マイクロコンピュータと
ワークステーション 52－4
（1988. 9. 27）

ＡＡＣ（アドバンスト アプリケーション コンセプト）の構造

浮川和宣
代表取締役

機能の拡張性、操作環境の共通性、および情報資産の継承と可搬性、という３点を骨子とするＡＡＣ構想を支える技術について報告する。ここで紹介するのは、マルチタスキングやマルチウィンドウ環境を提供するシングルユーザ向きのスクリーンシステムである。これは、プログラムモニタとスクリーン管理、そしてアプリケーションにわけられる。

プログラムモニタは、スクリーン管理とアプリケーションの、プログラムの実行とタスクの管理及びメモリ資源の管理を行う。プログラムモニタの上位にあるスクリーン管理は、機種に依存しないアプリケーションの実行環境を提供する。特徴として、マルチスクリーン論理メモリ管理、アプリケーション間通信、あるいはマルチリンガルや発音記号を含むコード体系があげられる。スクリーンの上で動くアプリケーションは、キーボードやメニューのカスタマイズを可能にし、アプリケーションの特殊な形態であるＶＡＦ（Value Added Function）を機能として取り込むことができる。

一方、ドキュメントファイルを共通化し、それによって Grouping、Binding といった応用技術をサポートする。

これらの機能によって、ＡＡＣ構想が実現することを明らかにする。

Structure of the Advanced Application Concept

Kazunori Ukigawa
President
3-46 Okihamahigashi Tokushima-shi Tokushima

We report on the technology supporting AAC (advanced application concept), which offers the extension of function, the common operating environment, and the succession and connectivity of data resources. This technology is the window system with multi-tasking and multi-window environment for single user. The screen system can be classified into three parts; program monitor, window management system, and application.

The program monitor is in charge of executing a program and task for the window management system and the application, as well as the memory managemant. The window management system, lying on the program monitor, offers a running environment for the application. And it also works independently from a hardware. The program monitor features multi-window logical memory management, communication between the applications, and multilingual code system with phonetic symbols. The application running on the multi-window make keyboard and menu customized. And VAF (value added function) works as an add-in software to extend the function for the application.

Furthermore, AAC supports grouping and binding techniques by generalizing a documentation file.

1. ＡＡＣに至る背景

現在主流の１６ビットパーソナルコンピュータは，その標準的なマイクロプロセッサがマルチタスキングをサポートしておらず，シングルタスクマシンである。また，１６ビットパーソナルコンピュータの標準ＯＳ，ＭＳ－ＤＯＳも，マルチタスキングの環境は提供していない。ＭＳ－ＤＯＳ上で動作するパーソナルコンピュータ用アプリケーションソフトウェアも，当然シングルタスクを前提に作られており，さらにワープロソフトなどは，ハードウェアの制限いっぱいのメモリを使用し，空き領域はほとんどない。これは，メモリを増設したとしてもＭＳ－ＤＯＳの制限があるため解決にはならない。

これらの問題を一気に解決しようとするのが，３２ビットパソコンとともに登場したＯＳ／２である。しかし，ＭＳ－ＤＯＳマシンは，すでに日本だけでも３４０万台（１９８７年実績）が厳然として存在し，現実的な問題として，ＭＳ－ＤＯＳの世界でＯＳ／２の環境の実現を望む声は強い。

2. ＡＡＣのねらいとするところ

われわれは，これらの要望を受け入れ，またオフィスでパソコンを１人１台持つ時代の利便を考えて，アプリケーションを作る側からの立場で，ＡＡＣ構想を発表した。ＡＡＣは，機能の拡張性，操作環境の共通性，情報資産の継承と可搬性の３点を骨子とする構想である。これは具体的には，ＭＳ－ＤＯＳ上で，スクリーンシステムを搭載し，マルチタスキング，そしてＭＳ－ＤＯＳが持つメモリ制限を取り払うといった環境を提供しようとするものである。

ここでは，ＡＡＣ構想を支える技術を紹介する。

3. スクリーンシステム

本稿では，ＡＡＣ構想を実現するためのものとして，一つのスクリーンシステムを考案したので紹介する。

３．１ スクリーンシステムの特徴

このスクリーンシステムは，ＭＳ-ＤＯＳ上のシングルユーザーに対し，マルチタスキング，マルチスクリーン環境を提供する。

このスクリーン上のアプリケーションに対する最も顕著な効果としては，機種に依存しない環境の提供とマルチタスキングが挙げられる。ＭＳ－ＤＯＳが成し遂げられなかったハードウェアの差異の吸収を完遂することにより，同じアプリケーションがどのメーカーの機種でも動作するようになる。また，マルチタスキングは個々のプログラムの構成を簡素化する。そして，複数のアプリケーションの共存を可能にし，例えば，あるアプリケーションで作ったデータを，その場で別のアプリケーションにコピーするなどの操作を可能にする。

このスクリーンシステムは，図-１に示すように，プログラムモニタ，スクリーン管理，及びその上で動くアプリケーションから構成される。

プログラムモニタはマルチタスキングの環境を与え，スクリーン管理はオーバーラップ型のマルチスクリーン環境を与えている。ＥＷＳなどで見られる汎用のスクリーンシステムとは異なり，ユーザーインターフェースの優れたアプリケーションを快速に実行することを目的として特殊化されている。

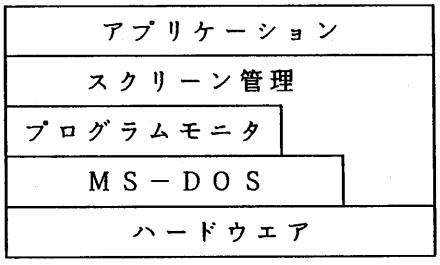

図-1　スクリーンシステムの構成

3．2　プログラムモニタ

プログラムモニタは，プログラムのロード，各タスクの実行，終了，タスク切り替え，コンテキストの管理を行う。また，プログラムのロードに付随して，メモリ資源の管理や拡張メモリの管理も行う。モニタは，タスク管理，メモリ管理，ローダ，スワッパで構成され，マルチタスキングを実現している。だたし，ＡＡＣ構想を実現するために，性能を第一に考えて設計したため，汎用のマルチタスキングを行うＯＳのようなタスク間のプロテクト，エラー処理などは行わない。

（１）　タスク管理

タスク管理では，複数のタスクを管理することができる。ここでは，キーボード入力やマウス入力など，アプリケーションプログラムが他からもらう入力をまとめてイベントと呼ぶ。イベント入力はスクリーンシステムのシステムコールで行う。このマルチタスキングは，タイマー割込みによるタイムシェアリングは行わず，イベント入力時にタスク切り替えが行われる。

（２）　メモリ管理

複数のアプリケーションを実行する場合の最大の問題はメモリ空間の大きさである。ＭＳ－ＤＯＳには６４０ＫＢの制約があるが，アプリケーションで共通のモジュールをスクリーンシステムが持つとともに，仮想メモリの概念を導入し，２次記憶まで利用することによって，この問題の解決を図る。

メモリ管理は，タスクごとにハードウェアメモリ空間を管理して，タスクに対して整合性のあるメモリアクセスを提供する。また，拡張メモリボードにも対応し，これを用いることにより，多くのアプリケーションを実行できるようにしている。

メインメモリはパラグラフ単位で管理する。割り当てられるメモリブロックは，低位のメモリから取るものと，高位のメモリから取るものに分けられる。低位のメモリにはコードセグメントを，高位のメモリにはデータセグメントを置いて使用する。使用中のメモリブロックは必ず隣接するように管理され，空きができた場合はメモリブロックを移動して，空き領域が一つにまとまるようにする。空き領域をまとめる作業はメモリ要求があったときに行う。

拡張メモリは６４ＫＢを１ページとして，ページ内に含まれるメモリブロックと空き領域の管理を行う。空き領域は，どのパラグラフから空きになっているかということだけを管理する。メモリがなくなった場合，必要性の低いセグメントを２次記憶上にスワップアウトする。再度そのセグメントが必要になっても，スワップアウトしておいたものを読み込むことで，シンボル参照の解決などの操作

が省略できるので，速度の向上が望める。

（３）　ローダーとスワッパ

アプリケーションプログラムは，ローダーとスワッパの働きによって，メインメモリ上へロードされる。スワッパは必要なメモリ確保を保証し，ローダーがプログラムをロードする。その特徴として，デマンドロードや外部参照を可能にしている点と，そのプログラムが既にロードされていたときはコード部分を共有化する点が挙げられる。

スクリーンシステム上のプログラムは，セグメント単位で，常駐または非常駐の指定が可能である。セグメントの常駐指定は，本当に必要があるものだけ限る必要があり，たとえば，スクリーン管理や時計といったものである。通常のアプリケーションでは，すべてのセグメントを非常駐指定しておく。その場合でも，デマンドロードもしくは別のアプリケーションの起動によるスワッピングが起こらない限り，必要なプログラムはメインメモリ上に存在するため，性能は落ちない。また，同じアプリケーションのコードを共有化することによって，メインメモリの効率を上げている。

（４）　ポインタ管理

タスク間で簡単なデータのやり取りを行うときにポインタ管理を用いる。ポインタ管理は，あるデータ領域をさすｆａｒポインタ（４バイト）を管理している。アプリケーションは，これに名前をつけることができ，その名前を用いて，タスク間におけるデータの授受を行う。

後述のアプリケーション間通信と比べ，複雑なことはできないが，簡単で扱い易いのが特徴である。

3．３　スクリーン管理

スクリーン管理はアプリケーションと各種の資源を制御する部分の間をとりもち，資源管理を行っている。マルチタスキングを実現するためには，アプリケーションがメモリ，ファイル，ＶＲＡＭなどの資源に直接アクセスすることを禁止しなければならない。即ち，全ての入出力操作はシステムコールを用いる必要がある。

（１）　スクリーン管理

スクリーンシステムでは，複数のタスクの画面管理にオーバーラップ型のマルチスクリーンをサポートし，ユーザーインターフェースにビジュアルなコミュニケーションを図っている。また，これまで，ほとんどのスクリーンシステムではビットマップ表示のみを使っていたが，ハードウェアの性能を考慮し，高速化のためにキャラクタ表示とビットマップ表示を混在して用いることにした。そのために，スクリーンは必ずキャラクタ単位のサイズとなり，表示関数もキャラクタ型とビットマップ型の２種類を持つこととなった。

このスクリーンシステムでは，表示以外においても，ハードウェア，ＢＩＯＳ，ＤＯＳファンクションを直接使うような機能をシステムコールとして用意している。

スクリーン管理はスクリーンシステムのシステムコールを提供し，スクリーンを利用する全てのアプリケーションの実行を管理する。スクリーンシステムのシステムコールの大別を以下に示す。

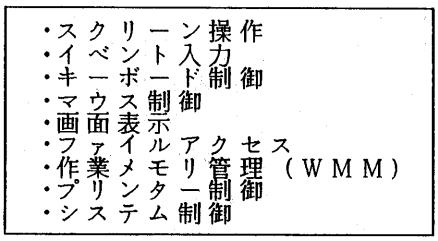

図-2　スクリーンシステムのシステムコール

（２）　Ｍｕｌｔｉ　Ｌｉｎｇｕａｌと発音記号

スクリーンシステム内の標準入出力モジュールは，日本語のみならず，独仏，発音記号の入出力をサポートしている。従って，スクリーンシステム上のアプリケーションは，標準に MultiLingaul 対応となる。これは，独仏発音記号を扱えるような拡張コード体系を持つことによって実現されている。

（３）　ＷＭＭ（Ｗｏｒｋ　Ｍｅｍｏｒｙ　Ｍａｎａｇｅｒ）

ＷＭＭはアプリケーションが可変長のデータを格納，参照及び編集するために使用する作業領域を確保し，提供する作業メモリ管理である。

ＷＭＭは，以下に述べる特長及び機能を持っている。

1）１つの作業領域は，物理的には複数のページを繋げた不連続の領域であるが，アプリケーションからは論理的な連続領域と見なすことができる。その作業領域へのアクセスはＷＭＭに対するサブルーチンコールとして実現される。アプリケーションは作業領域上での論理的な相対位置を指定すれば，作業領域上でのデータの操作を行うことができる。

2）仮想的な作業領域を複数個提供し，それぞれの作業領域は独立性を持っているため，アプリケーションはそれぞれの目的に応じた作業領域を利用することができる。

3）補助記憶上に作業ファイルを作成し，仮想記憶制御を行うことにより，実メモリを越える大量のデータを扱うことができる。

4）作業領域にはデータを圧縮して格納することも可能であり，作業メモリの効率を上げることができる。

（４）　アプリケーション間通信

通信というとネットワークのように複数のハードウェア間で行うことが多いのだが，ここでは，シングルマシンにおけるスクリーン間でのデータのやりとりを扱う。

アプリケーション間で，特定のプロトコルに従ってデータの送受信を行い，様々な処理が可能となる。たとえば，あるアプリケーションに不足している機能を，別のアプリケーションと通信することにより補うといったことが可能となる。つまり，この通信を使うことによって，あたかもそのアプリケーションの機能が追加されたのと同じ結果が得られる。一例を挙げると，あるアプリケーションが１つのファイルしか扱えなかったとしても，この通信を使うことによって異文書間

コピーが可能となる。

複数のアプリケーション間において，リアルタイムのデータのやり取りをすることにより，システムを自動的に動かすことが可能である。リアルタイムのデータの内容は，実際のデータもしくはコマンド列である。

前者は自動更新のデータリンクであり，アプリケーション間のやり取りが可能ならば，必ずしも同じフォーマットである必要はない。そして，自動更新の方法だが，変更があると常にそのデータを渡す場合と，変更があったことのみを通知するのみで，その変更を即座に処理するかどうかはそのアプリケーション任せの場合がある。通常，データの変換・同化に大量のメモリを消費するときに，変更のみを通知する。

後者は，他のアプリケーションのコマンドを自動的に実行させる。例えば，同じアプリケーションが複数起動していたときに，一方のコマンド状態を他方に伝えることによって，あるコマンドを複数のスクリーン間で同時に実現できる。

3．3　アプリケーション

アプリケーションは，スクリーンシステムで用意されたサービスルーチンを用いて入出力処理を行うことにより，ハードウェア・ＯＳからの独立性を保つことができる。

アプリケーションは，メニュー入力やファイル名入力などの共通ユーザーインターフェイスを使用し，ユーザーインターフェースの統一化を計るようにする。しかしながら，その一方で，ユーザーカスタマイズといった機能拡張も行う。ここでは，その例として，キーボード・メニューのカスタマイズと機能のカスタマイズとでもいうべきＶＡＦについて説明する。

（１）　キーカスタマイズとメニューカスタマイズ

これらのカスタマイズは，キーボード・メニューといった情報のテーブルの内容を変更することによって実現される。

キーカスタマイズは，機能に対するキー割付の自由な変更を可能とする。各アプリケーションは，スクリーンシステムから提供されるハードウェアに依存しない仮想キーコードと，アプリケーション固有の機能を結び付けるテーブルを持っている。アプリケーションは，キー入力によって処理を行うとき，直接キーコードを判断するのではなく，このテーブルによって処理を進める。

メニューカスタマイズは，不要な機能をメニューから外し，頻繁に使う機能を選択し易い位置に置くといったことを可能にする。アプリケーションは，階層化されたメニューテーブルを持ち，メニュー実行モジュールにより，このテーブルに登録された処理を実行する。

（２）ＶＡＦ

スクリーンシステム上では，外部プログラムをひとつのコマンドとして取込み，それを実行するような機能を持つアプリケーションを作ることができる。この機能を利用すれば，アプリケーション本体を変更することなく，機能を追加・拡張することができる。ここでいう外部プログラムとは，ある仕様に基づいて作成された実行プログラムのことで，これをＶＡＦ（Value Added Functions）という。

これに対し，ＶＡＦを登録可能なように作成されたアプリケーションのことを

コア・アプリケーション（以下，コア・ＡＰと略）と呼ぶ。

ＶＡＦは，コア・ＡＰに組込まれて初めて実行可能となる。従って，コマンドとしての登録作業が必要になる。登録してしまえば，そのＶＡＦはコア・ＡＰのコマンドとして，他のコマンドとまったく同じように，実行が可能になる。つまり，ＶＡＦによってコアＡＰの機能拡張が可能となる。

コアＡＰとの関係から，ＶＡＦは次の２種類に分類できる。

１）コア・ＡＰから情報・データ等を受け取り，ＶＡＦ内で処理を行い，再び，コア・ＡＰに加工したデータを渡すもの。ＶＡＦによっては，コア・ＡＰから情報・データ等を受け取らず，ＶＡＦ側のみでデータを作り出すかも知れないし，逆に受け取るだけでデータを返さないかも知れないが，とにかくなんらかのインターフェイスを必要とするもの。

２）コア・ＡＰと完全に連動する。ＶＡＦ呼出し後も常駐して，必要に応じて（編集，表示，印刷など）コア・ＡＰから機能呼出しによって動作するもの。単純に機能を呼出して実行するというより，一太郎などのコア・ＡＰの方から画面などと連動して必要な機能が呼出される

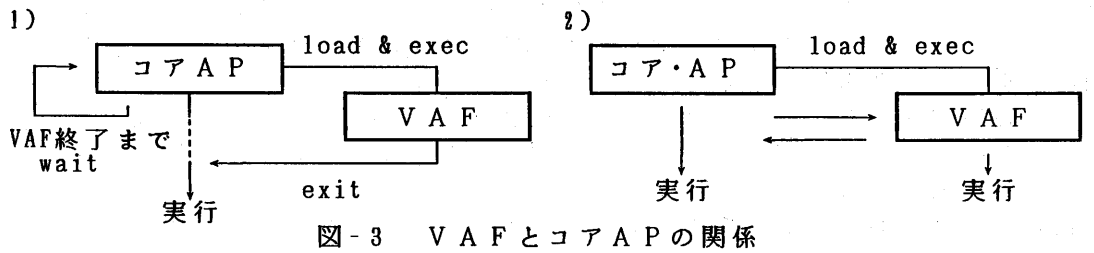

図-３　ＶＡＦとコアＡＰの関係

（１）　共通ドキュメントファイル

これまで，データファイルには様々なフォーマットがあり，それぞれ別々に扱われていた。たとえば，ワープロの文書ファイルと作図の文書ファイルは，ユーザーから見れば１つとして扱いたい場合があるにもかかわらず，それぞれ別々にファイルがあり，コピーなどの操作を何度も繰り返していた。

この共通ドキュメントファイルは，情報の可搬性を重視し，複数のデータファイルのフォーマットを統一化したものである。ワープロの文書ファイルも作図の文書ファイルも，１つとして扱うことができるのである。共通ドキュメントファイルのデータフォーマットは，拡張性に富んでおり，新しいデータファイルフォーマットを定義して追加することも可能である。

（２）　Ｇｒｏｕｐｉｎｇ　Ｆｉｌｅ　Ｓｙｓｔｅｍ

グルーピングファイル指定は，複数の共通ドキュメントファイルをグルーピングし，仮想的に１つの共通ドキュメントファイルとして扱えるようにすることを目的としている。グルーピングされた共通ドキュメントファイルは，単独で編集したり，グルーピングファイルとして編集したりすることが可能である。

これは，ドキュメントファイル間にリンク関係を与えるファイルであるが，共通ドキュメントファイル内のリンク情報データファイルに，ファイル名を記憶す

ることで実現される。

共通ドキュメントファイルが，ファイルデータの可搬性を重視したのに対し，ファイルデータの共有性を重視する。たとえば，ある文書を作成するときに，各章単位で別々の文書を作成する。全文を見たいときは，グルーピングファイルを読み込む操作のみで可能となり，挿入という作業は不要となる。使用する側からみると，各章単位の文書は，単独でしか存在していないにもかかわらず，グルーピングによって，あたかも別のファイルが存在しているようにみえる。グルーピングファイルが複数存在したとき，このことは更に顕著となる。グルーピングファイルの構成を変えることによって，全く新しいファイルを作ることが可能になるのである。

既存のデータを使って，新しいデータを作るといった新しい視点からの情報管理がここにある。

（3） Binding File System

アプリケーション間通信を利用することによって，データをリンクし，リアルタイムにデータを獲得することができることを前に述べたが，それをファイルシステムにも延長したものが Binding File System である。複数のアプリケーションに於て，永続的なリンクを確立することによって，ファイルの内容の変化に対して，自動的に反映させようとするものである。従って，ファイルが変更された場合，その内容についてリンクされているファイル全てが更新される。

グルーピングがファイル単位でしか指定できず，グルーピングファイル自身は内容となるデータを持たないのに対し，バインディングはファイルの一部分に対しても指定でき，それぞれのファイルが内容となるデータを持つ。つまり，バインディングは，一つのデータに対する変更を，それを利用している複数のファイルに自動的反映させるものである。

例えば，カルクとグラフのようなアプリケーションのファイル間でリンクしておくと，カルクのデータを修正しファイルセーブするだけで，自動的にグラフのファイルのデータも修正される。両者を常に自動更新するのであるが，ファイルセーブ時に，一方のアプリケーションが起動していなければ，そのアプリケーションを起動して自動更新が行われる。さらに，他の場所で修正されたデータがコピーされた場合にも対応できるように，ファイル読み込み時にもデータの変更があったかどうかをチェックし，変更が有れば自動更新が行われる。

5. 将来への構想

これまで，ＭＳ－ＤＯＳ上のシングルユーザーといった環境下のみでＡＡＣ構想を考えていたのだが，将来はネットワーク環境下にまで発展させるつもりである。

ネットワーク環境下の共通ドキュメントファイルに関して，以下に紹介する。

ＵＮＩＸマシンをファイルサーバとして，ＡＡＣ構想に基づくソフトウェアの動くマシン（ＡＡＣマシンと呼ぶことにする）は，ＮＦＳ経由でサーバ上の共通ドキュメントファイルにアクセスできるようにする（ＬＡＮ）。

サーバには，一般のファイルサーバ機能に加えて，メールサーバを提供していく。共通ドキュメトファイルの電子メールは文字・文字属性・文書制御情報・図形・イメージを統合的に含んだ，マルチメディア電子メールとなる。機能の面で

も，同報・至急・確認・遅延・転送・回覧・追跡など充実させて，本格的な事務処理の要求に応えていく。

共通ドキュメトファイル電子メールは，サーバ間通信やパソコン通信で，遠隔地とも交換されるようになる（ＷＡＮ）。共通ドキュメトファイルの仕様は，全てのＡＡＣマシンに共通する。現在でも，文書ファイルは機種依存しない構造となっていて，異機種間で文書の交換が可能となっている。ネットワークの上では異機種間の文書の交換が，実際に数多く実行されるようになり，ＡＡＣの可搬性がとくに重要となる。

メールに必要な，相手アドレスなどの情報は，共通ドキュメトファイルの中に新たな属性として取り込まれる。メール以外のＡＰはこの属性を無視して，自分に必要な部分だけを解釈することになる。

ＡＡＣの情報資産の継承という目的をふまえながら，共通ドキュメトファイルの仕様は将来も成長を続ける。電子メール用の拡張もその一つとなる。例えば，文書に変更情報を記憶して，文書の原型から現在に至るまでの履歴がとれるようにする拡張が考えられる。これを実現すると，電子メールで回覧している文書に，原型の情報を損なうことなく，各人がコメントを加えていける。

以上のように，ＡＡＣ構想は一つの環境下にとどまらず，システムが進化していく中においても重要な役割を果たしうるものである。