ＡＳ３０００の開発

4S-1　（１）　ビットマップディスプレイへの漢字表示の実現

吉田祐治　和泉任一　†鈴木達男　‡今村泰介

（（株）東芝　府中工場　†システムソフトウェア技術推進部　‡情報通信システム技術研究所）

1．はじめに

エンジニアリングワークステーションAS3000は、サンマイクロシステムズ社のＳｕｎ－３に、日本語処理機能や、周辺装置レパートリの強化など日本市場向けに機能強化したシステムである。

AS3000においては、かな漢字変換による日本語の入力、日本語の表示、および日本語化されたｕｎｉｘのコマンド、ライブラリの使用が可能である。

ここでは、ＡＳＣＩＩ文字のみ意識して作られた、ビットマップ表示ライブラリに日本語を表示できるようにするための問題点とその実現方式について説明する。

2．日本語化での問題点

ＳＵＮ－３のビットマップディスプレイへの文字表示はpixwin、pixrect という２つの層の文字列処理関数で行われる。

・pixwin層では、ウインドウ管理下でウインドウに対す表示制御を行なう。 pixwinはpixrect を使用する。

・pixrect 層では、フレームバッファなどの矩形イメージ領域に対して、コードをフォントイメージに展開してイメージ転送して表示制御を行なう。

・フォントはvfont と呼ばれる形式のフォントファイルに登録され、フォントオープン時にメモリへローディングされる。

これらを日本語化するには、以下の問題点があった。

2.1　文字セットの相違

オリジナルはＡＳＣＩＩなどの１バイト文字セットのみを意識しており、最大 256文字種までしか扱えない。 日本語は、２バイト文字セットで約9000文字種扱う必要がある。 そのため、オリジナルのフォントファイルやフォント制御データ構造を拡張する必要がある。

また、文字数が多いので、フォントのオープン処理でのフォントファイルからメモリへのローディングの時間が長いという問題がある。

2.2　上位互換性

日本語文字列表示を行う関数群は、基本レベルであるため、可能な限りオリジナルとの互換性を保づように設計しなければならない。 これは、上位ソフトウエアを日本語化する修正量を最小限にするために必要なことである。

3．実現方式

以上の問題点に対して次のような方式により日本語表示を実現した。

3.1　フォントファイル

オリジナルのフォントファイルは、図１のような構成となっている。

漢字フォントファイルは、オリジナルのフォントファイルとの互換性と１ファイル化による高速性を考慮して、図２に示すようなアーカイブファイルとした。 アーカイブファイルは以下の49個のフォントファイルから成る。

①半角英数、半角カナを格納したフォントファイル
（hf0102：１個）

②全角文字２区分を格納したフォントファイル
（kf0001～kf9495：計48個）

hf0102、kfxxxxの個々のファイルは、オリジナルのフォントファイルと互換性があるため、フォントを編集するfonteditなどのユーティリティをそのまま使うことができる。

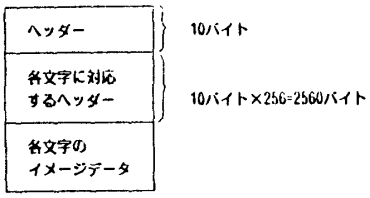

図１　フォントファイルの構成

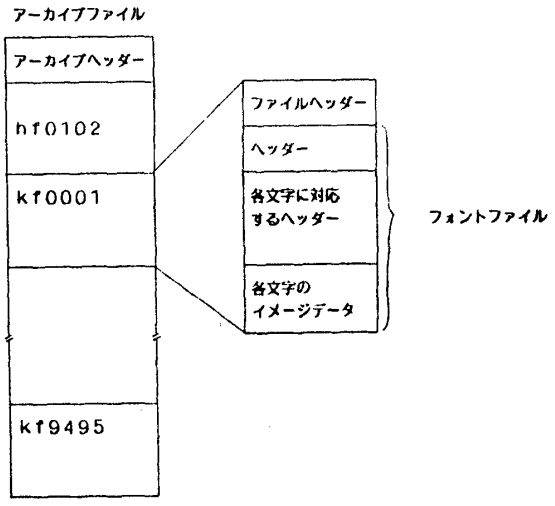

図２　漢字フォントファイルの構成

Development of AS3000
YUUJI YOSHIDA
TOSHIBA CORPORATION

3.2 フォント制御データ構造

オリジナルのフォント制御データ構造は図3のようになっており1文字ずつ独立したpixrect 即ち矩形イメージデータとして管理されている。

漢字のフォント制御データ構造は図4のようになっており漢字フォントだけでなく、従来の1バイトASCIIフォントも扱えるようにするためpixfont ptr を設け、漢字フォントと従来のASCIIフォントの切換を可能としている。

さらに、フォントイメージは、半角文字用／全角文字用の2つのpixrect に格納して、pixrect を管理するための中間管理データの大巾削減を行った。

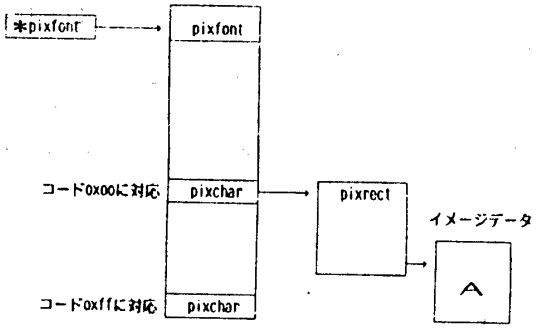

図3　オリジナルのフォント制御データ構造

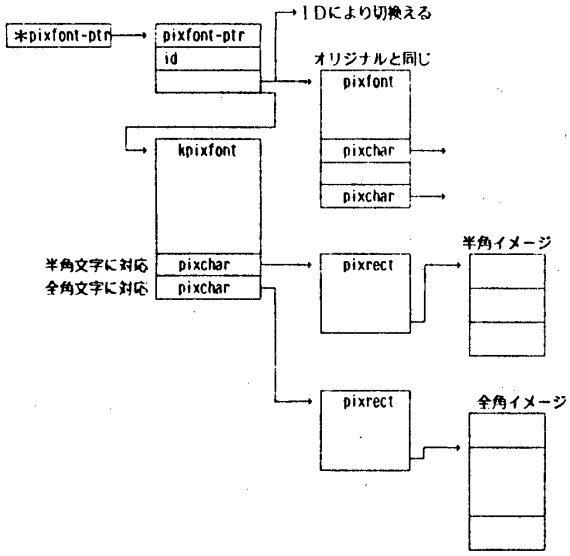

図4　漢字のフォント制御データ構造

3.3 フォントRAM

漢字のシメージデータは、16×16ドットのフォントで約250KB 必要である。オリジナルでは、フォントオープン時にプロセス毎の仮想空間に領域を確保し、フォントイメージをローディングするため、これを漢字フォントにそのまま適用すると時間、メモリ共に多く必要となる。

そこでメモリの一部をフォントRAMとして専用化させ、ブート時にフォントイメージをローディングする方式を採用した。フォントRAMはフォントRAMドライバを通して、メモリマップ機構によりアクセスすることができる。

この方式の利点を述べる。

①フォントオープン時間の短縮：　オープン時にフォントをローディングする必要がないため、フォントRAMのイメージアドレスをフォント制御データ構造（pixrect)へ設定するだけでよい。

②メモリの共有化：　複数プロセスでフォントRAMは共有されるため、フォントイメージでは、実メモリにただ1個のみあればよい。

③マルチフォント、フォントレパートリの拡張性：漢字フォントファイルの形式に従ったファイルであれば、どのフォントでも対象することができる。ユーザ自身が作成したフォントも、フォントRAMに置くことができる。

④汎用性：　AS3050/3052 では漢字RAMを実装できないハードウエア上の制約がある。フォントRAM方式ならば全機種に対応できる。

なお、フォントのローディングは、任意の時点で実行できるように、コマンドとしても用意されている。

3.4 フォントオープン処理

フォントオープン処理は以下の順序で行う。

①指定されたフォントファイルのフォントイメージがフォントRAMにあるか調べる。

②フォントイメージがあれば④へ。

③フォントイメージがなければ、フォントファイルから、そのプロセスの空間にフォントイメージをローディングする。

④フォント制御データ構造を作成する。

3.5 その他の機能

(1)フォントオープンレベルの制御

フォントオープン時に、ローディングするフォント範囲を3種類選択できる。

①ＪＩＳ第1水準のみ。

②ＪＩＳ第1水準、第2に水準。

③ＪＩＳ第1水準、第2に水準および外字。

(2)ピッチ制御

漢字フォントは、ピッチデータを持っていないため、フォントイメージのローディング時に、ピッチデータを付加するようにしている。

(3)マルチフォント

フォントRAMには、複数フォント置くことができる。フォントRAMサイズは、システムパラメータで与えられるので、簡単に再構成することができる。

4．まとめ

本方式により、上位ルーチンのほとんどは、フォント制御データ構造へのポインタのみ意識するように設計されているため、少ない修正量で日本語表示化することができた。

表示の性能は、オリジナルの漢字を表示しない場合とほとんど変らず、オープン時間もほぼ同じである。　今後の課題としては、フォントRAMの仮想記憶化、キャッシング機構の導入、等を考えている。