マイクロコンピュータを用いた出力検索システム：FAMOUS 1†

野上睦夫†† 重松保弘††

本稿では，マイクロコンピュータを端末（μT）とする出力検索システムについて述べる．本システムでは，ホスト計算機（F 230-45 S）のジョブ出力ファイルは，μT の両面フロッピディスク装置に格納され，更に，カラーディスプレイ装置に表示される．利用者はコマンドを用いて必要な部分を検索し，ハードコピーをとることができる．本システムは，端末の高度なインテリジェント化を目指したものであり，(1)出力検索がホスト計算機と独立に実行できる，(2)データの出力量を気にしなくてよいので，ホスト計算機のソフトウェアリソース（トレーサ，デバッガやメモリダンプ機能など）を有効に利用できる，などの特徴をもつ．

1. まえがき

バッチモニタをもつ電子計算機システムを教育用に利用する場合，ラインプリンタ(LP)に出力される大量のデータのうち一部分しか必要としないことが多い．例えば，プログラムのデバッグ中は各種のエラーメッセージやエラーの原因を確かめるだけで十分であり，計算結果のすべてを出力する必要はない．こうした無駄をなくすために，筆者らはマイクロコンピュータを用いた出力検索システム FAMOUS 1 (FACOM 230-45 S And Microcomputer Output retrieval System version 1) を開発した．FAMOUS 1 の特徴は，(1)マイクロコンピュータをホスト計算機 (FACOM 230-45 S) に外付けしてあるので検索がホスト計算機と独立に実行できる．(2)データの出力量を気にしなくてよいので，ホスト計算機のソフトウェアリソース（トレーサ，デバッガやメモリダンプ機能など）を有効に利用できる．(3)将来，リモートジョブエントリシステムへの発展が可能である．(4)計算機運営に必要な LP 用紙等の消耗品の節約に役立つ，などである．また，本システムのソフトウェアを開発する際，パスカル風言語でシステムを記述し，それをハンドコンパイルした後，得られたアセンブリコードを最適化するという手法を用いた．

以下，マイクロコンピュータを用いて開発した出力検索用ターミナルを μT（マイクロターミナル）と呼ぶ．

† An Output Retrieval System Using a Microcomputer: FAMOUS 1 by MUTSUO NOGAMI and YASUHIRO SHIGEMATSU (Department of Computer Science, Kyushu Institute of Technology).

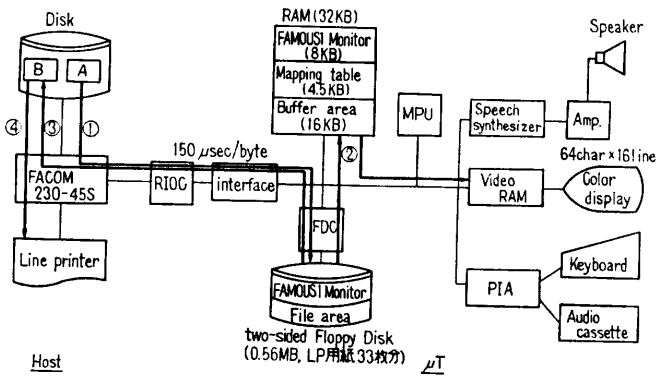
†† 九州工業大学情報工学科

2. FAMOUS 1 のシステム構成と検索手順

一般に，μT を用いて効率の良い出力検索システムを実現するには，μT 側に大容量のランダムアクセスファイルや，表示文字数の多いキャラクタディスプレイ装置等のハードウェア資源を必要とする．また，ホスト計算機との高速のデータ転送路も必要となる．本システムではこれらの要求を満たすために，**図1**に示すようなハードウェア構成を採用した．

本システムでは，ホスト計算機におけるジョブ出力は，次の4段階を経て LP に出力される（図1参照）．(1)通常バッチジョブで出力されるジョブの出力用ファイル（図1のファイルA）を RIOC (Remote I/O Controller) を経由して μT へ転送し，フロッピディスクに格納する．(2)転送されたファイルをディスプレイ装置に表示し，μT は検索可能の音声メッセージを出す．利用者は**表1**に示すコマンドを用いて出力ファイルを検索し，必要なら LP に印刷すべき行を指定する．(3)Tコマンドを入力すると検索は終了し，フロッピディスク中のファイルのうち印刷を指定された行だけをホスト計算機へ転送し，ディスク装置へ格納する（図1のファイルB，BはAの部分ファイル）．(4)ディスク上のファイルBを LP に印刷する．

データ転送が大量になっても，あえて，μT として独立させた理由は，ホスト計算機の出力ファイルが順編成ファイルであり，検索のための乱処理には適していないからである．また，ホスト計算機上に検索処理プログラムを常駐させ，検索を行うのは，ホスト計算機のメモリ使用効率を大きく低下させる．なお，ファ



RIOC: Remote Input Output Controller
PIA: Peripheral Interface Adapter

図 1 FAMOUS 1 のハードウェア構成 (→ はデータの流れ)
Fig. 1 The hardware construction of FAMOUS 1 system (→: data flow path).

表 1 FAMOUS 1 のコマンド
Table 1 Commands of FAMOUS 1.

コマンド種別	コマンド	機能
モード制御	S	表示画面をスクロールモード (行単位の移動) にする
	P	表示画面をページモード(画面単位の移動)にする
	64C	表示画面を 64 文字モードにする
	32C	表示画面を 32 文字モードにする
画面制御	n ↑	n ライン (n 画面) 上方向移動 ($1 \leq n \leq 99$)
	n ↓	n ライン (n 画面) 下方向移動 ($1 \leq n \leq 99$)
	n ←	n カラム左方向移動 ($1 \leq n \leq 99$)
	n →	n カラム右方向移動 ($1 \leq n \leq 99$)
	CR	最左端の画面へ戻る
	GO HOME	第一画面へ戻る
印刷指定	L	現在表示されている画面全体を印刷指定
	n L	現在表示されている画面の n 行目を印刷指定 ($1 \leq n \leq 14$)
	m , n L	現在表示されている画面の m 行目から n 行目までを印刷指定 ($1 \leq m \leq n \leq 14$)
	0 , 0 L	リストファイル全体を印刷指定
検索終了	T	出力検索を終了する

イルの転送に要する時間は比較的短い(最大転送時間* 約2分30秒，学生の演習用ジョブでは通常数秒).

3. 検索の効率化・高速化

出力検索システムは利用者の立場からはできる限り短い応答時間であることが望ましい．しかし，マイクロコンピュータやフロッピディスクの処理速度は，一般に，ホスト計算機と比較するとかなり遅い．そのため，本システムでは，以下に述べる方法で検索の効率化・高速化を図っている．

* ディスケットのファイル領域を満たすまでの時間.

3.1 主記憶のバッファ構造とその管理

μT の主記憶は現在 32 KB あり，このうち 16 KB をバッファ領域としている．ここには，64 行 (256 バイト／行) のデータが格納できる．バッファ領域の欄構成 (**図 2** 参照) における file_end 欄は，ジョブ出力ファイルの最後のレコードを示すために用いられる．list_record 欄は通常 LP に印刷される 1 行分のデータであり，color 欄はそれをディスプレイ装置に表示する際の色コードを保持している．logical_adress 欄には，そのレコードが格納されるフロッピディスクの論理アドレス (後述) を格納する．

このバッファ領域は環状になっており，ライン (出力リストの行単位) の境界 (最下端の行) を示すポインタとして bnd (boundary) が用意されている．また，ディスプレイ装置への表示用ポインタとして3つのポインタが用意されており，top (display top) ポインタが指すラインから btm (display bottom) ポインタが指すラインのうち，column ポインタが指すカラムから 61 文字がディスプレイ装置に表示される．

3.2 マッピングテーブル

マッピングテーブルは，出力リストのラインとそれ

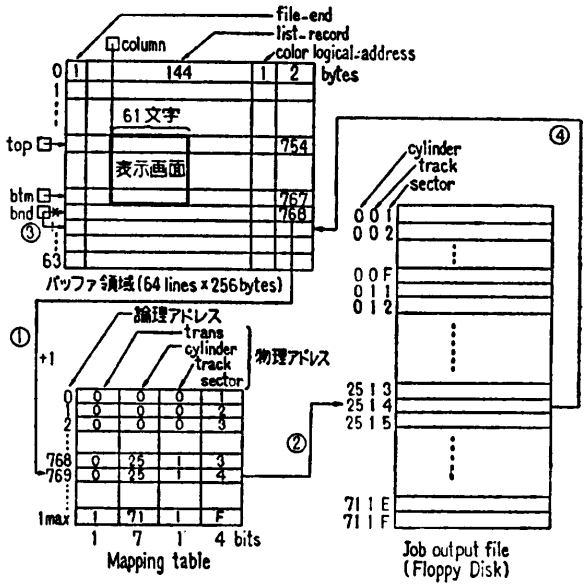


図 2 主記憶のバッファ領域，マッピングテーブルおよびフロッピディスクの各構成とラインインの過程
Fig. 2 The process of Line-in and the specification of the buffer area in the main memory, the mapping table and the file of the floppy disk.

が格納されているフロッピディスクのシリンダ，トラック，セクタ番号（これらをまとめて物理アドレスと呼ぶ）との1対1の対応関係を保持するために用いる．これは，ホスト計算機から転送されてきたジョブ出力ファイルをフロッピディスクに格納するときに作成される．このとき各ラインに対しシーケンス番号（これを論理アドレスと呼ぶ）を付け，これを各ラインの論理アドレス欄（図2のlogical_address欄）に格納する．マッピングテーブルの欄構成（図2）における cylinder, track, sector の各欄はラインの格納されているフロッピディスクの物理アドレスを保持している．trans 欄の値は，このラインが印刷指定されたときに“1”となる．

3.3 ラインインの手順

ホスト計算機のジョブ出力ファイルをすべて μT の主記憶上に格納できないとき，必要なラインをフロッピディスクから主記憶のバッファ領域に読込むことが必要になる．この動作をラインインと呼ぶ．

スクロールモードでは，表1で示したコマンドのうち↑コマンドを入力すると，top と btm がインクリメント（1を加えて 64 でモードをとる）される．しかし図2の例では，↑コマンドを2度入力して top と btm をインクリメントすると，btm の指すラインは利用者の期待するラインではなくなり，ラインインが必要となる（この状態をラインフォルトと呼ぶ）．すなわち，↑コマンドを入力したときに btm＝bnd ならば，top と btm をインクリメントする前にラインインが必要になる．最も基本的なラインインの手順は**図3**ⓐのようになる．ここで disk_read (phy) は，phy のもつ物理アドレスから1セクタ分のデータを読込んでくる関数である．すなわち，①bnd の指すラインの論理アドレスに1を加え，新たな論理アドレスとする（図2の例では 769 となる）．②マッピングテーブルを参照してこの論理アドレスに対応する物理アドレスを取出す（第25シリンダの第1トラックの第4セクタ）．③bnd をインクリメントする．④フロッピディスクから読込まれたラインをバッファ領域の bnd が指すライン領域へ格納する．

しかし，この手順では，一度ラインフォルトが起こると，同方向のコマンドを入力するたびにラインフォルトが起こる．そこで，ラインフォルトが起こったときに2画面分，すなわち 28 ライン分の先読みを行い（図3ⓑ）ラインフォルトの回数を減らしている．

フロッピディスクのトラック中のセクタ番号は，フロッピディスクの回転方向に添って付けているので，画面の上方向の移動時にラインフォルトが発生した場合は，ラインの連続読出しが高速になり，検索は速く

```
var
        d: disk_logical_address;
      phy: disk_physical_address;
        i: integer;
   d1,d2: buffer_address:
   作業用変数

d:=list_buffer [bnd]. disk_logical_address+1;
phy:=mapping_table [d]. address;
bnd:=(bnd+1) mod 64;
list_buffer [bnd]:=disk_read (phy);
   ⓐ最も基本的なラインインの手順

i:=1;
while i<28 and not end_of_file do
   begin
      d:=list_buffer [bnd]. disk_logical_address+1;
      phy:=mapping_table [d]. address;
      bnd:=(bnd+1) mod 64;
      list_buffer [bnd]:=disk_read (phy);
      i:=i+1
   end
   ⓑ28 ライン分先読みを行うためのラインインの手順

d2:=bnd;
d:=list_buffer [top]. disk_logical_address-28;
bnd:=(bnd-28+64) mod 64;
while d<0 do
   begin
      d:=d+1;
      bnd:=(bnd+1) mod 64
   end;
d1:=(bnd+1) mod 64;
while d1<>d2 do
   begin
      phy:=mapping_table [d]. address;
      list_buffer [d]:=disk_read (phy);
      d1:=(d1+1) mod 64;
      d:=d+1
   end
   ⓒ画面を下方向（バッファ領域では上方向）に移動させる時のラインインの基本的な手順

if page_mode and ¯operand>3 then
   begin
      d:=list_buffer [top]. disk_logical_address+operand*14;
      bnd:=0;
      repeat
         phy:=mapping_table [d]. address;
         list_buffer [bnd]:=disk_read (phy);
         bnd:=(bnd+1) mod 64;
         d:=d+1;
      until end_of_file or bnd:=0;
      bnd:=63;  top:=0;  btm:=13
   end
   ⓓ3画面以上離れた画面を上方向に検索する時のラインインの基本的な手順
```

図3　ラインインの手順
Fig. 3　The procedures of Line-in.

なる．しかし，画面の下方向の移動時にラインフォルトが発生した場合，フロッピディスクの回転方向と逆の順番でラインを読出すことになり，回転待ちのため検索速度は極めて遅くなる．そこで，このときは，2画面分前のバッファ領域の位置と目的ラインの論理アドレスを発見し，フロッピディスクの回転方向に添って順次ラインを読出す方法を採用し(図3ⓒ)，検索時間を短縮している．

また，検索画面が現在の画面と大きく離れている場合，これらの方法では目的画面に到達するまでのラインをすべてフロッピディスクから読出すことになり，検索時間が長くなる．このため，目的画面の先頭ラインの論理アドレスを求めて，マッピングテーブルを参照しながらフロッピディスクからラインを読出し，バッファ領域を満たしてゆく，という方法を採用（図3ⓓ）した．

これらの方法を用いることによって，どの画面でも短時間（最大約2秒）で検索することが可能になった．

4. むすび

マイクロコンピュータをホスト計算機に結合することによって，十分実用に耐えうる効率の良い出力検索システムを安価に作成することができた．今後，より使い易いシステムとなるように操作方法，処理機能に改良を加える予定である．更に，リモートジョブエントリシステムへの拡張についても現在検討中である．

最後に，有益な御助言をいただいた本学情報工学科の磯泰行教授，安在弘幸助教授，および，本システムの作成に協力していただいた卒論生諸氏に感謝します．

参考文献

1) 二村：出力検索システムの利用について，九州大学大型計算機センター広報，Vol. 11，No. 1，pp. 17-28 (1978)．
2) 野上，重松：マイクロコンピュータを用いた出力検索システム：FAMOUS 1，情報処理学会・マイクロコンピュータ研究会資料 9-2 (1979)．

(昭和54年7月24日受付)
(昭和54年10月25日採録)