファイル・システムとして登場してからでも5年）しているが，普及度は今一つである．その理由は多々あろうが，ハードウェア的には，処理速度を磁気ディスクに近付けること，書き換えやオーバーライトを可能にすること，低価格化などが大きな課題としてある．これらは記録材料の改善，半導体レーザの短波長化と高パワー化，光ヘッドの軽量化等々課題は多い．これらは光磁気や相変化型などこれから出現する光ディスクにとどまらず，既存の再生専用型や追記型についても言えることである．現在国内だけでも百余社がメモリ・デバイスとしての光ディスクに関連しており，普及に期待をかけて改善に努力しており，その目途もつきつつある．したがって，記録材料など追記型，再生専用型についても紙面を割くことによってもっと多くの読者の役に立つものとなるであろう．

また，光ディスクは，記録材料，基盤材料，加工技術から半導体レーザ，レンズ，フォーカス・サーボやトラッキング・サーボ技術，情報処理技術等に至る幅広い技術から構成されており，どれ一つをとっても技術的に光ディスク構成上重要な役割を果たしている．それだけに，記録材料やフォーカス・サーボ，トラッキング・サーボ技術に割くのと同等のウエート・バランスが基盤材料，半導体レーザ，レンズ等にも欲しい気がする．

以上，本書についての感想を述べたが，少なくとも光ディスクにとって，最も重要な部分を構成する最新技術についてまとめた書物として，光ディスクに携わる技術者にとっては非常に役立つ書であり，現在の技術水準からみて高水準にあり，他に類をみないメモリ分野における貴重な存在となる書物といえる．

（ハードウェアプロダクト四部　大石完一）

George B. Dantzig 著
Thomas L. Saaty
森口繁一　監訳
奥平耕造，野口悠紀雄　訳

豊かな生活空間　四次元都市の青写真

コンパクト・シティ

日科技連出版社，A5判，viii＋278 pp., 1974，
3500円

まず，1974年発刊になるいささか時を経た本著を紹介しようとする動機について若干説明させて頂きたい．それは，インテリジェントビルにかかわりを持っている．

インテリジェントビル・ムーブメントはビル建設に際して，情報通信，ビルオートメーション，オフィス環境など，従来は個別にアプローチされていた分野への統合した視点の必要性を喚起したことについては功績ありとされてはいるが，その目指す対象と採るべき方法については依然として個々別々に論じられるまま，展開を遂げているのが現状である．

その顕著な例が〔インテリジェント・コンプレックス〕化，つまりビル群，地域・都市（再）開発への展開である．〔頭脳複合都市〕などと言われているが，それは一体どんな代物なのか．インテリジェントビルに携わり，しかもこれをユーザにリコメンドする立場の者にとっては，このターゲット不在は誠に気を病まずにはいられないものであるが，この〔コンパクト・シティ〕が同病の士にいささかなりとも薬効あればと考えた次第である．

「働く人間は，ロボットの働き蜂や働き蟻と違って，何よりもまず人間であり，単に付随的に労働者であるに過ぎない．（中略）人間は生きた魂である．労働条件を人間にとって耐えられるものにすることが必要であり，それを達成することは大したことであるが，しかしそれだけではまだ十分でない．人間は精神的に根なし草の状態で生きてゆくことはできない．根なし草の状態は人間を狂気に追いやり，犯罪的な方向を辿るよう駆り立てる恐れがあり，しかもこの脅威は（中略）出現しつつある世界都市のスラムやバラック街に，現在流れ込んでいる何億という人類全部の上にのしかかっている．」

これは，A. J. Toynbee の"Cities on the Move"

の邦訳"爆発する都市（長谷川松治 訳，社会思想社，1975年10月発刊）"からの引用である．インテリジェント・コンプレックス化の目的の一つは，オフィスに働く人々のためにオフィスワークの生産性の向上とオフィスワーク環境の快適さを向上させ，さらに両者の融合を図ることであり，このことは先のToynbeeの「…大したこと」までの範囲に限り当てはまる．

「都市は複雑にして怪奇である．都市人の内部には反都市と都市礼賛との二つの感情が混在しており，都市に魅せられながら都市に反発しているとして，むしろそれが一般的であるのだ．都市は理論化しにくいし，まして科学的にとらえることはむずかしく，大抵の場合はその断片，破片をとらまえて理論化したと思い，科学したと思い込んでいるにすぎない．はたして統一理論が可能なのであろうか．」

これは前掲書での平良敬一氏による解説からの一節である．〔コンパクト・シティ〕でも，「人がその生活している都市にいかに関連しているかを理解することは，人生のすべてを理解することになるのである」とまで述べられている．どうやら都市を語るためには，人間の生活のあらゆる側面を統合した観点を必要とするもののようである．

ギリシャの建築家，都市計画家であるDr. Constantinos A. Doxiadisは，建築学，都市計画論，交通論，経済学，社会学，心理学，医学，生物学などを互いに関連させ，人間集落の統一的研究を行う新しい学問〔Ekistics〕を創設した．そこでは対象とする人間集落が最小ユニットとしての〔部屋〕から，都市，メトロポリス，メガロポリス，そして地球規模で唯一存在する〔エクメポリス〕にまで展開されている様子が前出の解説に紹介されている．

一方〔コンパクト・シティ〕では，「われわれが従来のやり方で都市を拡張したり再開発したりすることを続けるならば，いろいろな都市の危機がおそらくわれわれを悩まし続けるであろうと警告する」ことから始まり，「われわれの誰もが実際になしうる最善のことは，都市開発のいろいろな設計案を評価する助けとなるような一つの観点を持つことである」との認識の下に，「これから紀元2000年までの間，われわれの社会を特徴づけると思われる社会的受容度，経済的実現可能性および社会経済的諸傾向に合わせて，ほどほどのものになっている」（下線筆者）計画が「全体系的，相関的接近法と呼ぼうとする，都市の再設計のための一つの手法」に従って展開されている．

〔Compact City〕の著者の一人Dantzigは，ORの権威，LPの創始者として音に知れた学者であり，もう一人の著者Saatyも著名な数学者でありORの大家である．この二人が約6年の歳月を費やして未来都市のモデルケースとして世に問うたのが〔Compact City〕であり，わが国OR界の第一人者森口繁一教授監訳により邦訳されたものが〔コンパクト・シティ〕である．

本書は3部17章および結びより構成されており，第1部〔提案〕では都市開発への問題提起と基本原理，都市モデルの提案およびその提案の有用性について，第2部〔オペレーションズ・リサーチと全体系的接近法〕では，提案のベースとなっているシステム分析について，すなわちOR手法導入の必要性と効果，推奨する都市インフラの構造とシステム設計，財務分析，さらには本開発計画の代替案について言及している．第3部〔いくつかの社会的関連事項〕では，コンパクト・シティでの生活全般を想定し，それが一般的都市生活全般に引き合いに出されるテーマ，たとえばある場所へのアクセスのしやすさ，24時間生活反応，都市での健康・医療サービスのあり方，レクリエーションなど，広範な分野にわたり，その有効性や課題が論述されている．これだけでも推測できるように，本著は都市のスプロール(sprawl)化［肥大・空洞化］に正対し，部分的問題解決の弊に陥ることなく，OR手法を駆使してのトータルシステム・アプローチを試みており，一見日常語化された「コンパクト」という語感から見受けがちな軽い感じのものではなく，漏れのない重量感のある試論を展開している．

半径1320メートル，高さ9メートルの背の低い円筒状の構造物を基底階とし，その上に半径が18メートルずつ小さくなる同様の構造物が7層同心円状に重なり，全体で8層の，高さ72メートルよりなる台地状の構造物ができる．この最上面（ビルで言えば屋上）は豊かな緑地となっている．各層の外周部をなす階段状の斜面から内部に向けて，居住用の住宅とアパートが組み込まれる．住宅一戸当たりの敷地面積は，平均540平方メートルとゆったりしている．中心から半径450メートルまでの部分はコアと呼ばれるビジネス，商業，

公共地域で，オフィス，工場(重厚長大型を除く)，倉庫，病院，学校，商店街，ホテルコンベンション施設，コンサートホール，シアター，スポーツ施設，レジャー施設，行政官公庁などが設置される．エネルギー，空気，上・中・下水道，ごみ処理等の都市インフラについては，すべて総合的にシステム化されコンピュータ・システムの支援により制御・監視される．道路，輸送機関，物資の自動集配施設，大量輸送機関（航空機，鉄道等）へのアクセスは，効率よく効果的にシステム化されている．まさに軽薄短小型対応都市と言える．これがコンパクト・シティの第一段階の概要で，この段階で25万人の人口を擁している．コンパクト・シティは，その発展に応じて規模を拡張することができ，200万人を越えるとその利点のいくつかが失われ始めるため，200万人が規模による上限となる．この場合，基底部の半径および層の数ともそれぞれの2倍の，2640メートル，16層となる．ちなみに人口200万人の最大規模のコンパクト・シティの基底部の面積は23平方キロメートルとなるが，これを現在の都市の形での占有面積に換算すると，450平方キロメートル（約20倍）が必要となる．

コンパクト・シティを提案するに際し，二つの原則が提起される．その一つは〔空間の原則〕で，「人口密度を低く保ち，土地の利用を保全し，かつ都市拡大の難点を避けるためには，人間は垂直の次元をより効果的に利用しなければならない」とする．垂直次元の重要さを主張する建築家Paolo Soleriの〔アーコロジー(Arcology)〕に同調しつつも，たとえば都市の高層化や次に述べる時間次元の利用などでは異なる見解を示している．もう一つの原則は〔時間の原則〕で，「空間を保全し，空間の効果的な利用を最大にし，そして生活の煩わしさを減少するためには，人間は都市の施設を1日24時間全般にわたって，より平均して利用することにより昼夜別の症候群から抜け出す必要がある」とするものである．一斉に起き，一斉に働き，一斉に食べ，一斉に寝る人間の習性を〔蝉のリズム〕(cicadian rhythm，人間の概日周期 circadian rhythmに引っ掛けている）と呼んでやゆし，この蝉のリズムの習慣をできるだけ改めて，通勤，就業，食事，などの社会生活に生じるピークの平準化の重要性を説いている．

コンパクト・シティの計画に際しては，全体系的方法（total system approach）が提唱され，部分最適化は，断固拒否する姿勢が貫かれている．計画とは，「多数の代替的可能性を評価した結果」であり，「適応的な自己修正過程」であって，「良い計画とは，新しい条件や新しい方針が生起した場合に，すでに提案されている解決策を敏速に変更するための手続きを組み込んでいなければならない」ものである．このような計画に際してのスタンスを確認した上で，都市設計の六つの目標設定に際し，とくに注意を払うべき地球レベルでの環境保全の問題，都市の全体系モデル作りの進め方，そしてコンパクト・シティ一般に適用した場合の各施設要素ごとの数値の決定が行われる．主要な施設，たとえばコンパクト・シティの躯体，水処理，照明等についての概要が紹介される．さらにコンパクト・シティの建設と運営に関する収支計算，現在の方法と比較しての経済的効用，起債の方法など財務上の事柄についても論述されている．

本書には興味深い挿絵や図解がふんだんに採り入れられており，直観により理解される部分を多く含んでいる．コンパクト・シティそのものが与えてくれる未来都市像に尽きせぬ興味を抱かれるであろうことはもとより，著者達の造詣深い広い視野に導かれての都市開発論の格好の入門書であることを最後に強調しておきたい．

（設備設計部　大嶋重光）

ISSN 0914-9996

技　報
UNISYS TECHNOLOGY REVIEW
Vol. 8　No. 2　(No. 18)

発行日	昭和63年8月31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 東京都港区赤坂2-17-51 〒107 TEL(03)585-4111 (大代表)
頒布価格	1,500円
印刷所	三美印刷株式会社