センターでの日本語処理の一例（利用者の送付先名簿作成）

共同利用掛* 長谷部　功

送付先名簿
（速報）

昭和５８年　５月３０日

大阪市立大学連絡所　　御中

下記のとうり配付方よろしくお願いいたします。

記

学　部	学　科	氏　名	身　分	部　数
経済学部	経済学科	○○ ○○	教授	1
文学部	人間関係学科	○○ ○○	教授	1
理学部	物理学科	○○ ○○	助教授	1
	化学科	○○ ○○	助教授	1
		○○ ○	院生（後）	1
	地学科	○○○ ○	教授	1
工学部		○○ ○○	教授	1
	建築学科	○○ ○○	助手	1
	建築工学科	○ ○○	講師	1
原子力基礎研究所		○○ ○○	助手	1
計算センター		連絡所		1

合計　１１部

大阪大学大型計算機センター　共同利用掛

558
大阪市住吉区杉本３−３−１３８

大阪市立大学
計算センター

大阪市立大学連絡所

御中

564−00
吹田市山手町３−３−３５

関西大学
情報処理センター

関西大学連絡所

御中

＊現在工学部教務掛

実際は、各連絡所へ送られるので、直接、利用者の目には触れていないかもしれませんが、今年の３月から、センター・ニュース、速報等の送付先名簿及び、住所シールが漢字になっています（先頭の図がそれです)。センター業務のいわゆる日本語化の一つであり、カタカナに比べれば、当然、読みやすく、郵便上のトラブルや誤配付の減少に役立つものと考えています。

また、経理責任者宛の利用通知書や納入告知書も、近いうちに、漢字になる予定です。

利用者の中にも、なんらかの形で、日本語情報処理システム（JIPS)の利用を考えている人が居ると思いますが、残念な事に、JIPS は容易に理解出来るとは言いがたく、そのうえ、理解した後も、ユーザ・サイドに、かなりの労力を強いる事になると思われます。

そこで、そういった事に対する、ささやかなアドバイスになればと思い、一例として、送付先名簿の日本語化の過程を紹介することにしました。

なお我々はプログラム言語としてCOBOL 74を使用したので、利用者側に、直接あてはめる事は出来ないかもしれません。しかし、JIPS の機能そのものは、プログラム言語に左右されないので、FORTRAN77 を使用する場合にも、概念的には、なんら変ることがありません。

そういう意味において、本稿が、利用者の JIPS 利用に役立つことを期待しています。

1）フローチャート

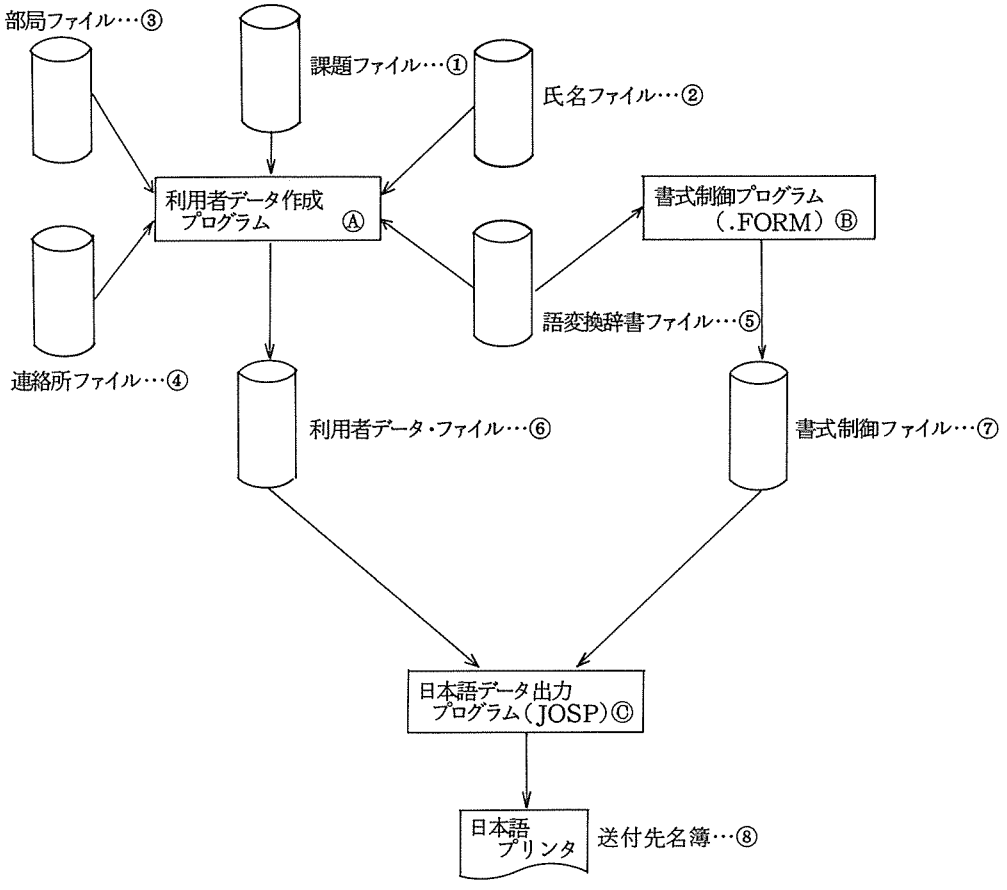


図１

図１が送付先名簿作成の基本的な流れである。

まず、利用者データ作成プログラムⒶが、各種の利用者情報ファイルを読み込んで、送付先名簿用の利用者データ・ファイル⑥を作成する。

それと平行して、名簿の帳票イメージ等を、書式制御プログラムによって作成し、その結果をファイルに出力する。つまり、書式制御ファイル⑦の作成である。

最後は、日本語データ出力プログラムⒸが、⑥、⑦のファイルを読み込み、両者を重ね合わせ（オーバーレイと言う）、日本語プリンタに、送付先名簿⑧として出力する訳だ。

それでは、このフローチャートの細部を順に説明して行く事にしよう。

2）利用者情報ファイル

課題ファイル①には、さまざまな利用者情報（課題番号、氏名、パスワード、所属、連絡先、身分、予算額、その他）が登録されていて、センターはこれを基に利用者管理を行なっている。

これらの情報の中で、送付先名簿に必要な情報は氏名、身分、連絡先である。この３つを漢字に置き換えてやれば、おしまいなのである。

まず、氏名。利用者情報の中で、各利用者ごとに絶対ユニークなものは課題番号下４桁。だから、この４桁をキーにした漢字の氏名ファイル②を作成する。

連絡先は、大学、学部、学科がコードで登録されているので、それぞれのコードをキーにした部局ファイル③を作成する。ただし、送付先名簿の宛名は、大学名ではなく、連絡所名なので、連絡所ファイル④を作成する必要もある。課題番号の上４桁が、その利用者の属する大学の連絡所番号なので、その連絡所番号をキーにすればいい訳である。

身分も、連絡先と同じようにコードで登録されているが、種類が２０程度なので、身分ファイルを用意するのは不能率である。そこで、後で述べる語変換辞書ファイル⑤（単語、熟語を登録して置く）の中に含める事にした。

又、送付先名簿には関係ないが、住所シールのための情報も必要である。しかし、住所というものは、大学によって決まるものである（ある学部が離れたところにある場合には、大学、学部によって決まる）から、大学、学部コードをキーにした住所ファイルを作成すればいい訳である。

ここで各ファイルのフォーマットを図解してみよう。（図２。一人の利用者だけに注目している。課題ファイルにしても、送付先名簿に関係のある情報だけに限っているので、本当は、もっと多量の項目が存在しているのである）

課題ファイル①

課題番号		氏　名	大学コード	学部コード	学科コード	身分コード	配付部数
601233	0001	イチダイ△タロウ	126002	040	1101	01	01

氏名ファイル②

課題番号下4桁	氏　名
0001	市大△大郎

部局ファイル③

大学コード	学部コード	学科コード	部局名
126002	△△△	△△△△	大阪市立大学
△△△△△△	040	△△△△	経済学部
△△△△△△	△△△	1101	経済学科

連絡所ファイル④

連絡所番号	大学コード	学部コード	学科コード	連絡所名
6012	126002	753	0000	大阪市立大学連絡所

連絡所自体の部局コード（連絡所宛、住所シール作成時に必要）

住所ファイル

大学コード	学部コード	住　所
126002	△△△	〒558-00大阪市住吉区杉本3-3-138

（△はブランクを表わす）

図２

氏名、部局、連絡所及び住所ファイルは、それぞれのキーをもとに、ランダム・アクセス出来る事が望ましい。だから、氏名ファイルについては、課題番号下4桁を、セクション・ナンバーにみたてたUFAS相対編成ファイルにすることにした。

しかし、部局、連絡所及び住所ファイルはそうはいかない。なぜなら、キーそのものに、かなりの空き番があるため、ダミーレコードの量が多く、効率がいちじるしく悪くなるからである。

そこで、この3つはUFAS索引編成ファイルにすることにした。索引編成ファイルは、データ部分のファイルと別に、インデックス部分のファイルを持ち（つまり、論理的には一つのファイルであるが、物理的には、二つのファイルに分かれている）、このインデックス部分を使用することによって、順編成ファイルのようにレコードが並んでいるデータ・ファイルをランダムにアクセスすることが出来るのである。

もちろん、最初にFMSかACCESSサブシステムでファイルを作成する時には、UFAS索引編成ファイルともに、ランダム・ファイルとして作成する訳で、問題は、プログラムの方で、そのファイルを、どう扱うかの違いである。

残る問題は、漢字の入力である。実は、これが一番やっかいな問題なのである。

我々が、この漢字化を行った時点では、漢字入力の方法が二つしかなかった。一つは16進によるコード入力。もう一つが、和文タイプのボードのように漢字の並んだタブレットからのペンタッチ入力。どちらも、かなりの労力を必要とした。日本語化作業の半分を占めたと言ってもいい。

利用者数が2000人、一人当たりの氏名の文字数が4文字として、氏名ファイルに関しては、4×2000＝8000字。連絡所の数が250、一件当たりの、連絡所名の文字数が15文字として、連絡所ファイルは15×250＝3750字。住所の種類は500種類程度、1件当たりの住所の文字数が15文字として、住所ファイルに関しては、15×500＝7500字。計19250字。約20000字である。（部局ファイルに関しては、東京大学が各大学に、磁気テープで配付した漢字のデータをコード変換することによって、直接入力をさけることが出来た）

現時点においては、住所変換ユーティリティ「ADDKS」や姓名変換ユーティリティ「NACKS」が用意されている。これらは、カタカナ・データをキーにして、対応する漢字を検索するシステムであるが、使用方が非常に難解で、しかも、その割には、漢字の的中率が低い。

だから、利用者にとって、一番良いと思われる方法は、端末によるカナ・ローマ字／漢字変換だろう。センターの日本語端末には、キー・ボード操作のみで、カタカナやローマ字で入力したデータを、その場で漢字に変換する機能が付いている。しかも、この方法なら、たとえ一回目の検索による漢字が違っていても、簡単なキー・ボード操作で、見つかるまで、何度も検索を行う事が出来るのである。

3）語変換辞書ファイル

前にも言ったように、利用者の身分については、身分ファイルを用意するのが極めて不能率なので、プログラムの中で、日本語直定数として指定することにした。

日本語直定数には、１６進コードで指定するものや、直接、日本語で指定するものなど、数種類あるが、その中の一つに、語変換辞書ファイルを用いた索引直定数（COBOL74での呼び名）というのがある。

図３が語変換辞書ファイルの作成ディレクティブである。

```
0010##S,U
0020¥       IDENT   USERID,MAIL,R,A,,JPR
0030¥       PROGRAM DICMTN
0040¥       LIMITS  ,80K
0050¥       PRMFL   N*,W,R,USERID/JISHO-DATA
0060¥       PRMFL   *N,W,R,USERID/JISHO-INDEX
0070¥       SYSOUT  PJ
0080¥       SYSOUT  PP
0090¥       DATA    I*,,COPY
0100¥       JIS
0110//LOAD EDIT=YES,FIELDSZ=(16,2,32,5),
0120        FIELDSEP=("=",NX"1A70",NX"1A71",".",SORT=YES
0130ｷｮｳｼｭ=1教授.
0140ｷｮｳｼｭ=2享受.
0150ｼﾞｮｷｮｳｼｭ=0助教授.
0160ｺｳｼ=0講師.
0170ｼﾞｮｼｭ=0助手.
0180ｷｮｳﾑ=0教務.
0190ｷﾞｼﾞｭﾂ=0技術.
0200ｼｮｸｲﾝ=0職員.
0210ｲﾝｾｲ=0院生.
0220ｶﾞｸｾｲ=0学生.
0230ｹﾝｷｭｳｾｲ=0研究生.
0240ｹｲ=0系.
0250ｺｳｶﾞｸ=0工学.
0260ﾘｶﾞｸ=0理学.
0270ﾘｺｳ=0理工.
0280ｲﾔｸ=0医薬.
0290ﾌﾞﾝｶ=0文科.
0300ｷｮｳｮｳﾌﾞ=0教養部.
0310//LIST ALL,FIELDLENG=(16,2,32,5)
0320¥       ENX
0330¥       ENDCOPY
0340¥       ENDJOB
```

図３

ライン番号　　０１３０を例にとって、分かりやすく図解する。（図４）

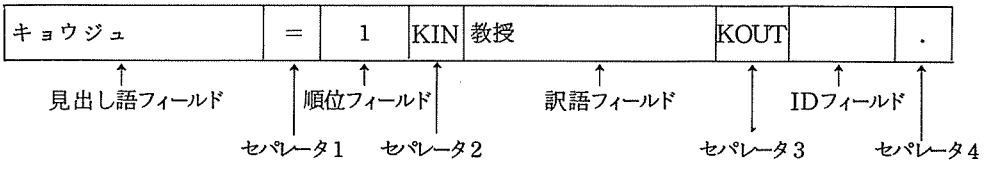


図４

見出し語フィールドは、プログラムのコーディング上に記述するキーのようなもので、ここではカタカナであるが、アルファベットでも数字でも良い。

順位フィールドは同一見出し語に対する訳語が複数ある場合、それぞれを区別するために指定するもので、たとえば、ライン番号0140は、見出し語が同じ「キョウジュ」であるため、順位フィールドに2を指定している。訳語が一つであれば、他の例のように、順位フィールドは「0」で良い。

訳語フィールドには、普通、目的とする漢字を指定する。ここでは、直接、漢字文字を指定しているが、そのためには、漢字を入力出来る端末が必要である。普通の端末で行う場合には、16進コードで指定する方法もある。

IDフィールドは、辞書参照ルーチン（ここでは触れない）を利用して、ユーザ・プログラムから、自由に書き込む事が出来るエリアである。使い方によっては、有意義なものになるかもしれないが、語変換自体には意味のない、いわば、遊びエリアである。

これら4つのフィールドの終わりを示すのが、各セパレータであり、等記号やコンマ、あるいはピリオドなどを使用するのが適当だろう。しかし、この例では、セパレータ2とセパレータ3に一風変わったものを使用している。「KIN」、「KOUT」というのは、文字列ではなく、シフト・コードの名称である。

ここで、少し、「KIN」、「KOUT」について触れておこう。

従来の英数字やカタカナ等の標準文字は1バイトコード、つまり1キャラクタ＝1バイト（JISの場合9ビット）で表わされるのに対して、日本語文字（漢字やひらがな等）は2バイトコード、つまり1キャラクタ＝2バイト（18ビット）で表わされる。JIPSでは、この二つのデータが混在するので、それを区別するため、なんらかの方法が必要である。

そこで、シフト・コードの登場となる訳だ。「KIN」は2バイトコードの始まりを表わし、「KOUT」は2バイトコードの終わりを表わす。つまり、システム側はデータを見る時、何もなければ1バイトコードとして見ていて、「KIN」が現われたら、それ以後を2バイトコードとして見るのである。

もちろん、「KIN」、「KOUT」も2バイトコードであり、16進コードで表わすと、それぞれ〝1A70〟、〝1A71〟である。

再びセパレータの説明にもどろう。各セパレータに何を使用するのかは、利用者の自由であるから、図3のライン番号0120において、セパレータを指定している。「NX〝1A70〟」は「KIN」、「NX〝1A71〟」は「KOUT」を表わしている。（NX〝～〝は16進コードで表わす日本語直定数である）

なぜ、コンマ等ではなく、「KIN」、「KOUT」を使用しているのかと言うと、日本語端末か

らオンラインで日本語を入力する時、前もって、JIPSコマンドを入力しておけば、日本語データの前後に「KIN」、「KOUT」が自動的に付加される。だから、それをセパレータとしておけば、改めて、セパレータを入力する必要がないし、辞書の内容を日本語プリンタへリスト出力した時、シフト・コードとして働き、日本語が正しく印字される。（日本語の前後に、「KIN」、「KOUT」が無ければ、システムは、それを2バイトコードとしては見てくれないので、正しく印字されない）

又、図3を見れば分かるように、「KIN」、「KOUT」は、存在していても、実際には印字されない。

このようにして、作成された辞書は、語変換辞書ファイル⑤（図1参照）に納められる。語変換辞書は、前に述べたUFAS索引編成ファイルと同じようにランダム・ファイルであり、しかも、データ・ファイルとインデックス・ファイルの2つのファイルから成っている。図3のライン番号0050および0060がその定義文である。（「JISHO－DATA」はデータ・ファイル、「JISHO－INDEX」はインデックス・ファイル）

次に、この語変換辞書ファイルを、ユーザ・プログラム側で使用する方法であるが、その例を図5に示した。

ライン番号0050の$CBL74文において、必ず指定しなければならないオプションは、「JIPS」（日本語処理機能を使用する）と「DICT,（D1, D2……）」（日本語辞書ファイルを使用する）である。D1，D2は辞書ファイルのデータ・ファイルのファイルコードであり、インデックス・ファイルのファイルコードは、それぞれ、1D、2Dという具合にする。ライン番号0080と0090が、一組のファイル定義文である。辞書ファイルは、コンパイル時に使用されるので、必ず、コンパイルのアクティビティ中に定義文がなければならない。

ライン番号4140の中の「NI″%キョウジュ＃1″」が、辞書ファイルを使用した索引直定数の形式である。「キョウジュ」が見出し語であり、「＃1」は順位フィールドに「1」が入っている事を示している。これで訳語「教授」を引っぱって来る訳だ。

この直定数の転記先である変数「MIBUN」はライン番号1880を見れば分かるように、N項目（日本語項目）でなければならない。N項目は2バイトコード用の項目だから、当然、標準の英数字項目の2倍の大きさを持っている。したがって、N(5)＝X(10)である。

そういう意味では、仮に漢字を標準の16進文字直定数（たとえば、「教授」なら「″″3635 3C75″″」）で表わしてやれば、漢字の文字数の2倍の大きさを持ったX項目に格納する事も可能な訳である。

```
0010###M
0020¥       IDENT   USERID,MAIL,R
0030¥       USERID  USERID¥PASSWORD
0040¥       LIMIT   ,,,10000
0050¥       CBL74   LSTIN,JIPS,DICT,(D1)
0080¥       PRMFL   D1,R,R,USERID/JISHO-DATA
0090¥       PRMFL   1D,R,R,USERID/JISHO-INDEX
0100        IDENTIFICATION DIVISION.
0110        PROGRAM-ID. SOUFU.
0120        AUTHOR. CENTER.

                                (
                                )

1780        DATA DIVISION.
1790        01  USER-LIST.
1800            02  RENRAKU-CODE     PIC 9(4).
1810            02  DAIGAKU-CODE     PIC 9(6).
1820            02  GAKUBU-CODE      PIC 9(3).
1830            02  GAKKA-CODE       PIC 9(4).
1840            02  MIBUN-CODE       PIC 9(2).
1850            02  GAKUBU           PIC N(30).
1860            02  GAKKA            PIC N(30).
1870            02  NAME             PIC N(15).
1880            02  MIBUN            PIC N(5).
1890            02  BUSU             PIC 9(2).
1900            02  SMC-NAME         PIC N(10).
1910            02  CHECK-KYODO      PIC X(1).

                                (
                                )

4130        SET-MIBUN.
4140            IF MIBUN-CODE EQUAL 1 MOVE NI"%ｷﾖｳｼｭ#1" TO MIBUN
4150                ELSE NEXT SENTENCE.
4160            IF MIBUN-CODE EQUAL 2 MOVE NI"%ｼｭﾖｷﾖｳｼｭ" TO MIBUN
4170                ELSE NEXT SENTENCE.
4180            IF MIBUN-CODE EQUAL 3 MOVE NI"%ｺｳｼ" TO MIBUN
4190                ELSE NEXT SENTENCE.
4200            IF MIBUN-CODE EQUAL 4 MOVE NI"%ｼｭﾖｼｭ" TO MIBUN
4210                ELSE NEXT SENTENCE.
4220        EXIT-BUKYOKU.
4230            EXIT.
4240¥       EXECUTE
4250¥       LIMIT   25,200K,-6K,10000


4300¥       ENDJOB
```

図5

4) 利用者データ・ファイル

今まで述べて来たようにして、作成した利用者情報ファイルを利用者データ作成プログラムⒶ（図1参照）が読み込み、利用者データを作り上げる。利用者データ・ファイル⑥（図1参照）のフォーマットは、図6のとおりであり、図7は、一大学（例として、大阪市立大学）のデータの内容を示した。

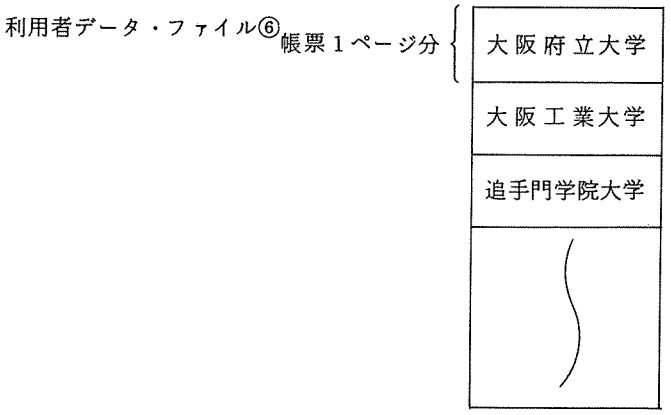


図６

大阪市立大学のデータ

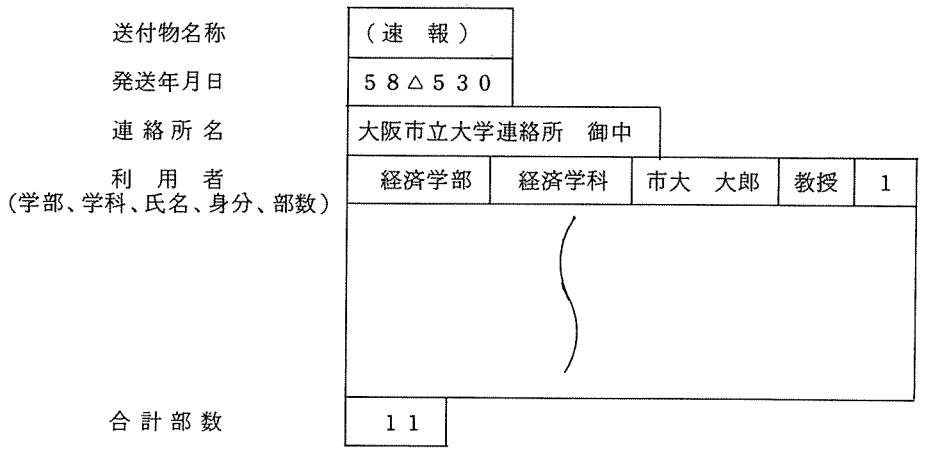


図７

一大学のデータが帳票１ページ分のデータである。しかし、それは、あくまで 6）で述べる書式制御プログラム側で指定するのであって、利用者データ・ファイル自体は、図７で示したようなデータが、図６のように、ただ単純に並んでいるだけである。

これを、日本語データ出力プログラム◎（図１参照）が読み込んで、帳票とオーバーレイさせて日本語プリンタに出力するのである。

我々は、この日本語データをCOBOL 74のプログラムで作成した訳であるが、もちろん、FORTRAN77プログラムで作成したデータや、あるいは、日本語エディタ（NEDIT）で、端末から直接入力したデータでも、同じように、日本語データ出力プログラム◎を使って、帳票とオーバーレイさせる事が出来る。

5) 日本語プリンタ

書式制御プログラムによる帳票作成に入る前に、日本語プリンタの印字処理の概念について説明しておこう。

日本語プリンタへの印字位置は行とカラムによって定められる。（図８）

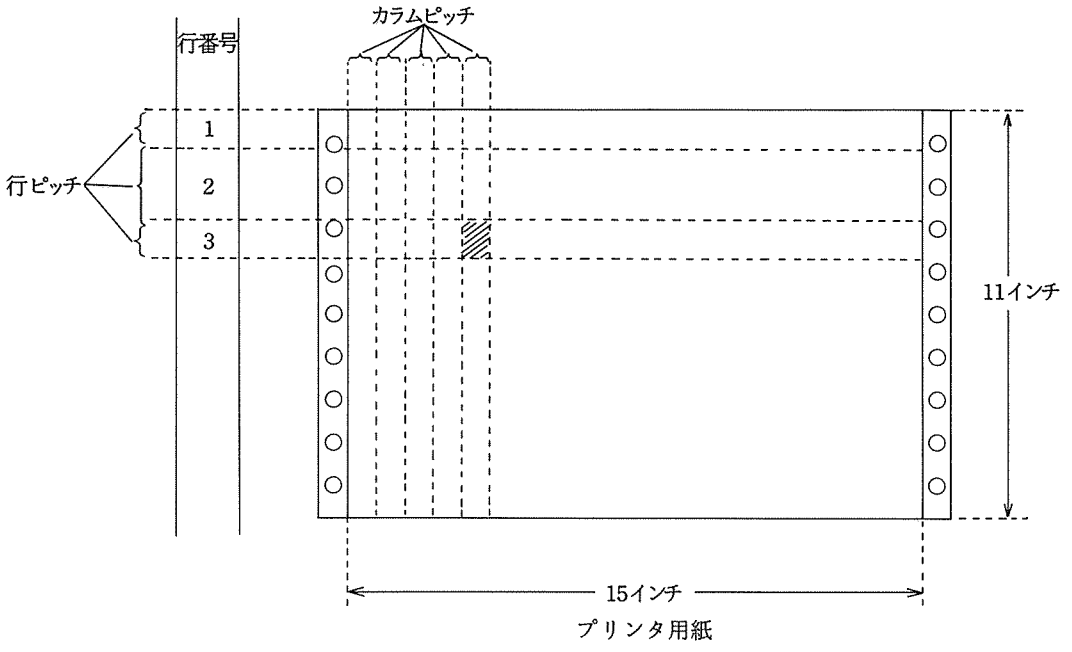


図８

行ピッチは、通俗的な言い方をすれば行の太さであり、これは書式制御プログラムで、任意に、しかも、行ごとに指定出来る（図８においても、行ごとに太さが違う）。指定出来る行ピッチの種類は｛12, 8, 6, 4, 3 LPI｝の５種類である。LPIは、行ピッチの単位であり、１インチを何行に分けるかを表わしている。だから数値の小さい方が、実は行が太い事になる。また、標準のプリンタ用紙１ページの縦幅は１１インチであるから、たとえば、全ての行を、行ピッチ８LPIで指定すれば、一ページが88行に分けられる訳である。

カラムピッチも、書式制御プログラムで指定出来るが、こちらの方はすべての行に対して等間隔な幅としてしか、指定出来ない。カラムピッチの種類は、｛20, 16, 12, 10, 8, 6, 5, 4, 3 CPI｝の９種類である。CPIはカラムピッチの単位であり、同じように、１インチを何カラムに分けるかを表わしている。たとえばカラムピッチ10CPIを指定すれば、プリンタ用紙１ページが150カラムに分けられるのである。

こうして、縦と横に、碁盤の目のように区切られた用紙に文字が印字される訳である。印字位

置は、たとえば、図８の斜線部分なら（３：５）というように表現される（３行目の５カラム目という意味）。

さて、今度は、文字の方である。２バイトコードの文字の種類は、フォント・セット名で表わされ、代表的なものは、図９のとおりである。

フォント・セット名	文字縦ピッチ（LPI）	文字横ピッチ（CPI）
KM－７P	8	10
KM－９P	8	8
KM－12P	6	6
KM－14P	4	5
KM－18P	4	4
KM－24P	3	3

図９

図９中の文字縦ピッチは文字の縦幅、文字横ピッチは横幅を示す。（文字縦ピッチ、文字横ピッチは、日本電気のマニュアルの中では、それぞれ、行ピッチ、カラムピッチと書かれているが、前に述べた日本語プリンタに対する、行ピッチ、カラムピッチと混同する恐れがあるので、仮に、こういう呼び方をした） だから、たとえば行ピッチ８LPI を指定した行にKM－18Pの文字を印字すれば、上半分が前の行にはみ出てしまい、もし、前の行にも、文字を印字してある場合、前の行の文字の下半分は消えてしまう。

しかし、カラムピッチの場合は別である。つまり、プリンタ用紙に対するカラムピッチの指定は印字位置を定めるためのものにすぎず、文字列の印字には、影響を与えない。文字列の印字はカラムピッチに関係なく、その文字の文字横ピッチによって行われる。また、文字と文字の間隔を広げたい時には、書式制御プログラムの中で、その文字列に対する印字ピッチを指定すればよい。印字ピッチの種類は{20, 15, 12, 10, 8, 6, 5, 4, 3CPI}である。だが、注意しなくてはならないのは、その文字の文字横ピッチ以下の印字ピッチを指定してはいけない事である。なぜなら、文字と横隣りの文字が重なってしまうからである。

6）書式制御プログラム

さて、いよいよ、帳票の作成である。

COBOL 74やFORTRAN 77 のプログラムで帳票を作成しようと思えば、横線一本引くのにさえ、〃－〃を並べた定数を用意しなくてはならない。複雑な表を作り、しかも、そこへデータをあてはめるとなれば、かなりの労力が必要になる。

それに比べると、書式制御プログラムは、比較的楽に表を作成出来る。図形処理のサブルーチンをコールするような形で、線や表が描けるからである。

たとえば「BOX（26, 78）POS（11：1）；」と記述するだけで、長方形の箱が描ける。しかも、図形処理サブルーチンとは違って、座標の指定が行とカラムで行える事も帳票作成には有効である。

書式制御プログラムは、いくつかの情報を持つセクションから成り、書式セットという形で統

一される。図10に構成を図解した。（各セクションには、「 」でくくられた名前を付けているが、これは、ユーザ・プログラム側で任意に定義できるものであり、後に出てくる図16のプログラム例の中のセクション名と対応づけている）

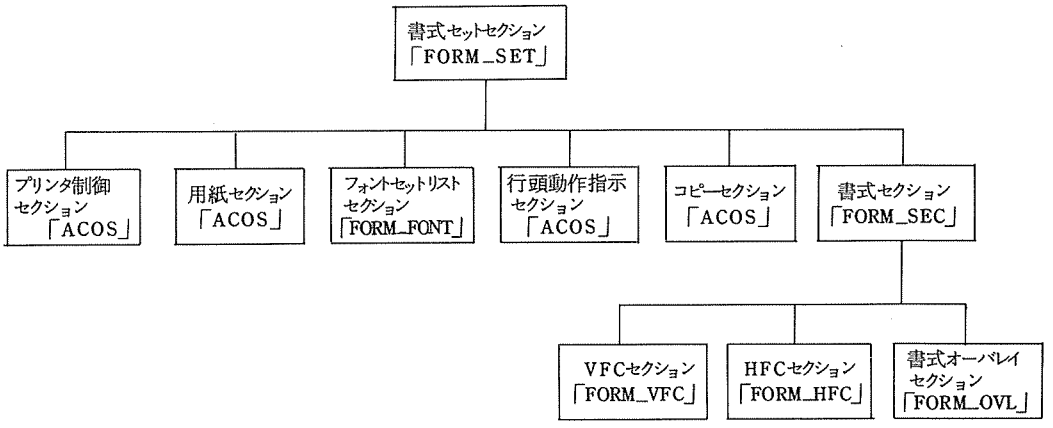


図10

○書式セットセクション

書式セットは、図のように、6つのセクションから成り立っている。各セクションは別々に定義し、その時にセクション名も与える訳だが、ここでは、その定義された各セクション名を指定する。

又、書式セット全体としての書式セット名は、このセクションで定義され、日本語データ出力プログラム◎（図1参照）では、使用すべき書式セットとして、ここでの名前を指定することになる。

○プリンタ制御セクション

日本語プリンタの動作（印字データ・エラーが発生した場合の処理等）を指定する。

○用紙セクション

印字する用紙のサイズを指定する。

○フォントセットリストセクション

使用する文字のフォントセット名を指定する。

○行頭動作指示セクション

JIPSにおける印字データの場合、ある一つの印字行に注目してみるとコード系（1バイトコード、2バイトコードの別）、フォントセットは、その行の先頭にある機能コード（たとえば「KIN」、「KOUT」など）によって表現される。しかし、もし機能コードが行の途中まで現わ

れなかった時、そこまでのデータを、どのコード系、どのフォントセットで印字するのかを指定しておくのが、このセクションである。

だから、もし行内に機能コードが全く無ければ、このセクションの指示通りに印字され、反対に行の先頭から機能コードによって、コード系、フォントセットが指定されていれば、このセクションの指示は無意味となる。

○コピーセクション

コピーの形式、コピー枚数を指定する。（詳しくは、書式オーバレイセクションのところで触れる）

○書式セクション

書式セクションは、VFC、HFC、書式オーバレイの3セクションに分かれていて、ここでは、その3つのセクション名の指定を行う。

○VFCセクション

ページ内の各行の行ピッチ、マージン（プリンタ用紙の上から何インチまでは印字に使用しない。つまり、一番目の行を何インチか下へずらす）、印字不能行（書式オーバレイセクションの所で説明する）の指定を行う。

また、たとえば、上から10行なら10行を印字に使用し、それより下の部分を使用しないのであれば、10行分だけの行ピッチを指定すればいい。

○HFCセクション

ページ内の全ての行に関して、カラムピッチ、マージン（プリンタ用紙の左端から何インチまでは、印字に使用しない。つまり1カラム目を何インチか右へずらす）、印字領域の幅（カラム数で表わす）の指定を行う。

○書式オーバレイセクション

書式オーバレイセクションは、図11のような構造を持つ。それぞれのブロックはコード化オーバレイ定義言語で記述して行く。

送付先名簿の場合として、説明して行く事にする。

入力レコード定義ブロックでは、帳票1ページ分の利用者データのフォーマット（図

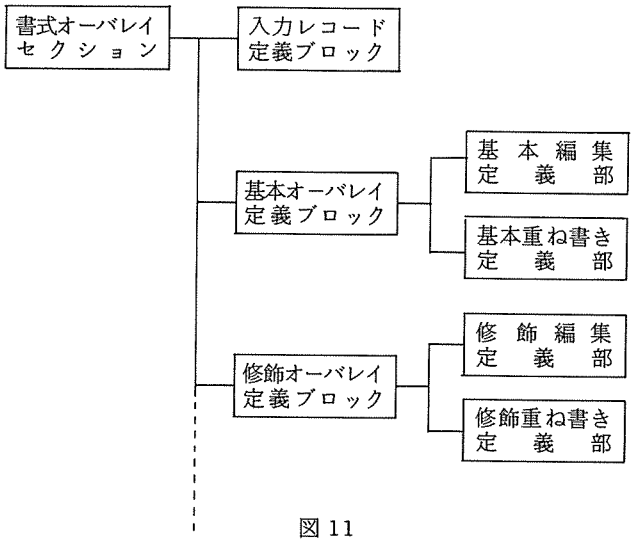


図11

7参照）を記述する。

基本オーバレイ定義ブロックは、帳票イメージそのものの記述であり、2つのブロックに分かれている。

まず、基本編集定義部。ここでは、入力レコード定義ブロックで指定したレコードを、帳票のどの位置へ、どの文字フォントセットで、どれくらいの印字ピッチで印字するのかを記述する。つまり、図12のようなイメージを作り上げるのである。

（速報）

58　5　30

大阪市立大学連絡所　　御中

経済学部	経済学科	○○ ○○	教授	1
文学部	人間関係学科	○○ ○○	教授	1
理学部	物理学科	○○ ○○	助教授	1
	化学科	○○ ○○	助教授	1
		○○ ○	院生（後）	1
	地学科	○○○ ○	教授	1
工学部		○○ ○○	教授	1
	建築学科	○○ ○○	助手	1
	建築工学科	○ ○○	講師	1
原子力基礎研究所		○○ ○○	助手	1
計算センター		連絡所		1

図12

このように、利用者データの各レコードがどの行に印字されるのかは、まえもって定められる訳であるが、その定められた行以外の全ての行を印字不能行（VFCセクションの説明参照）という。つまり、印字不能行とは、文字通りの印字できない行という意味ではなく、この基本編集定義部で参照されなかった行を示すのである。基本重ね書き定義部では、帳票の固定部分（表を形づくる罫線や文字定数）の記述を行い、図13のようなイメージを作り上げる。

送付先名簿

昭和　　年　　月　　日

下記のとうり配付方よろしくお願いいたします。

記

学　　部	学　　科	氏　　名	身　分	部　数

合計　　部

大阪大学大型計算機センター　共同利用掛

図 13

送付先名簿作成のための書式制御プログラムでは使用していないが、書式オーバレイセクションには、もう一つのブロックがある。

修飾オーバレイ定義ブロックである。これは、基本オーバレイ定義ブロックで作り上げたイメージ（図12、13）を一部変更して、数種の帳票を作り上げるためのものである。（このブロックは7つまで定義出来る）

もちろん、入力レコード定義ブロックは変更しないので、同じデータを相手に、数種の帳票を作る、つまりはコピー（カーボン紙による複写に相当する）なのである。

分かりやすく図解してみよう。（図 14 ）

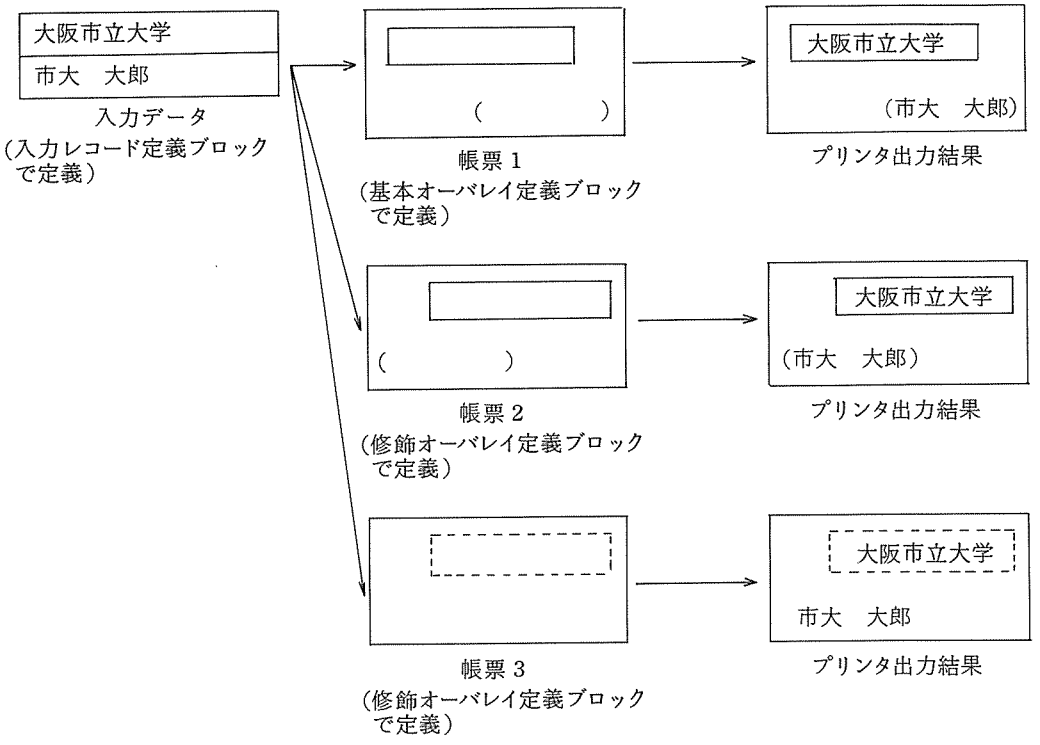
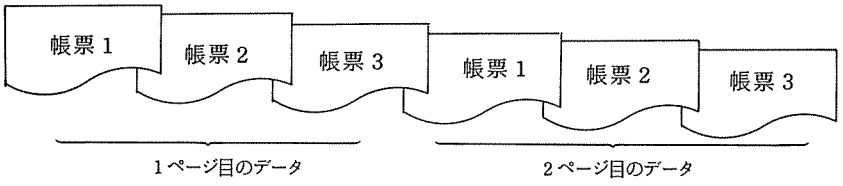


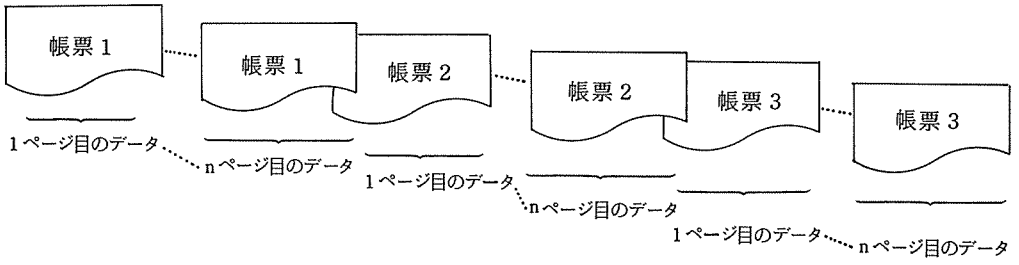
図 14

修飾オーバレイ定義ブロックでは帳票イメージを定義するだけで、実際のコピー動作は、前述のコピーセクション(図 10参照)で指定する。コピーの形式は図 15 のとおりである。

ページ単位のコピー



ファイル単位のコピー



両方を併用したコピー

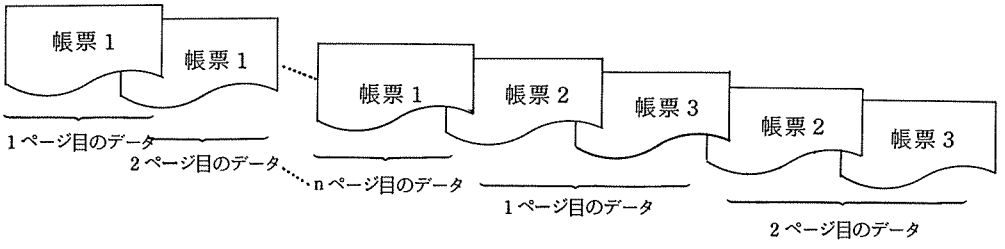


図 15

以上で、各セクションの説明を終わった訳であるが、最後に、書式制御プログラムの実行例を示して置く（図16）。

JCL

```
0010¥       IDENT   USERID,MAIL,R,,,JPR
0020¥       USERID  USERID¥PASSWORD
0030¥       JOBDEF  DEST=ONL
0040¥       PROGRAM .FORM
0050¥       LIMIT   25,200K,-6K,20000
0060¥       PRMFL   F*,W,R,USERID/FORM-DATA
0070¥       PRMFL   *F,W,R,USERID/FORM-INDEX
0080¥       PRMFL   D1,R,R,USERID/JISHO-DATA
0090¥       PRMFL   1D,R,R,USERID/JISHO-INDEX
0100¥       SYSOUT  PJ
0110¥       DATA    I*,,COPY
0120¥       JIS
```

ディレクティブ

```
0130//INITIALIZE LIST
0140//OPTION SHIFTCODE
0150//ADD LIST
```

フォントセットリストセクション

```
0160FORM_FONT:FONTSETLIST;
0170FONTSETKI=(KM_7P,KM_9P,KM_12P,KM_14P);
0180END;
```

VFCセクション

```
0190FORM_VFC:VFC;
0200MARGIN=1/3;
0210LINE=((4,NP)((4)*3)((4,NP)*6)((8,NP)*3)((4)*22)(4,NP)((4)*1)((4,NP)*3));
0220END;
```

HFCセクション

```
0230FORM_HFC:HFC;
0240MARGIN=0;
0250PITCH=6;
0260PRINTCOL=79;
0270END;
```

```
0280FORM_OVL:FORMOVERLAY;
0290VFC=FORM_VFC;
0300HFC=FORM_HFC;
0310FONTSETKI=KM_9P;
0320INPUT;
0330SEND_R:RECORD;
0340SEND_NAME:NCHAR(20);
0350END;
0360DATE_R:RECORD;
0370YY:NCHAR(2);
0380MM:NCHAR(2);
0390DD:NCHAR(2);
0400END;
0410RENRAKU_R:RECORD;
0420RENRAKUSHO:NCHAR(35);
0430END;
0440MEIBO_R:RECORD;
0450GAKUBU:NCHAR(30);
0460GAKKA:NCHAR(30);
0470NAME:NCHAR(15);
0480MIBUN:NCHAR(5);
0490BUSU:NCHAR(2);
0500END;
0510TOTAL_R:RECORD;
0520TOTAL_CNT:NCHAR(3);
0530END;
0540END;
0550BASE;
0560EDIT;
```

書式オーバレイセクション

```
0570SEND_OUT:RECORD(SEND_R);
0580SEND_NAME COL(40) FONT(KM_12P) CP(6);
0590END;
0600DATE_OUT:RECORD(DATE_R);
0610YY COL(47) CP(6);
0620MM COL(50) CP(6);
0630DD COL(53) CP(6);
0640END;
0650RENRAKU_OUT:RECORD(RENRAKU_R);
0660RENRAKUSHO COL(21) CP(6);
0670END;
0680MEIBO_OUT:RECORD(MEIBO_R);
```

```
0690GAKUBU COL(2);
0700GAKKA COL(28);
0710NAME COL(54);
0720MIBUN COL(67);
0730BUSU COL(73);
0740END;
0750TOTAL_OUT:RECORD(TOTAL_R);
0760TOTAL_CNT COL(70) CP(6);
0770END;
0780SEND_OUT LINE(2);
0790DATE_OUT LINE(3);
0800RENRAKU_OUT LINE(4);
0810MEIBO_OUT LINE((14,22,1));
0820TOTAL_OUT LINE(37);
0830END;
0840OVER;
0850BOX(26,78) POS(11:1) THICK;
0860HLINE(78) POS(13:1) THICK;
0870VLINE(26) POS(11:27,11:53,11:66,11:72);
0880NI'%カ゛ク%フ゛' POS(12:11) CP(6);
0890NI'%カ゛ク%カ' POS(12:37) CP(6);
0900NI'%シ%メイ' POS(12:57) CP(6);
0910NI'%ミ%フ゛ン' POS(12:68) CP(6);
0920NI'%フ゛%スウ' POS(12:74) CP(6);
0930NI'%ソウフ%サキ%メイホ゛' POS(1:35) FONT(KM_12P) CP(4);
0940NI'%シヨウワ' POS(3:45) CP(6);
0950NI'%ネン' POS(3:49);
0960NI'%カ゛ツ' POS(3:52);
0970NI'%ニチ' POS(3:55);
0980NI'%カキ%ノ' POS(7:29) CP(6);
0990NI'%トウリ%ハイフ%カタ%ヨロシク%オネカ゛イ%イタシマス' POS(7:32) CP(6);
1000NH'。' POS(7:50);
1010NI'%キ' POS(9:39);
1020NI'%コ゛ウケイ' POS(37:68) CP(6);
1030NI'%フ゛' POS(37:73);
1040NI'%オオサカ%タ゛イカ゛ク%オオカ゛タ%ケイサンキ' POS(38:57) CP(6);
1050NK'センター' POS(38:66) CP(6);
1060NI'%キヨウト゛ウ%リヨウ%カ゛カリ' POS(38:71) CP(6);
1070END;
1080END;
1090END;
```

書式セクション

```
1100FORM_SEC:FORM;
1110FORMOVERLAY=FORM_OVL;
1120END;
```

書式セットセクション

```
1130FORM_SET:FORMSET;
1140CONTROL=ACOS;
1150PAPER=ACOS;
1160FONTSETLIST=FORM_FONT
1170HEADOFLINE=ACOS;
1180COPY=ACOS;
1190FORM=FORM_SEC;
1200END;
```

JCL

```
1210¥       ENX
1220¥       ENDCOPY
1230¥       ENDJOB
```

図 16

このプログラムを実行することによって、帳票イメージ等の書式制御情報が書式制御ファイル⑦（図1参照）に格納されるのである。書式制御ファイルも、前述の語変換辞書ファイルと同じく、ランダム・ファイルであり、実際には、データ・ファイルとインデックス・ファイルの2つのファイルからなっている。

ライン番号60、70がそのファイルの定義文である。

ライン番号1130から1200までが、書式セットセクションの記述である。各セクションの名前を指定しているのだが、フォントセットセクションと書式セクション以外のセクション名はどれも「ACOS」になっている。これは、システム標準の既存セクションであり、ユーザ・プログラム側で定義する必要がない。だから、このプログラム・リストを見れば分かるように、「ACOS」という名のプリンタ制御セクションや用紙セクション等の定義は記述されていない。

7）日本語データ出力プログラム

もう一度、図1のフローチャートを見て欲しい。

今までの作業によって、利用者の漢字データの入った利用者データ・ファイル⑥と、書式制御情報の入った書式制御ファイル⑦が出来上がった。

日本語データ出力プログラムは、この2つのファイルを読み込んで送付先名簿を日本語プリンタへ出力するためのサービス・プログラムである。

図12、13で示したイメージがオーバレイされ、送付先名簿が出力される。

書式制御プログラムでは、前に述べたように、利用者データ・ファイルのレコードを、どう印字するかを制御する事が出来る訳で、もちろん、このレコードはいらないので、印字しないといった指定も可能な訳だ。

日本語データ出力プログラムでは、使用する書式制御を、書式セット名(第6項の書式セットセクションの説明を参照）によって指定するのだから、同じ利用者データ・ファイルをもとに、その都度、使用する書式制御を変えて、全く違った、名簿その他、各種リストを作成出来るのである。しかも、日本語データ出力プログラムは、ディレクティブごとに、入力するデータ・ファイルや書式制御を変える事が出来るので、同一アクティビティの中で、様々なリストを得る事が出来るという点も便利なのではないかと思われる。

それでは、最後に、完成した送付先名簿を図17に示しておこう。

送付先名簿
（速報）

昭和５８年　５月３０日

大阪市立大学連絡所　　御中

下記のとうり配付方よろしくお願いいたします。

記

学　部	学　科	氏　名	身　分	部　数
経済学部	経済学科	○○ ○○	教授	1
文学部	人間関係学科	○○ ○○	教授	1
理学部	物理学科	○○ ○○	助教授	1
	化学科	○○ ○○	助教授	1
		○○ ○	院生（後）	1
	地学科	○○○ ○	教授	1
工学部		○○ ○○	教授	1
	建築学科	○○ ○○	助手	1
	建築工学科	○ ○○	講師	1
原子力基礎研究所		○○ ○○	助手	1
計算センター		連絡所		1

合計　１１部

大阪大学大型計算機センター　共同利用掛

図 17