

Appendix B

In the following we present the different list types in the archive 'KB-LIST' covering the local elections 1909 to 1966. The list votes variables in the 'KB-ARCHIVE' are based upon single or combined list type votes. The list labels are reproduced in Danish because it has not been possible to translate many of the names into a suitable English.

Tab. B. List type identifications and labels in archive 'KB-LIST'.

(Source: "Statistiske Meddelelser": various publications containing the local elections 1909-68).

Folketingspartier og kombinationer af samme (Political parties).

coding value	List label
01	Socialdemokratiet (A)
02	Det radikale Venstre (B)
03	Det konservative Folkeparti, Højre (C)
04	Venstre, Reformvenstre, Moderate Venstre m.v. (D)
05	Danmarks Retsforbund (E)
06	Socialistisk Folkeparti, Venstreorienteret Samling (F)
07	Bondeparti (F)
08	Socialistisk Parti (1958) (J)
09	Danmarks kommunistiske Parti (K)
11	D.N.S.A.P. (N)
12	Slesvigsk Parti (S)
13	De Uafhængige (U)
14	Dansk Samling (R)
15	Andre politiske lister i 1937.
16	L.S. og Venstre
17	Erhvervspartiet (1921-25)
18	Socialdemokratiet + Radikale
19	Radikale + Retsforbundet
20	Venstre + Radikale, Venstre + Radikale + Retsforbundet
21	Venstre + Konservative
22	Socialdemokratiet + Venstre
23	A + B + D, A + B + E
24	B + C + D, B + C, B + C + D + E
25	A + B + C + D, A + B + C, A + C
26	E + C, E + D, E + D + C

Erhvervs- og stillingslister (Occupational lists).

coding value	List label
29	primærerhverv landmandsl., jordejere, gårdmænd og husmænd, hartkornsl., gmd. og skippere og fiskere, fiskeri og landbrug m.v.
30	gods- og proprietærbrug
31	gårdbrug gårdmænd, gårdejere m.v.
32	husmandsbrug husmandsl., boelsmænd m.v.
10	landbrugernes sammenslutning, L.S.
92	landbrugslister 1937
33	fiskeri og søfart havneliste, fiskerlejernes liste, søfart, skippere m.v.
34	landbrug og byerhverv m.v. gmd. + håndv. + husm., gmd. + husm. + næringsdrivende, landbrugere og handlende, landmænd og næringsdrivende fiskere + håndværkere + gårdejere, gårdm. + embedsm + handelsstand, erhverv og gårdejere m.v.
35	byerhverv i almindelighed næringsdrivende, handel + håndværk + industri, frie erhverv, håndværkere og forretningsdrivende m.v.
36	håndværk og industri
37	handel
38	erhvervslister erhvervsliste
27	danske erhvervslister dansk landbrugsl., dansk håndværksl. m.v.
28	tyske erhvervslister tysk landbrugsl., tysk handelsl m.v.
91	andre erhvervslister 1937
39	funktionærer og tjenestemænd postbude, højskoleliste, gendarmeriliste, flyvepladsen, lærere, embedsmænd
40	arbejdere faglige arbejdere, dansk arbejdsmandsforbund, fagforening, landarbejdere, arbejdere og husmænd m.v.

Ikke-erhvervsmæssige interessegruppelister (Non-occupational interest groups).

coding value	List label
42	indre mission, mission
43	ikke indre mission
44	valgmenighed
45	folkekirken sognemenighed, højkirkelig, luthersk mission
52	grundtvigianere

coding value	List label
46	andre religiøse lister methodister, frikirkelig, religiøs liste, brødremenighed, baptister, kristelig social
47	afholdslisters blå kors, ædruelighedsliste
48	ikke afholdslisters gæstgiverliste
49	kvindelisters kvindevalgret, husmødre
51	unge
55	foreninger idrætsforeninger, lejerforening, foredragsforening m.v.
53	demokratiske listers demokratisk forening, grundlovsforening m.v.
54	sociale listers social liberal
82	liberale listers
77	blandede politiske listers folkepartiet, politisk liste, forskellige partier, fremskridtsparti, samlingsparti m.v.
80	frisindede listers
74	danske listers
73	tyske listers
86	mindre bemidlede mindre skatteydere, småfolk, sygekassens liste, mindre ejendomsbesiddere, almindelig vælgerklasse, mindre hartkorn, arbejdsløshedsbevægelsen m.v.
87	mere bemidlede større skatteydere, højstbeskattede, antispygekasseliste, større hartkorn, besiddende m.v.
88	lønmodtagere fastlønnede, faste indtægter, fastlønnede og arbejdere m.v.
89	understøttelse aldersrente m.v.
90	grundejere
41	middelstand

Borger-, fælles- og samlingslister (Bourgeois lists).

coding value	List label
56	borgerlister borgerlig fællesliste, fælles borgerliste, det borgerlige parti, skatteborgerforening, borgerligt samarbejde, borgergruppen m.v.
57	lokale borgerlister stednavn + borgerliste, kommunal borgerliste, borgerlig særliste

coding value	List label
58	borger- og erhvervslistes borger- og landbol., borger- og fiskeril. m.v.
59	fælleslistes samlingsliste, samarbejdsliste
60	lokale fælles- eller samlingslistes
61	konservativ borger-, fælles-, eller samlingsliste
62	antisocialistes opposition mod Socialdemokratiet, blandingsliste uden Socialdemokratiet, socialistisk opposition m.v.
63	borger- og fælleslistes i 1937
64	frisindet eller liberal fælles- eller borgerliste
69	danske borgerlistes
75	tyske borgerlistes

Lokallistes (Local lists).

coding value	List label
65	vælgerforening kommunal vælgerforening
66	listes, der refererer til et geografisk område Fladsøliste m.v.
67	kommunallistes lokale listes, skoledistrikt, valgdistrikt, sognelistes m.v.
68	kommunale og lokale listes 1937
70	bylistes stationskvarteret, beboere i x-by m.v.
71	landlistes landboliste, markliste, overdrevsliste, strandliste m.v.
72	forstads- og villalistes
50	listes med tilknytning til kommunale opgaver skoleliste, sygehusliste, syge- og hjælpeforeningen, lærestrids- liste, et vejspørgsmål, skatteligningsliste, privatskoleliste m.v.
78	upolitiske listes uafhængig liste, upartisk liste, lokalpolitisk liste, tværpolitisk liste, friliste, frie vælgere m.v.
79	upolitisk interessegruppe eller borgerliste m.v. upolitisk borgerliste, fri vælgerforening m.v.
81	personlige listes det gamle sogneråd, løsgængerliste m.v.
83	udflytter- eller tilflytterlistes

Andre lister (Other lists).

coding value	List label
-----------------	------------

76	lister uden betegnelse liste a, b, c.... indtil 1933 inkl.
84	andre og uangivne lister 1937
85	øvrige lister sammenholdsliste, samfundsliste, blandingsliste, særliste, separatliste, privatliste, ny liste m.v.

Ingen lister (No lists).

coding value	List label
-----------------	------------

98	fællesliste valgt uden afstemning
99	manglende statistiske oplysninger

Appendix C

Data Files in the Archive.

C 1. Basic Data Files.

The following files contain the basic information for all file manipulations described in Appendix D.

IDARKIV

The file contains information about every commune unit in the archive which can be used for case selection, file integration and file aggregation procedures. IDARKIV is reproduced in Appendix E.

JEORIG

The file has information for the creation of JEARKIV.

OBARKIV

The file is used directly as archive file OBARKIV.

KBSTEM

The file contains information about voters and votes for the local elections 1909-66.

KBLIST

The file contains information about list types (see Appendix B.) and vote cast on the list types for the local elections 1909-66. KBSTEM and KBLIST are used for the generation of KBARKIV.

NKARKIV

The file has data in and after 1970 and is used directly in the archive as NKARKIV.

C 2. Basic Archive Files.

JEARKIV

Created from JEORIG. See C 1.

KBARKIV

Created from KBSTEM and KBLIST. See C 1.

Other Basic Data Files used as Basic Archive Files are: IDARKIV, OBARKIV and NKARKIV

C 3. Integrated Archive Files (IDARKIV, JEARKIV, OBARKIV, KBARKIV).

IGARKIV

The file contains information from IDARKIV and from one or more of the files: JEARKIV, OBARKIV, and KBARKIV. See Section 5.

INARKIV

The file has data from IDARKIV and NKARKIV. See Section 7.

C 4. Aggregated Archive Files.G6ARKIV

The file is a 1966 aggregation of a version of IGARKIV. See Section 6.

G7ARKIV

The file is a 1970 aggregation of a version of IGARKIV. See Section 7.

N7ARKIV

The file is a 1970 aggregation of INARKIV. See Section 7.

C 5. Integrated Archive Files (G7ARKIV, N7ARKIV).GNARKIV

The file is a combined version of G7ARKIV and N7ARKIV. See Section 7.

Appendix D

A Programmer's Guide to the Archive

D 1. Introduction

The various archive files described in the former sections have been created by means of a number of program routines assembled in the so-called, 'PG-ARCHIVE'. All programs have been written in the Fortran Extended language and run on a CDC 6400 computer under the SCOPE 3.3 operative system at RECAU, the Regional Computer Center at the University of Aarhus.

It means that the following description only addresses itself to users familiar with CDC computers and specially the SCOPE 3.3 operative system.

In a later edition of the Archive Manual, the programs will be presented under the CDC KRONOS operative system and probably too in versions adjusted to other computers, such as UNIVAC at RECKU, the Regional Computer Center at the University of Copenhagen.

The programs in its CDC SCOPE 3.3 version have been stored in an UPDATE mode as an OLDPL file which means that the programs can be transformed by inserting various fortran text lines before written on the COMPILE file. By inserting fortran text lines the user can control the program execution in a number of predetermined ways.

The number of text line insertions has been standardized and reduced to a minimum, so that the user control of the programs in many ways resembles program packages, such as SPSS and OSIRIS. The difference is mainly, that the user in our case must have a certain knowledge of Fortran programming.

The various programs in the 'PG-ARCHIVE' have been divided in several *decks, so that in an UPDATE run in NORMAL MODE only those decks which have been modified using a number of *INSERT cards or specified on a *COMPILE card will be written on the COMPILE file.

In the following sections the different program routines will be presented, what they perform and how to activate and control them during a number of text line insertions.

D 2. The Creation of 'JE-ARCHIVE'

Program Feature

The program creates the 'JE-ARCHIVE' using 'JEORIG' as input data. The 'JEORIG' is almost identical to the archive established by Jørgen Elklit¹⁾ apart from the fact, that a second variable has been inserted in sequence 002 indicating the corresponding commune identification on the 'KB-ARCHIVE'.

1) K-H Bentzon et al. Kommune Data Arkivet 1909-68, pp. 11-43.

In addition more statistical data have been added. Using the inserted variable 002 as an aggregational variable, a number of units are aggregated having the same identification number¹⁾. By this procedure the number of commune units are reduced from 1732 to 1500. As a result of the aggregation a number of identification variables are transformed or suppressed:

Variable number	JEORIG ²⁾	JE-ARCHIVE ³⁾
001	Original identification	Identification from KB-ARCHIVE
002	Identification from KB-ARCHIVE	First original identification in the aggregation group
003	Commune name	Commune name
004	(cont.)	(cont.)
005	(cont.)	(cont.)
006	date of creation	date of creation in reversed order (see p. 8) for the first commune in the aggregation group
007	date of termination	date of termination in reversed order for the last commune in the aggregation group
008	kind of boundary change	Number of aggregated units
009	number of persons involved in boundary change	number of voters in 1920
010	first commune involved in boundary change	number of votes cast in 1920
011	second commune involved in boundary change	Social Democrats in 1920
012	third commune involved in boundary change	The Radical Liberals in 1920
013	number of voters in 1920	The Conservatives in 1920
.	.	.
.	.	.
.	.	.

1) see p. 6 2) K-H Bentzon, op.cit. p. 19 3) pp. 6-8.

The user can control the program execution by determining what variables from 'JEORIG' which shall be aggregated and written upon 'JE-ARCHIVE'.

Update Deck

*DECK JEARKIV, contains main program 'JEARKIV'.

Files

(Files are listed in the following order on the Program Card)

TAPE1: reads information from JEORIG

TAPE2: writes information on JEARKIV

TAPE3: writes a control transcript from JEARKIV

TAPE4: scratch file

OUTPUT: normal output print from program execution

Update Control Cards

*IDENT xxxx

xxxx is any name identifying the following text line insertions.

*INSERT JEARKIV.6 (R)¹⁾

INTEGER VARA(), VARB()

The number in the two brackets indicates the number of variables from JEORIG which shall be aggregated and written on JEARKIV, i.e. from variable 013 on JEORIG corresponding to variable 009 on JEARKIV.

*INSERT JEARKIV.7 (R)

VARANT = X

X equals the number inside the brackets in the former card.

*INSERT JEARKIV.8 (R)

1 FORMAT ()

Format on JEORIG.

*INSERT JEARKIV.9 (R)

9 FORMAT ()

Format on JEARKIV.

*INSERT JEARKIV.11 (O)

KONTROL = 0

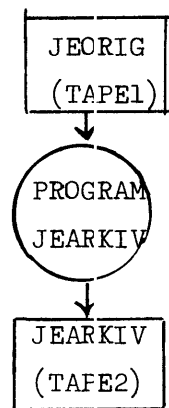
If specified no control transcript will be written on TAPE3.

*INSERT JEARKIV.12 (O)

REWIND = 0

If specified TAPE1, if CONTROL is set to zero, and TAPE2 will not be rewound after program termination.

1) R = required O = optional

Flow Chart of program JEARKIVSCOPE 3.3 Control Card Deck creating standard version of JE-ARCHIVE

xxxxxx,CM47000,T700,NT1. (specification of job card name)

ACCOUNT,X,Y,Z. (specification of user's account card)

LIMIT,1000.

RFL,100.

RPACK,XXX,E,XXX. (specification of user's RPACK)

REQUEST,JEARKIV,PK,XXX. (specification of user's RPACK)

RFL,35000.

UPDATE,P=PGARKIV,D.

RFL,47000.

FTN,I=COMPILE.

REDUCE.

LGO,JEORIG,JEARKIV.

RFL,10000.

COPYSBF,TAPE3.

7/8/9

*IDENT KONTROL

*INSERT JEARKIV.6

INTEGER VARA(261), VARB(261)

*INSERT JEARKIV.7

VARANT=261

*INSERT JEARKIV.8

1 FORMAT (2I7,2X,3R10,3X,11I7,14(/,18I7),/5I7)

*INSERT JEARKIV.9

9 FORMAT (2I7,2X,3R10,3X,11I7,14(/,18I7),/1I7)

6/7/8/9

It is assumed that JEORIG and PGARKIV are files stored on RPACK.

D 3. The Creation of 'KB-ARCHIVE'

Program Feature

When it was decided to establish an archive containing data from the local elections a serious problem was encountered because of differences in number and names of the lists put up from one commune to another. As indicated in appendix B, a very extensive list classification of 92 numbers was established, but it meant, that it was extremely space wasting to allocate a fixed variable position to every list type. Instead, a special archive, the socalled 'KBLIST' was created having floating variable positions. For every list put up in a commune at a given election a unit record was created on 'KBLIST' having the following variable outline:

Variable number	Variable name
001	Commune identification Position 1-2: two last digets in the year of election Position 3-4: County number. See p. 31. Position 5 : Commune type. See p. 32. Position 6-8: Within county commune number. See p. 32.
002	List type code (See Appendix B)
003	Number of votes on the list

Number of variables: 3

Fortran Format (I8,2X,I2,2X,I6)

Number of Cases: 79628

Sorting Order: 1. Position 3-8 on variable 001 in ascending order.
2. Position 1-2 on variable 001 in ascending order.

For variables containing information about male and female voters, votes and representation a more normal file layout was adopted. The socalled file 'KBSTEM' has the following variable outline:

Variable number	Variable Name
001	Commune identification (similar to 'KBLIST')
002	Male Voters
003	Female Voters
004	Male Votes
005	Female Votes
006	Total Valid Votes
007	Elected Male Candidates
008	Elected Female Candidates

Number of Variables: 8

Fortran Format (I8,7I7)

Number of Cases: 9228

(this number will be increased considerably in the near future, when data from all local elections concerning male and female voters, votes, etc. have been coded)

Sorting Order: As 'KBLIST'

Using 'KBLIST' and 'KBSTEM' as input files, the program performs the following file manipulations:

- 1) The main program 'SKTAPE2' aggregates on the user's choice a number of list types from 'KBLIST' and assigns to the reduced list classification fixed variable position for every election in which a given commune unit has existence.
- 2) Owing to the fact that a number of commune units have not been in existence in the whole period, i.e. from 1909 to 1966, the subprogram 'DKTAPE3' creates number of commune units for those elections in which the commune is non-existent. The created commune units consist of a simulated commune number followed by a number of zero value variables. By this procedure we get a double unit system: a number of supra units each consisting of 15 subunits in ascending time order, corresponding to the 15 local elections from 1909 to

1966. Using the supra units as basic units enables the user to make longitudinal analysis of the list votes.

- 3) The subprogram 'DKTAPE5' impose the unit structure described under point 2 on 'KBSTEM' by inserting the commune number non-existent on 'KBSTEM' followed by 7 zero value variables. By this procedure the revised 'KBLIST' and 'KBSTEM' get the same unit structure.
- 4) Finally subprogram 'KBARKIV' merges the revised 'KBLIST' and 'KBSTEM' into a single file based solely upon the supra unit structure, the so-called 'KBARKIV'. Each unit is identified by a commune number without time indication containing information from all elections, including the original variables as well as the inserted zero value variables during the former file manipulations.

Update Decks

*SKTAPE2: Contains mainprogram 'SKTAPE2'.
 *DKTAPE3: Contains subroutine 'DKTAPE3'.
 *DKTAPE5: Contains subroutine 'DKTAPE5'.
 *KBARKIV: Contains subroutine 'KBARKIV'.

Files

(Files are not listed in the following order on the program Card. See p. D9)

TAPE1: reads 'KBLIST'. See p. D 5.

TAPE2: writes first revised version of 'KBLIST':

Variable number	Variable name
001	Commune identification (similar to variable 001 on 'KBLIST')
002	Kind of local election (See p. 32)
003	Number of lists put up
004	Total votes for lists put up
005	Number of votes on first single or aggregated list type
006	Number of votes on second single or aggregated list type
007	Number of votes on third single or aggregated list type
.	.
.	.
.	.
nnn	Number of votes on nnn'th single or aggregated list type

Number of variables: 4 + number of single or aggregated list types

Fortran format: depending upon number of variables

Number of cases: 19914

(the number equals the total sum of communes in every election, i.e. the number of communes in 1909 + the number of communes in 1913..... + the number of communes in 1966)

Sorting order: as 'KBLIST'

TAPE3: writes the second revised version of 'KBLIST'.

This file is identical to TAPE2 apart from the number of cases, which has been increased in order to establish a longitudinal supra unit structure. See point 2 on page D 6-7. It means that the subunits can be structured in series of 15, corresponding to the number of elections:

09xxxxxx

13xxxxxx

17xxxxxx

21xxxxxx

25xxxxxx

29xxxxxx

33xxxxxx

37xxxxxx

43xxxxxx

46xxxxxx

50xxxxxx

54xxxxxx

58xxxxxx

62xxxxxx

66xxxxxx

Number of cases: 24825

(the number equals 15 x 1655, which means, that 1655 geographical different commune units have been identified on 'KBLIST')

TAPE4: reads 'KBSTEM'. See p. D 6.

TAPE5: writes the first revised version of 'KBSTEM':

This file is identical to TAPE4 apart from the number of cases, which has been increased in order to establish a unit structure identical to TAPE3. See point 3 on page D 7.

Number of cases: 24825

TAPE6: writes 'KBARKIV'. See pp. 29-68

TAPE7: writes a control transcript of 'KBARKIV'

OUTPUT: normal output print from program execution

The files are ordered in the following way on the Program Card in order to facilitate file name alterations on the LGO control card:

TAPE1 = KBLIST

TAPE4 = KBSTEM

TAPE6 = KBARKIV

TAPE7

TAPE2

TAPE3

TAPE5

OUTPUT

Update Control Cards

* IDENT XXX (0)

XXX is any name identifying the following text line insertions.

* INSERT SKTAPE2.5 (R)¹⁾

INTEGER VARA(X), VARB(Y), VARC(Z,15), VARD(Z,15)

X = number of variables on TAPE2, i.e. 4 + number of single and/or aggregated list types

Y = X - 1

Z = Y + 7

* INSERT SKTAPE2.6 (R)

DATA LISTED/...../

As indicated in point 1 page D 6, the mainprogram 'SKTAPE2' performs the function of reducing the origins list classification to a more suitable number of different list types which can be allocated a fixed variable position for a given commune at every election. The procedure involves a number of list vote aggregations for those list types which are to be combined into a single list. Using the DATA LISTEID/ / text line insertion the user can determine to what extent he wants the original list type classification to be reduced, what list he will have transferred unaltered and what lists he will have aggregated.

Unfortunately the DATA LISTEID/ / text line is a bit difficult to handle, so we have to make some remarks about the logic of the program. LISTEID is a predefined integer array of 92 cells or put in another way, the array consists of 92 indexed variables from 1 to 92. Consequently a

1) R = required 0 = optional

given cell or variable in an array can give us two kinds of information, on the one hand side the index or sequence number of the variable in the array and on the other hand side the assigned value of the variable. Using these two kinds of information and having an extended and a reduced array the program can perform a number of functions in a very simple way:

- 1) reducing the number of list types
- 2) giving each list type a fixed position
- 3) determining the sequence order of list types

Primarily the user must decide how and to what extent he wants to reduce the original list classification of 92 types. The reduced classification, if reduced at all, determines the array definition of VARA, VARB, VARC and VARD. The next step is to inform the program using the DATA LISTEID/ / what list types to be combined and aggregated and their sequence order. The user must insert 92 integers separated by commas between the two slashes by using a number of FORTRAN continuation text lines.

When the program reads a list type code in a given commune at a given election from 'KBLIST' the variable on array LISTEID having the same sequence number is activated. The value given the activated variable by the user on the DATA LISTEID/ / text line tells the program, that the variable on array VARA having that value as sequence number plus 4 should be activated. Finally the activated variable on array VARA is given the value of the number of votes cast for the list type in question. If the activated variable on array VARA already has a value above zero because other earlier read list types in the commune have been assigned to the same variable, the list votes are aggregated. If for a given commune at a given election a list vote variable on array VARA is not activated because the lists assigned to that variable have not been put up the variable remains in its initial zero value position.

If the explanation seems unclear to the reader, a number of examples will clarify how to use the DATA LISTEID/ / text lines.

If the DATA LISTEID/ / is filled with 92 ones, and only ones, the array VARA must be indexed to 5 and variable 004 and variable 005 on TAPE2 will have the same value.

If DATA LISTEID/ / is filled with 92 different numbers, the range must be from 1 to 92, the array VARA must be indexed to 96 and no list types will be aggregated.

If DATA LISTEID/ / is first filled with 30 ones and then with 62 twos, the array VARA must be indexed to 6 and votes for list type 1 to 30 will be aggregated and the result written on variable 005 on TAPE2 while votes for

list types 31 to 92 will be written on variable 006. We can summarize the use of DATA LISTEID/ / by formulating a number of rules:

- 1) the text lines must contain exactly 92 numbers separated by commas.
- 2) the maximum range is 92.
- 3) a rank order of the numbers must have as the lowest number 1 and an incremental factor of 1. The observed order will often be quite different.
- 4) the highest number indicates consequently the number of different list types on TAPE2.

*INSERT SKTAPE2.7

VARANTA=X (R)

X equals the index number of VARA

*INSERT SKTAPE2.8

11 FORMAT() (R)

Format for TAPE2. The number of variables to be formatted equals X in the former text line. It must be remembered, that the first variable, i.e. the commune identification must at least be given 8 positions. For the rest of the variables 7 positions will assure, that no variable will be out of range.

*INSERT SKTAPE2.10

KONTROL = 0 (0)¹⁾

If specified no control transcript will be written on TAPE7

*INSERT SKTAPE2.11

REWIND = 0

If specified TAPE1 and TAPE4 will not be rewound, if KONTROL is also set to zero TAPE6 will not be rewound either.

*INSERT DKTAPE3.8

2 FORMAT() (R)

Format for TAPE3 similar to TAPE2 apart from the fact, that variable 001, i.e. commune identification, must be split up into two variables, the first indicating the year of election, i.e. two first positions, the second indicating the commune number, i.e. the last six positions of the original variable.

*INSERT DKTAPE5.8

1 FORMAT() (R)

Format for TAPE3, but reads only variable 001 and 002, i.e. the whole

1) 0 = optional

commune identification of 8 positions as one variable. It must be remembered to skip line two, three, etc. by inserting slashes.

*INSERT KBARKIV.8

4 FORMAT()

(R)

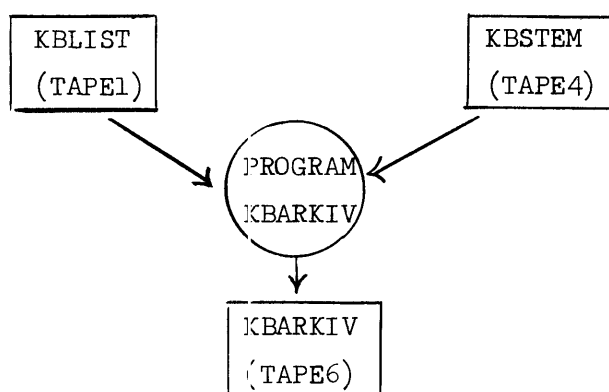
Format for TAPE3

*INSERT KBARKIV.9

8 FORMAT()

Format for TAPE6 or 'KBARKIV'. Number of variables to be specified equals number of indexed variables in array VARC or VARD plus 1, see page D 9. Put into a formula: $Z \times 15 + 1$. It should be realized, that the number of variables for a unit on 'KBARKIV' equals: commune identification + 15 x ((number of variables on TAPE3 - 2) + (number of variables on TAPE5 - 2)), where the two 2 digets in the two inner brackets represent the split commune identification.

Flow Chart of Program KBARKIV



SCOPE 3.3 Control Card Deck creating Standard Version of KB-ARCHIVE

XXXXXX,CM47000,TY300,NT1.	(specification of user's job name)
ACCOUNT,X,Y,Z.	(specification of user's account card)
LIMIT,1000.	
RFL,100.	
RPACK,XXX,E,XXX.	(specification of user's RPACK)
REQUEST,KBARKIV,PK,XXX.	(do.)
RFL,35000.	
UPDATE,P=PGARKIV,D.	
RFL,47000.	
FTN,I=COMPILE.	
REDUCE.	
LGO,KBLIST,KBSTEM,KBARKIV.	
RFL,10000.	

COPYSBF,TAPE7.

8/8/9

* IDENT KONTROL

* INSERT SKTAPE2.5

INTEGER VARA(47),VARB(46),VARC(53,15),VARD(53,15)

* INSERT SKTAPE2.6

DATA LISTEID/ 1,2,3,4,5,6,7,8,7,9,11,12,10,38,7,13,14,16,16,15,
A16,16,16,16,36,37,22,22,20,19,21,23,25,25,25,25,31,24,32,26,26,
B26,26,27,27,28,29,28,26,17,17,29,33,35,33,34,35,33,33,39,17,35,
C35,35,40,36,35,35,37,36,37,35,17,18,18,17,18,17,35,43,35,30,32,
D31,30,32,42,41/

* INSERT SKTAPE2.7

VARANTA=47

* INSERT SKTAPE2.8

11 FORMAT(I8,6X,17I7,/,19I7,/,10I7)

* INSERT DKTAPE3.8

2 FORMAT(I2,I6,6X,17I7,/,19I7,/,10I7)

* INSERT DKTAPE5.8

1 FORMAT(I8,//)

* INSERT KBARKIV.8

4 FORMAT(I2,I6,6X,17I7,/,19I7,/,10I7)

* INSERT KBARKIV.9

8 FORMAT(44(18I7,/),4I7)

6/7/8/9

It is assumed, that PGARKIV, KBLIST and KBSTEM are stored on the user's RPACK.

D 4. The Creation of the 'IG-ARCHIVE'.

Program Feature

The setting up of the three original data archives, i.e. 'JE-ARCHIVE', 'OB-ARCHIVE' and 'KB-ARCHIVE' resulted in the very unfortunate situation, that to a very high extent information for a given geographical commune unit were dispersed on three different files preventing simultaneous use of the data from the three files. In fact the first step in order to establish a more usable data archive was to integrate the original data archives into a single archive, where a given geographical commune unit should contain information from all three archives. Owing to the fact, that the three archives used different unit identification systems, it was necessary to create an identification list telling what identification numbers in the three archives which represented the same geographical commune unit. Variable 001 to 003 in the 'ID-ARCHIVE' give us information about corresponding commune identification numbers.¹⁾ In the case where a commune is non-existent in one or two of the archives, the identification number has been filled with 9 digets. Finally, a program has been worked out which for every commune unit in the 'ID-ARCHIVE' selects those commune units in the three data archives having a commune identification number similar to variable 001 to 003 in the 'ID-ARCHIVE' and merges the three units to a single unit. If a commune unit is non-existent in one or two of the archives, the variable positions for that archive in the unified archive, the socalled 'IG-ARCHIVE', are zero filled, so that every unit in the 'IG-ARCHIVE' gets the same size.

While the logic of the program is extremely simple, the technical solution is a bit difficult owing to the fact, that it is a very high computer time consuming job to search for data spread around in big sequential files. Instead a randomization procedure has been adopted, which enables the user to select data spread around in a file at a very high speed.

In a normal formatted FORTRAN file, the variables, which in CM, Central Memory, are represented each by a binary filled 60 bits word, are transformed into a character strings on the mass storage device, i.e. DISC, where each character is represented by a 6 bits configuration according to the DISPLAY CODE. The organization of the string is determined by the FORTRAN FORMAT statement. The physical representation on the DISC is organized in a number of RECORD BLOCKS, for the time being each consisting of 28 PRU's, scattered around on the DISC. A table in LOW CODE of CM contains position and sequence of the BLOCKS and

1) See p. E1.

PRU's on the DISC. Since the user has no access to that table, the file operates as sequential for the user, meaning that a file always has to be written in the same order as it has been written. Accordingly, if the user wants to get access to a given case in the file, he has to read all the previous ones first. Since the ordering of commune identifications on the original archive files are different from the case order on the 'ID-ARCHIVE', a merging of the three files requires an enormous computer time, because each file on the average has to be fully read 849 times, i.e. approximately the number of commune units on 'ID-ARCHIVE' divided by 2.

In order to solve that problem, the program using a special RANDOM BACK-ING STORAGE ROUTINE for FTM¹⁾ generates scratch random editions of the original data archives, if all three has to be merged. The randomization gives us two important advantages:

First we get access to the physical representation of the randomized file on the DISC by means of POINTER, which tell the I/O routines the position on the DISC of desired information. The method is logically equivalent to indexed variables described in section D 3, where we used both the position in the array and the value of the variable as information. Using the POINTER we do not have to read the previous cases in a file first, but can direct the read routine immediately to the data identified by a given pointer number.

Next the random routine stores the data as binary information, i.e. identical to the CM representation of the variables, which means, that a time consuming conversion process to character strings is avoided. Each variable is stored as an unformatted binary words of 60 bits which can be directly read into Central Memory. The program organizes the randomization process in such a way, that a pointer is assigned to every case in the file. The pointer and the commune identification number are then stored in a double array (2,X), where X is identical to the number of cases in the archive file in question. Accordingly, when the randomization process has finished we are left with three double arrays, one for each archive file, if all three archive files are randomized.

The next step is to read successive cases from 'ID-ARCHIVE'. For a given case, the program, using a special subroutine, finds that identification in the array for 'JE-ARCHIVE', that is identical to variable 002 in 'ID-ARCHIVE'. The corresponding pointer is then used to read the variables which are contained in the case having the commune identification in question. The same procedure for variable 003 and 001 is then used for 'OB-ARCHIVE' and 'KB-ARCHIVE'. Finally all variables are written as one case on a sequential and formatted file, the so-called 'IG-ARCHIVE'. If an identification in 'ID-ARCHIVE' is filled with 9 digets, the variables for the archive in question are zero filled on 'IG-ARCHIVE'.

1) Jens U. Mouritzen: Random Backing Storage Routines for Algal, FTM, Pascal, run. RECAU 72-18.

The program has a number of options which enables the user to create different versions of 'IG-ARCHIVE':

- 1) The user can determine the sequence of cases by resorting the 'ID-ARCHIVE' before program execution. The sorting procedure is performed by a standard program SORTMRG.¹⁾
- 2) The user also decides how many and the combination of the original archive files which must be randomized and written on 'IG-ARCHIVE'.
- 3) Using special format statements it is possible to extract only a subset of variables from a given archive.
- 4) A select procedure can select a subset of cases from 'ID-ARCHIVE' and consequently on 'IG-ARCHIVE'.
- 5) Finally different lines from a case on 'ID-ARCHIVE' can be written on the corresponding case on 'IG-ARCHIVE'.

The reason will be explained later on.

Update Decks

*IGARKIV: Contains mainprogram 'IGARKIV' and subroutines 'SQG' and 'UDSKRIV'.
 *JERAND : Contains subroutine 'ARAND'.
 *OBRAND : Contains subroutine 'BRAND'.
 *KBRAND : Contains subroutine 'CRAND'.
 *IJE : Contains subroutine 'UNIA'.
 *IOB : Contains subroutine 'UNIB'.
 *IKB : Contains subroutine 'UNIC'.
 *IJEOKB: Contains subroutine 'UNIABC'.
 *IJEOKB : Contains subroutine 'UNIAB'.
 *IJEKB : Contains subroutine 'UNIAC'.
 *IOBKB : Contains subroutine 'UNIBC'.

Files

(Files are listed in the following order on the Program Card)

TAPE1: Reads 'JEARKIV'
 TAPE2: Reads 'OBARKIV'
 TAPE3: Reads 'KBARKIV'
 TAPE7: Reads 'IDARKIV'
 TAPE8: Writes 'IGARKIV'
 TAPE9: Writes a control transcript of 'IGARKIV'
 TAPE4: Writes a randomized version of 'JEARKIV'
 TAPE5: Writes a randomized version of 'OBARKIV'
 TAPE6: Writes a randomized version of 'KBARKIV'
 OUTPUT: Writes normal print output from program execution.

1) CDC 6000 version 3. SORT/MERGE. Reference Manual. Nr. 60252600E.

Update Control Cards

*IDENT XXX (O)¹⁾

XXX is any name identifying the following text line insertions.

*INSERT IGARKIV.8 (R)¹⁾

INTEGER A(X), AA(X), B(Y), BB(Y), C(Z), CC(Z)

As you recall from the former section about program feature, information from Central Memory is directly transferred to the mass storage device without use of BUFFER and change of data representation. It means that a data transmission from CM to DISC and vice versa always will contain 1 PRU = 64 words = 64 CM binary variables. It is possible for a given random writing order (WRITRN)²⁾ to specify more than one PRU's to be transferred from CM to DISC, but it means that a variable array to be transferred always must be a multiple of 64. If the actual number of variables, i.e. a case on an archive file, do not make up a multiple of 64, the array must be defined to the nearest higher multiple of 64. If e.g. there is 630 variables to be transferred in a single WRITRN order, i.e. identified by a single POINTER, the array must be indexed to 640 = 10 x 64. It means, that the last 10 variables will be empty. According to the reasoning above, X, Y, and Z must be defined in the following way:

X = the nearest higher multiple of 64 from the number of variables on a case on 'JEARKIV'.

Y = the nearest higher multiple of 64 from the number of variables on a case on 'OBARKIV'.

Z = the nearest higher multiple of 64 from the number of variables on a case on 'KBARKIV'.

If not all three archives must be randomized, only those arrays for the archives to be randomized have to be defined. If e.g. only 'JEARKIV' and 'KBARKIV' are to be randomized, it is only necessary to define the arrays: A(X), AA(X), C(Z), CC(Z).

*INSERT IGARKIV.9 (R)

INTEGER REGISTA(2,XX), REGISTB(2,YY), REGISTC(2,ZZ)

By inserting this text line, the user defines the double arrays mentioned above combining POINTER and case identification:

1) R = required O = optional

2) See Jens U. Mouritser, op.cit. p.8.

XX = number of cases in 'JEARKIV'.
 YY = number of cases in 'OBARKIV'.
 ZZ = number of cases in 'KBARKIV'

Like the former text line only the arrays for the archive files to be randomized have to be defined.

*INSERT IGARKIV.10 (R)
 IXXX=1

By this text line the user determines the number and combinations of archive files to be randomized and finally written upon 'IGARKIV':

IXXX=IJE: 'JEARKIV' will be randomized and written upon 'IGARKIV'.
 IXXX=IOB: 'OBARKIV' will be randomized and written upon 'IGARKIV'.
 IXXX=IKB: 'KBARKIV' will be randomized and written upon 'IGARKIV'.
 IXXX=IJEOKB: 'JEARKIV', 'OBARKIV', and 'KBARKIV' will be randomized
 and written upon 'IGARKIV'.
 IXXX=IJEOB: 'JEARKIV' and 'OBARKIV' will be randomized and written upon
 'IGARKIV'.
 IXXX=IJEKB: 'JEARKIV' and 'KBARKIV' will be randomized and written upon
 'IGARKIV'.
 IXXX=IOBKB: 'OBARKIV' and 'KBARKIV' will be randomized and written upon
 'IGARKIV'.

*INSERT IGARKIV.11 (R)
 PRUSA=XXX
 PRUSB=YYY
 PRUSC=ZZZ

Remember that the file randomization requires, that variables will be transferred from CM to DISC and vice versa in PRU's, where one PRU equals 64 words on binary CM variables. Consequently:

XXX=X/64
 YYY=Y/64
 ZZZ=Z/64

where X, Y, and Z have been defined in the second text line insertion. Only the variables for the archive files to be randomized have to be defined.

*INSERT IGARKIV.12 (R)
 IA=XXXX
 IB=YYYY
 IC=ZZZZ

XXXX = number of variables to be read from 'JEARKIV' and finally written upon 'IGARKIV'.

YYYY = number of variables to be read from 'OBARKIV' and finally written upon 'IGARKIV'.

ZZZZ = number of variables to be read from 'KBARKIV' and finally written upon 'IGARKIV'.

Only the variables for the archive files to be randomized have to be defined.

*INSERT IGARKIV.13

NA=XX

NB=YY

NC=ZZ

where XX, YY, and ZZ are defined as the number of cases in each archive file. See the second text line insertion. Only the variables for the archive files to be randomized have to be defined.

*INSERT IGARKIV.14

(0)

IDARKIV=N

The text line insertions controls to what extent lines from 'IDARKIV' are written upon 'IGARKIV'. If the text line is omitted, IDARKIV is automatically set to zero, which instructs the program to write the first line from each case in 'IDARKIV' on 'IGARKIV'. If the program is only used to integrate the archive files, this procedure should be used. If N is set to 1, the first and the second line from 'IDARKIV' are written upon 'IGARKIV'. This procedure must be used, if 'IGARKIV' has to be aggregated to the 1966 commune structure. If N is set to 2, the first and the third line from 'IDARKIV' are written upon 'IGARKIV'. This procedure must be used, if 'IGARKIV' has to be aggregated to the 1970 commune structure. The two last options, i.e. N set to 1 or 2 must be used in connection with the case aggregational procedures described in section D 5. and D 6.

*INSERT IGARKIV.16

(0)

KONTROL=0

If specified no control transcript will be written on TAPE9.
TAPE8 will not be rewound.

*INSERT IGARKIV.17

(0)

REWIND=0

If specified TAPE1, TAPE2, and TAPE3 will not be rewound.

*INSERT JERAND.10

(0)

9 FORMAT ()

Format for 'JEARKIV' has only to be specified, if 'JEARKIV' must be randomized.

*INSERT OBRAND.10

(0)

14 FORMAT ()

Format for 'OBARKIV' has only to be specified, if 'OBARKIV' must be randomized.

*INSERT KBRAND.10

(0)

18 FORMAT ()

Format for 'KBARKIV' has only to be randomized, if 'KBARKIV' must be randomized.

The user finally has to select the subroutine which integrates the desired archives, i.e. those files which have been randomized. The user selects the appropriate subroutine by using the corresponding INSERT IXXX.yy text line:

*INSERT IJE.yy selects subroutine 'UNIA' for compilation,
'UNIA' writes 'JEARKIV' on 'IGARKIV'.

*INSERT IOB.yy selects subroutine 'UNIB' for compilation.
'UNIB' writes 'OBARKIV' on 'IGARKIV'.

*INSERT IKB.yy selects subroutine 'UNIC' for compilation.
'UNIC' writes 'KBARKIV' on 'IGARKIV'.

*INSERT IJEObKB.yy selects subroutine 'UNIABC' for compilation.
'UNIABC' writes 'JEARKIV', 'OBARKIV' and 'KBARKIV' on 'IGARKIV'.

*INSERT IJEOb.yy selects subroutine 'UNIAB' for compilation.
'UNIAB' writes 'JEARKIV' and 'OBARKIV' on 'IGARKIV'.

*INSERT IJEKB.yy selects subroutine 'UNIAC' for compilation.
'UNIAC' writes 'JEARKIV' and 'KBARKIV' on 'IGARKIV'.

*INSERT IOBKB.yy selects subroutine 'UNIBC' for compilation.
'UNIBC' writes 'OBARKIV' and 'KBARKIV' on 'IGARKIV'.

*INSERT IXXX.10 , where IXXX represents one of the above-mentioned subroutine
Decks.

9 FORMAT ()

(0)

14 FORMAT ()

(0)

18 FORMAT ()

(0)

9 FORMAT () represents the format for 'JEARKIV' and must be specified if IXXX equals IJE, IJEObKB, IJEOb or IJEKB.

14 FORMAT () represents the format for 'OBARKIV' and must be specified if IXXX equals IOB, IJEObKB, IJEOb or IOBKB.

18 FORMAT () represents the format for 'KBARKIV' and must be specified if IXXX equals IKB, IJEObKB, IJEKB or IOBKB.

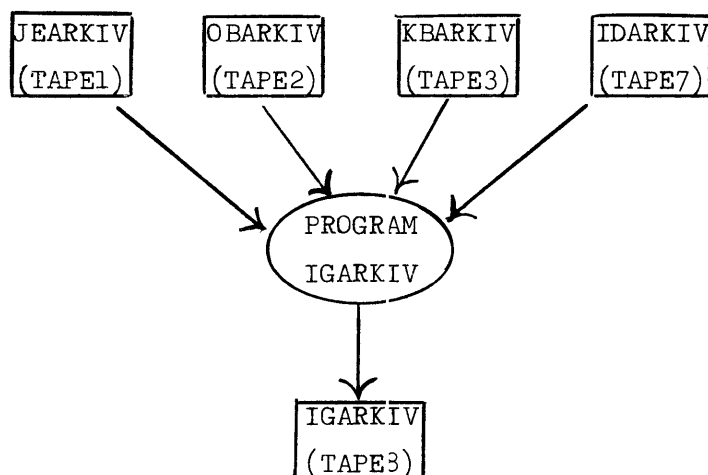
*INSERT IXXX.27

IF () 29,20

(0)

If the logical expression inside the brackets is true, a case from 'IDARKIV' is selected, if not, the case is rejected, i.e. not written on 'IGARKIV'. If the text line is omitted, all cases from 'IDARKIV' are written upon 'IGARKIV' and as well all cases from the randomized archive files.

Flow Chart of Program IGARKIV



SCOPE 3.3. Control Card Deck creating Standard Version of 'IG-ARCHIVE'.

XXXXX,CM100000,T3000,NT1. (specification of user's job name)
 ACCOUNT,X,Y,Z. (specification of user's account card)
 LIMIT,1000.
 RFL,100.
 RPACK,XXX,E,XXX. (specification of user's RPACK)
 REQUEST,IGARKIV,PK,XXX. (specification of user's RPACK)
 RFL,2000.
 ATTACH,RANDOM,RECAURANDOMIO,CY=2.
 RFL,35000.
 UPDATE,P=PGARKIV,D.
 RFL,47000.
 FTN,I=COMPILE.
 RFL,100000.
 LOAD,RANDOM.
 REDUCE.
 LGO,JEARKIV,OBARKIV,KBARKIV,IDARKIV,IGARKIV.
 RFL,10000.
 COPYSBF,TAPE9.

7/8/9

*IDENT KONTROL

*INSERT IGARKIV.8

INTEGER A(320),AA(320),B(128),BB(128),C(832),CC(832)

*INSERT IGARKIV.9

INTEGER REGISTA(2,1500),REGISTB(2,1120),REGISTC(2,1655)

*INSERT IGARKIV.10

IJEQBKB=1

*INSERT IGARKIV.11

PRUSA=5

PRUSB=2

PRUSC=13

*INSERT IGARKIV.12

IA=269

IB=110

IC=796

*INSERT IGARKIV.13

NA=1500

NB=1120

NC=1655

*INSERT JERAND.10

9 FORMAT(2I7,2X,3R10,3X,11I7,14(/,18I7),/I7)

*INSERT OBRAND.10

14 FORMAT(6(18I7,/),2I7)

*INSERT KBRAND.10

18 FORMAT(44(18I7,/),4I7)

*INSERT IJEQBKB.10

9 FORMAT(2I7,2X,3R10,3X,11I7,14(/,18I7),/I7)

14 FORMAT(6(18I7,/),2I7)

18 FORMAT(44(18I7,/),4I7)

6/7/8/9

(It is assumed, that PGARKIV,JEARKIV, OBARKIV, KBARKIV and IDARKIV are stored on the RPACK)

5. The Creation of the 'G6--ARCHIVE'.

Program Feature

On page 73 to 77 we have given a detailed description of how to create the 'G6--ARCHIVE'. In this section we will recall the stepwise file manipulations and shortly consider the main programs AGGREG1 and AGGREG2.

- 1) The 'IDARKIV' is sorted according to variable 013 in ascending order using the standard program SORT/MERGE¹⁾. By this procedure the communes completely or primarily included in a given 1966 commune are grouped together.
- 2) Using the main program 'AGGREG1' from 'PGARKIV' a number of 1966 communes are aggregated according to the user determined per cent population deviance.
- 3) Owing to a number of identification alterations of variable 013 accomplished under point 2, the revised 'IDARKIV' is sorted once more according to variable 013 in ascending order.
- 4) Using the sorted revised version of 'IDARKIV' created under point 3, a version of 'IGARKIV' is created setting IDARKIV = 1. (See p. D 19). It means that the first and the second line from the sorted revised version of the 'IDARKIV' are printed on every case on 'IGARKIV'.
- 5) Using variable 013 from the sorted revised 'IDARKIV' on 'IGARKIV' as aggregational variable, all cases on the 'IGARKIV' having identical values on variable 013 are aggregated using the main program 'AGGREG2' from 'PGARKIV'.

The stepwise file manipulations can be accomplished by a number of separate jobs, saving the necessary files for the following job, or by a single job performing a number of independent program executions. The last solution will be presented here.

It appears from above, that we during the file manipulations are using three mainprograms:

a) Mainprogram 'AGGREG1'.

This program performs the rather complicated task to alter identifications and names on the 'IDARKIV' according to the user determined per cent population deviance (See p. 76). Furthermore the program can either work upon the 1966 commune structure, i.e. variable 013 to 027, or upon the 1970 commune structure, i.e. variable 028 to 042.

In order to reduce the computer time considerably, it is required, that the user sorts the 'IDARKIV' according to either variable 013 or variable 028 before running the program.

The program performs its identification and name alterations in two steps.

1) CDC manual nr. 60252600E, SORT/MERGE Reference Manual 6000 version 3.

First the program controls, using information in variables 019 and 024 resp. variables 034 and 039, to what extent a given number of old communes primarily allocated to a 1966 resp. 1970 commune, result in an excess of per cent population units from the real 1966 resp. 1970 commune. If the deviation exceeds the user determined population deviance, one or more adjacent 1966 resp. 1970 communes are aggregated by giving them identification and name of the greatest 1966 resp. 1970 commune.

Next, in a similar way as above, the program controls, using information in variables 020 and 025 resp. 035 and 040, to what extent a given number of old communes primarily allocated to a 1966 resp. 1970 commune, result in a deficit of per cent population units from the real 1966 resp. 1970 commune. If the deviation exceeds the user determined population deviance, one or more adjacent 1966 resp. 1970 communes are aggregated by giving them identification and name of the greatest 1966 resp. 1970 commune.

Finally a special version of the 'IDARKIV' is written for the creation of the 'INARKIV'.

This file contains for every line first number and name of the original commune in the 'NKARKIV' extracted from 'IDARKIV', see p. 80, followed by the eventually altered commune number and name during program execution. If no alteration numbers and names will be identical:

Variable number	Variable name	Type	Source
001	original 1970 commune identification	I	IDARKIV
002	1970 commune name	R	IDARKIV
003	1970 commune name continued	R	IDARKIV
004	1970 commune name continued	R	IDARKIV
005	eventually altered 1970 commune identification	I	Program
006	eventually altered 1970 commune name	R	Program
007	eventually altered 1970 commune name continued	R	Program
008	eventually altered 1970 commune name	R	Program

Fortran Format: (I4,1X,3R10,1X,I4,3R10)

Number of cases: 278 (if all 1697 cases on IDARKIV is selected. See p. E6.)

Sorting order: Ascending order on variable 001.

b) Mainprogram 'IGARKIV'.

See pp. D 14 to D 22.

c) Mainprogram 'AGGREG2'

This program performs a unit aggregation of selected variables from 'IGARKIV' resp. 'INARKIV', using variable 013 on the second line resp. variable 004 on the first line as aggregational variable. The Program specifically handles the two first lines on 'IGARKIV' resp. the first line on 'INARKIV' and writes preliminary the first line on every case in 'G6ARKIV' resp. G7ARKIV' resp. 'N7ARKIV'. The first line has the following format:

Variable number	Variable name	Type	Source
001	1966 Commune identification on OBARKIV or 1970 Commune identification on NKARKIV	I	VAR013 or VAR028 on IDARKIV
002	1966 Commune name on OBARKIV or 1970 Commune name on NKARKIV	R	VAR014 or VAR029 on IDARKIV
003	1966 Commune name on OBARKIV or 1970 Commune name on NKARKIV, continued	R	VAR015 or VAR030 on IDARKIV
004	1966 Commune name on OBARKIV or 1970 Commune name on NKARKIV	R	VAR016 or VAR031 on IDARKIV
005	Number of unit aggregations on IGARKIV or INARKIV	I	Program

Fortran Format (I7,2X,3R10,3X,I7).

Update Decks

- *AGGREG1: Contains mainprogram 'AGGREG1' and subroutines 'IDENTIS', 'DIFFERE', 'SORT', 'IDFIND1', 'IDFORAN', 'UDSKRIV', 'IDFIND2', 'FIL3', and 'SLUT'.
- *IGARKIV: Contains mainprogram 'IGARKIV' and subroutines 'SQG' and 'UDSKRIV'.
- *JERAND: Contains subroutine 'ARAND'.
- *OBRAND: Contains subroutine 'BRAND'.
- *KBRAND: Contains subroutine 'CRAND'.
- *IJEOKKB: Contains subroutine 'UNIABC'.
- *AGGREG2: Contains mainprogram 'AGGREG2'.

Files

(Files are listed in the following order on the program Cards)

a) Files in program 'AGGREG1':

TAPE1: Reads the sorted 'IDARKIV'

TAPE2: Writes the revised 'IDARKIV' for program 'IGARKIV'

TAPE3: Writes the revised 'IDARKIV' for program 'INARKIV'

TAPE4: Scratch file

OUTPUT: Writes normal print output from program execution

b) Files in program 'IGARKIV':

(See p. D 16)

c) Files in program 'AGGREG2':

TAPE1: Reads 'IGARKIV'

TAPE2: Writes aggregated version of 'IGARKIV'

TAPE3: Writes a control transcript of TAPE2

OUTPUT: Writes normal print output from program execution

Update Control Cards

Since the use of several mainprograms in the same job requires a corresponding number of UPDATE run, the UPDATE control cards must be divided into a similar number of logical records, i.e. separated by 7/8/9 cards.

1. Update Control Card Deck for Mainprogram 'AGGREG1':

IDENT XXX (0)¹⁾

XXX is any name identifying the following text line insertions

INSERT AGGREG1.14 (0)

PROCENT=X

X = Per cent Population Deviance. An integer ranging from 0 to 100. If not specified X has a default value of 5.

INSERT AGGREG1.15 (0)

OBARKIV=1

If specified the program will control the 1966 commune structure, i.e. variables 013 to 027 on 'IDARKIV'. If not specified, the program will control the 1970 commune structure, i.e. variables 028 to 042 on 'IDARKIV'.

INSERT AGGREG1.16 (0)

OPSTIL=1

1) 0 = Optional. R = Required.

If not specified variables on 'IDARKIV' 019 and 020, if variable 021 equals 2, and variables 034 and 035, if variable 036 equals 2, will be set to zero. This text line should be inserted for longitudinal files not including variables from 1962 and later.

*INSERT AGGREG1.17

(O)

IF() 4,5

If specified, the user can select a subset of cases from 'IDARKIV'. If not specified all cases from 'IDARKIV' will be entered in the program execution. If the logical expression inside the brackets is true the case in question will be selected. The logical expression must refer to a value or value combinations of one or more variables in 'IDARKIV'. The variables are defined as an integer array: VAR(42). If e.g. the logical expression is defined as (VAR(8).LE.21.AND.VAR(9).GE.21) only those commune units will be selected having information about 1921 local selection.

*COMPILE AGGREG1

(O) (R)

If the program is used completely in its standard version, i.e. without *IDENT and one or more *INSERT text lines, it is required to insert this text line.

2. Update Control Card Deck for Mainprogram 'IGARKIV'.

See pp. D 17-21.

3. Update Control Card Deck or Mainprogram 'AGGREG2'.

*IDENT XXX

(O)

XXX is any name identifying the following text line insertions.

*INSERT AGGREG2.6

(R)

INTEGER VARA(X), VARB(X)

X = number of variables to be aggregated. Since the program automatically handles the first and second line on 'IGARKIV', and the first line on 'INARKIV' only the variables in the original archive files to be aggregated should be defined here.

*INSERT AGGREG2.7

(R)

VARANT=X

X has been defined in previous text line insertion

*INSERT AGGREG2.8

(R)

2 FORMAT()

Format for variables on 'IGARKIV' or 'INARKIV' (TAPE1) to be unit aggregated and written upon 'G6ARKIV', 'G7ARKIV' or 'N7ARKIV'. The number of variable

format specifications must equal X in former text line insertion. Notice, that the two first lines on every case on 'IGARKIV' and the first line on 'INARKIV' are automatically read by the program.

•INSERT AGGREG2.9

(R)

10 FORMAT()

Format for aggregated variables written upon 'G6ARKIV', G7ARKIV' or 'N7ARKIV' (TAPE2). The number of variable format specifications must equal X. Notice, that the first line on every case on 'G6ARKIV', G7ARKIV' or 'N7ARKIV' is automatically written by the program. See p. D 25.

•INSERT AGGREG2.10

INARKIV=1

If specified, the program must read from 'INARKIV' (TAPE1). If not specified, the program must read from 'IGARKIV'.

•INSERT AGGREG2.12

(O)

KONTROL=0

If specified, no control transcript of 'G6ARKIV' will be written upon TAPE3.

•INSERT AGGREG2.13

(O)

REWIND=0

If specified, TAPE1, if KONTROL is set to zero, and TAPE2 will not be rewound.

SORT/MERGE Control CARDS

In the following we will demonstrate the use of CDC Standard Program SORT/MERGE called as a main program. The Program is called with the following control card:

SORTMRG, I=NN, O=MM.

The parameters are optional. I reads SORT/MERGE directives, O writes print output from program. Default values: I=INPUT, O=OUTPUT.

The program requires about 50.000 octal words. A smaller field length will result in a lower efficiency during program execution.

The control card deck from which the program reads a number of user specified directives is demonstrated in a standard version. For more special features, the user must consult the manual:

SORT(P₁, P₂, P₃)

P₁ = Number of sort input files. The program can sort and merge up to 32 input files.

P_2 = Number of key fields. The parameter indicates the number of character fields in the record determining the sequence of records in the output file. 1 to 100 key fields can be specified.

P_3 = Number of characters in the record. Similar to P_2 on RECORD control card. See below.

FILE(P_1 ,S,B,, P_2 ,N)

P_1 = logical file name on input file to be sorted

P_2 = R file will be rewound and closed. = C File will be closed but not rewound.

FILE(P_1 ,O,B,, P_2 ,N)

P_1 = logical file name on sorted output file. The file must have been opened before the SORTMRG control card using a REWIND,lfn., a REQUEST,lfn,*PF. or a REQUEST,lfn,PK,XXX. control card.

P_2 = Similar to P_2 in the former control card.

KEY(P_1 ,C, P_2 , P_3)

P_1 = A ascending sorting order. = D descending sorting order.

P_2 = the position of the first character in the key field relative to the first character in the record, counting from one.

P_3 = number of characters in the key field.

A maximum of 100 KEY control cards can be inserted. The file will be sorted first according to the first KEY control card and then according to the second KEY control card, etc. The standard ascending sequence for integer characters are: blank, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

RECORD(I, P_1 , P_2)

P_1 = U The record consists of one text line. = F The record consists of more than one text line.

P_2 = Number of characters in the record. If P_1 = U then P_2 = max. number of characters in the text line. If P_1 = F, then P_2 = number of characters in the first line plus a number of characters so that the total number of characters equal the nearest higher multiple of ten. If the number of added characters are only zero or one, ten more characters must be added. If e.g. the number of characters are 72, the number must be raised to 80. If e.g. the number of characters are 79, the number must be raised to 90. An empty line contains 10 characters. The same procedure is then applied to the following text line and finally the total number of characters in each line are summed. The user must be very carefully in counting the total number of characters in an F record. If not the sorting procedure will give totally misleading results.

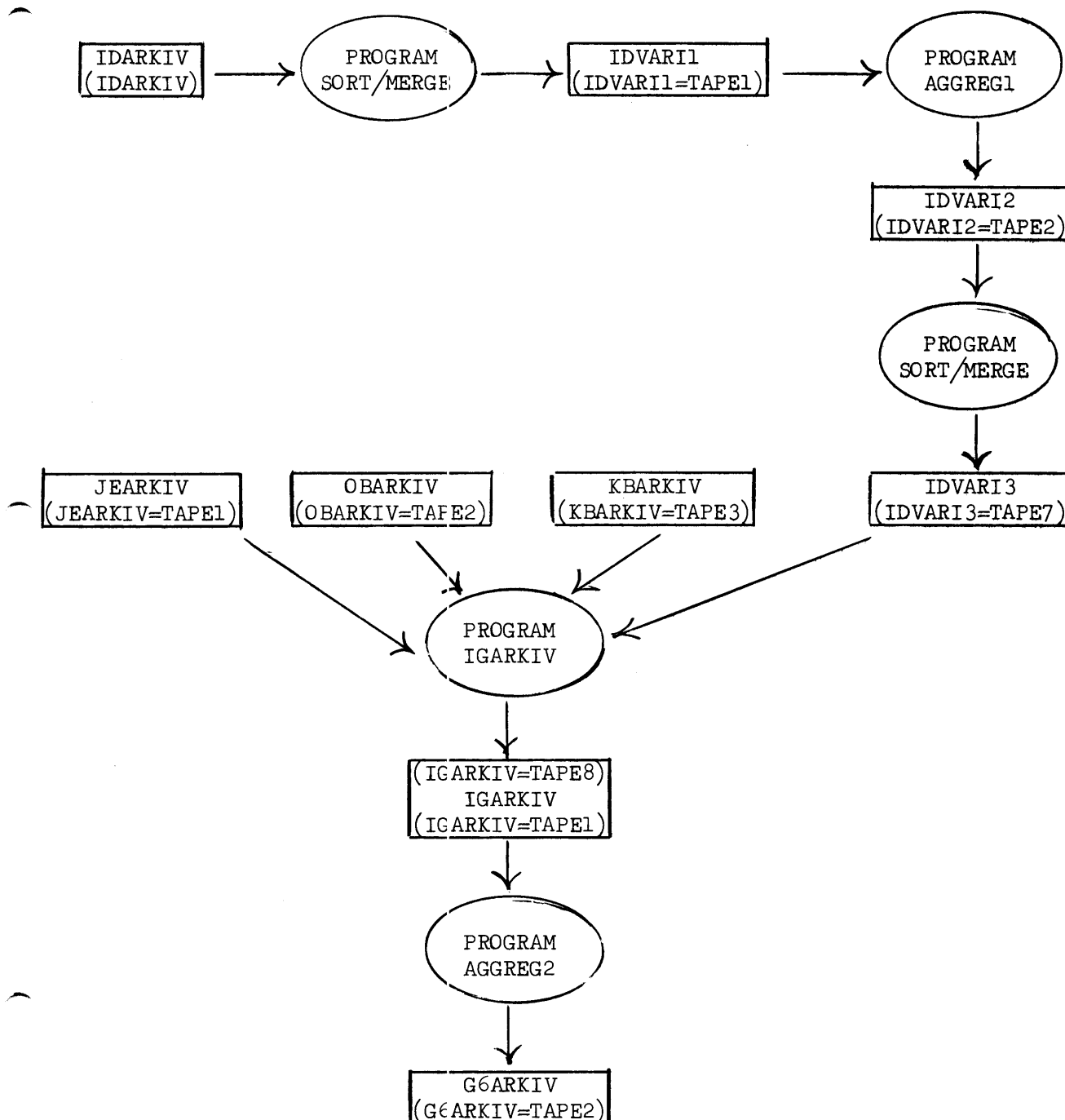
END

This card must terminate a SORT/MERGE control card deck

Sequence of Logical Records for the Creation of 'G6ARKIV'.

1. SCOPE Control Cards
2. SORTMRG. IDARKIV to IDVARI1
3. Program AGGREG1. IDVARI1 to IDVARI2
4. SORTMRG. IDVARI2 to IDVARI3
5. Program IGARKIV. JEARKIV, OBARKIV, KBARKIV and IDVARI3 to IGARKIV
6. Program AGGREG2. IGARKIV to G6ARKIV

A Flow Chart of the Creation of the 'G6-ARCHIVE'.



SCOPE 3.3. Control Card Deck creating Standard Version of the 'G6ARKIV'.

```

XXXXX,CM100000,T4500,NT1. (SPECIFICATION OF USER'S JOB NAME)
ACCOUNT,X,Y,Z. (SPECIFICATION OF USER'S ACCOUNT CARD)
BIGTXT. GENERATION OF "G6ARKIV"
RFL,100.
LIMIT,1000.
RPACK,XXX,E,XXX. (SPECIFICATION OF USER'S RPACK)
COMMENT. CREATION OF IDVARI1,IDVARI2 AND IDVARI3
RFL,1000.
REWIND,IDVARI1.
RFL,50000.
SORTMRG. SORTING OF IDARKIV TO IDVARI1
RFL,35000.
UPDATE,P=PGARKIV,D. AGGREG1 TO COMPILE FILE
RFL,47000.
FTN,I=COMPILE,R=0.
RFL,77000.
REDUCE.
LGO,IDVARI1,IDVARI2. CREATION OF IDVARI2
RFL,1000.
RETURN,LGO,IDVARI1,TAPE3,TAPE4.
REWIND,IDVARI3.
RFL,50000.
SORTMRG. SORTING OF IDVARI2 TO IDVARI3
RFL,1000.
RETURN,IDVARI2.
RFL,100.
COMMENT. CREATION OF IGARKIV
REQUEST,IGARKIV,PK,XXX. (SPECIFICATION OF USER'S RPACK)
RFL,2000.
ATTACH,RANDOM,RECAURANDOMIO,CY=2.
RFL,35000.
UPDATE,P=PGARKIV,D. IGARKIV TO COMPILE FILE
RFL,47000.
FTN,I=COMPILE,R=0.
RFL,100000.
LOAD,RANDOM.
REDUCE.
LGO,JEARKIV,OBARKIV,KBARKIV,IDVARI3,IGARKIV. CREATION OF IGARKIV
RFL,10000.
COPYSBF,TAPE9.
RFL,1000.
RETURN,LGO,TAPE4,TAPES,TAPE6,TAPE9,IDVARI3.
RFL,100.
COMMENT. AGGREGATION OF IGARKIV TO G6ARKIV
REQUEST,G6ARKIV,PK,XXX. (SPECIFICATION OF USER'S RPACK)
RFL,35000.
UPDATE,P=PGARKIV,D.AGGREG2 TO COMPILE FILE
RFL,47000.
FTN,I=COMPILE,R=0.
REDUCE.
LGO,IGARKIV,G6ARKIV. CREATION OF G6ARKIV
RFL,10000.
COPYSBF,TAPE3.
REMOVE,IGARKIV.
7/8/9
SORT(1,1,250)
FILE(IDARKIV,S,B,,R,N)
FILE(IDVARI1,O,B,,R,N)
KEY(A,C,81,4)
RECORD(I,F,250)
END
7/8/9

```

```
*IDENT KONTROL
*INSERT AGGREG1.15
      OBARKIV=1
```

7/8/9

```
SORT(1,1,250)
FILE(IDVARI2,S,B,,R,N)
FILE(IDVARI3,O,B,,R,N)
KEY(A,C,81,4)
RECORD(I,F,250)
END
```

7/8/9

```
*IDENT KONTROL
*INSERT IGARKIV.8
      INTEGER A(320),AA(320),B(128),BB(128),C(832),CC(832)
*INSERT IGARKIV.9
      INTEGER REGISTA(2,1500),REGISTB(2,1120),REGISTC(2,1655)
*INSERT IGARKIV.10
      IJE0BKB=1
*INSERT IGARKIV.11
      PRUSA=5
      PRUS3=2
      PRUSC=13
```

```
*INSERT IGARKIV.12
      IA=269
      IB=110
      IC=796
```

```
*INSERT IGARKIV.13
      NA=1500
      NB=1120
      NC=1655
```

```
*INSERT IGARKIV.14
      IDARKIV=1
```

```
*INSERT JERAND.10
      9 FORMAT(2I7,2X,3H10,3X,11I7,14(/,18I7),/,I7)
```

```
*INSERT OBRAND.10
      14 FORMAT(6(18I7,/),2I7)
```

```
*INSERT KBRAND.10
      18 FORMAT(44(18I7,/),4I7)
```

```
*INSERT IJE0BKB.10
      9 FORMAT(2I7,2X,3H10,3X,11I7,14(/,18I7),/,I7)
      14 FORMAT(6(18I7,/),2I7)
      18 FORMAT(44(18I7,/),4I7)
```

7/8/9

```
*IDENT KONTROL
*INSERT AGGREG2.6
      INTEGER VARA(520),VARB(520)
*INSERT AGGREG2.7
      VARANT=520
*INSERT AGGREG2.8
```

```
      2 FORMAT(70X,8I7,14(/,18I7),/,I7,/,7X,17I7,/,5(18I7,/),2I7,/,
      A7X,10I7,49X,/,126X,/,126X,/,10I7,56X,/,126X,/,119X,I7,/,
      89I7,63X,/,126X,/,112X,2I7,/,8I7,70X,/,126X,/,105X,3I7,/,
      C7I7,77X,/,126X,/,98X,4I7,/,6I7,84X,/,126X,/,91X,5I7,/,
      D5I7,91X,/,126X,/,84X,6I7,/,4I7,98X,/,126X,/,77X,7I7,/,
      E3I7,105X,/,126X,/,70X,8I7,/,2I7,112X,/,126X,/,63X,9I7,/,
      F1I7,119X,/,126X,/,56X,10I7,/,126X,/,126X,/,49X,10I7,7X,/,
      G126X,/,126X,/,42X,10I7,14X,/,126X,/,126X,/,35X,10I7,21X,/,
      H126X,/,126X,/,28X)
```

```
*INSERT AGGREG2.9
      10 FORMAT(28(18I7,/),16I7)
```

6/7/8/9

(It is assumed, that PGARKIV, JEARKIV, OBARKIV, KBARKIV and IDARKIV are stored on the RPACK). NOTICE: DO NEVER RETURN A RPACK FILE. ALL INFORMATION ON THE RPACK DISC CAN BE LOST)

D 6 The Creation of the 'GN-ARCHIVE'.

Program Feature

The creation of 'GNARKIV' is a rather complicated task involving a range of program routines. The user, who is interested in a simultaneous use of data from the communes before and after the 1970 Amalgamation Reform is recommended first to read the purely verbal description on page 88 to 91 and then read the technical description of how to create 'G6ARKIV' on page D 23 to D 32, because the 'GNARKIV' is only an extension of the procedures involved in the creation of 'G6ARKIV'.

As an introduction we will recall the stepwise file manipulation and shortly consider the two mainprograms 'INARKIV' and 'GNARKIV' which have not been discussed until now:

- 1) The 'IDARKIV' is sorted according to variable 028 in ascending order using the standard program SCRT/MERGE. See pp. D 28. to D 30. By this procedure the old communes completely or primarily included in a given 1970 commune are grouped together. (IDARKIV to IDVARI1)
- 2) Using the mainprogram 'AGGREG1' a number of 1970 communes are aggregated, i.e. given the same identification and name, according to the user determined per cent population deviance. Default value 5. Two versions of 'IDVARI1' are created, one for 'IGARKIV' (IDVARI2) and one for 'INARKIV' (IDVARI4). (IDVARI1 to IDVARI2 and IDVARI4)
- 3) Owing to a number of identification alterations of variable 028 accomplished under point 2, the 'IDVARI2' is sorted according to variable 028 in ascending order. (IDVARI2 to IDVARI3)
- 4) Using 'IDVARI3' as identification file, a version of 'IGARKIV' is created setting IDARKIV = 2. See p. D 19. It means that the first and the third line from 'IDVARI3' are printed on every case on 'IGARKIV'.
- 5) Using the mainprogram 'AGGREG2', cases on 'IGARKIV' having identical values on variable 013, corresponding to variable 028 on 'IDVARI3' are aggregated (IGARKIV to G7ARKIV).
- 6) Owing to a number of identification alterations of variable 028 accomplished under point 2, the 'IDVARI4', see p. D 24, is sorted according to variable number 005 in ascending order. (IDVARI4 to IDVARI5)
- 7) Using 'IDVARI5' as identification file and data from 'NKARKIV' a version of 'INARKIV' is created. See below.

- 8) Using the mainprogram 'AGGREG2', setting INARKIV = 1, cases on 'INARKIV' having identical values on variable number 005 from 'IDVARI5' are aggregated. (INARKIV to N7ARKIV)
- 9) Using the mainprogram 'GNARKIV', see below the file 'G7ARKIV' and 'N7ARKIV' are merged into a single archive file. (G7ARKIV and N7ARKIV to GNARKIV)

It appears from above, that we during the file manipulations are using 6 mainprograms:

- a) Mainprogram SORT/MERGE. A SCOPE standard program. See p. D 28. to D 29.
- b) Mainprogram 'AGGREG1'. See pp. D 23.to D 24. and D 26. to D 27.
- c) Mainprogram 'IGARKIV'. See pp. D 14 to D 22.
- d) Mainprogram 'AGGREG2'. See pp. D 25 and D 27 to D 28.
- e) Mainprogram 'INARKIV'.

It is a program in structure identical to 'IGARKIV'. It merges the archive 'NKARKIV' with the special edition of 'IDARKIV'. i.e. 'IDVARI5'.

- f) Mainprogram 'GNARKIV'. This program merges the 'G7ARKIV' and the 'N7ARKIV', files both having the same number of cases and the same order of commune identification numbers. The program automatically reads the first line of each case on 'G7ARKIV' and 'N7ARKIV' and writes the first line of each case on archive file 'GNARKIV':

Variable number	Variable name	Type	Source
001	1970 Commune Identification on 'G7ARKIV' and 'N7ARKIV'	I	VAR001 on 'G7ARKIV'
002	1970 Commune name on 'G7ARKIV' and 'N7ARKIV'	R	VAR002 on 'G7ARKIV'
003	1970 Commune name. Continued	R	VAR003 on 'G7ARKIV'
004	1970 Commune name. Continued.	R	VAR004 on 'G7ARKIV'
005	Number of aggregated commune cases from 'IGARKIV'	I	VAR005 on 'G7ARKIV'
006	Number of aggregated commune cases from 'INARKIV'	I	VAR005 on 'N7ARKIV'

Format (I7,2X,3R10,3X,2I7)

The first line is then followed by the aggregated variables from 'G7ARKIV' on the following lines and finally the aggregated variables from 'N7ARKIV'.

Update Decks

*AGGREG1: Contains mainprogram 'AGGREG1' and subroutines 'IDENTIS', 'DIFFERE', 'SORT', 'IDFIND1', 'IDFORAN', 'UDSKRIV', 'IDFIND2', 'FIL3', and 'SLUT'.
 *IGARKIV: Contains mainprogram 'IGARKIV' and subroutines 'SQG' and 'UDSKRIV'.
 *JERAND: Contains subroutine 'ARAND'
 *OBRAND: Contains subroutine 'BRAND'
 *KBRAND: Contains subroutine 'CRAND'
 *IJEOKKB: Contains subroutine 'UNIABC'
 *AGGREG2: Contains mainprogram 'AGGREG2'
 *INARKIV: Contains mainprogram 'INARKIV' and subroutines 'SQG' and 'UDSKRIV'.
 *NKRAND: Contains subroutine 'DRAND'
 *INK: Contains subroutine 'UNID'
 *GNARKIV: Contains mainprogram 'GNARKIV'

Files

(Files are listed in the following order on the Program Cards)

- a) files in program 'AGGREG1':
 (see p. D 26)
- b) files in program 'IGARKIV'
 (see p. D 16.)
- c) files in program 'AGGREG2':
 (see p. D 26.)
- d) files in program 'INARKIV':
 TAPE1: Reads 'NKARKIV'
 TAPE2: Reads special version of 'IDARKIV', i.e. 'IDVARI5'.
 TAPE3: Writes 'INARKIV'
 TAPE4: Writes control transcript of 'INARKIV'
 TAPE5: scratch file
 OUTPUT: Writes normal print output from program execution
- e) files in program 'GNARKIV':
 TAPE1: Reads 'G7ARKIV'
 TAPE2: Reads 'N7ARKIV'
 TAPE3: Writes 'GNARKIV'
 TAPE4: Writes control transcript of 'GNARKIV'
 OUTPUT: Writes normal print output from program execution

Update and SORTMRG Control Cards

Since the use of several mainprograms in the same job requires a corresponding number of UPDATE runs plus a number of SORTMRG runs, the UPDATE and the SORTMRG control cards must be divided into a similar number of logical records, i.e. separated by 7/8/9 cards. In this section the use of different control card deck will be demonstrated:

1. SORTMRG Control Card Deck:

See pp. D 28. to D 30.

2. Update Control Card Deck for program 'AGGREG1':

See pp. D 26. to D 27.

3. Update Control Card Deck for program 'IGARKIV':

See pp. D 17. to D 21.

4. Update Control Card Deck for program 'AGGREG2':

See pp. D 27. to D 28.

5. Update Control Card Deck for program 'INARKIV':

IDENT XXX (O)

XXX is any name identifying the following text line insertions

INSERT INARKIV.4 (R)

INTEGER D(X)

X = nearest higher multiple of 64 from the number of variables to be read from a case on 'INARKIV'. See D 17.

INSERT INARKIV.5 (R)

INTEGER REGSTD(2,Y)

Y = number of cases in 'INARKIV'. See p. D 17. to D 18.

INSERT INARKIV.6 (R)

PRUSD=Z

Z = X/64. See p. D 18.

INSERT INARKIV.7 (R)

ID=XX

XX = number of variables to be read from a case on 'INARKIV' and finally written upon 'INARKIV'. See p. D 18. to D 19.

INSERT INARKIV.8 (R)

ND=Y

Y = number of cases on 'INARKIV'. See p. D 19.

INSERT INARKIV.10 (O)

KONTROL=0

If specified no control transcript of TAPE3 on TAPE4 will be written and TAPE3 will not be rewound.

*INSERT INARKIV.11 (O)

REWIND=0

If specified TAPE1 will not be rewound

*INSERT NKRAND.6 (R)

9 FORMAT()

Format on TAPE1, i.e. 'NKARKIV'

*INSERT INK.6 (R)

9 FORMAT()

Format for second to the last line on TAPE3, i.e. 'INARKIV'. The first line is automatically written by the program as a line from TAPE2. See p. D 24.

*INSERT INK.12 (O)

IF() 29,20

If the logical expression inside the brackets is true, a case from TAPE2 is written upon TAPE3 followed by the corresponding case from TAPE1. If not, no case is written upon TAPE3. The variables to be specified in the logical expression is an array VAR(8), corresponding to the 8 variables on TAPE2. See p. D 24. If the text line is not specified, all cases from TAPE2 will be written on TAPE3.

6. Update Control Card Deck for Program 'GNARKIV'.

*IDENT XXX (O)

XXX is any name identifying the following text line insertions

*INSERT GNARKIV.4

INTEGER VARA(X),VARB(Y)

X = number of variables on TAPE1, i.e. 'G7ARKIV' apart from first line

Y = number of variables on TAPE2, i.e. 'N7ARKIV' apart from first line

*INSERT GNARKIV.5 (R)

1 FORMAT()

Format on TAPE1, i.e. 'G7ARKIV' from second to last line on the case

*INSERT GNARKIV.6 (R)

2 FORMAT()

Format on TAPE2, i.e. 'N7ARKIV' from second to last line on the case

*INSERT GNARKIV.8 (O)

KONTROL=0

If specified no control transcript of TAPE3 will be written on TAPE4 and TAPE3 will not be rewound.

INSERT GNARKIV.9

(0)

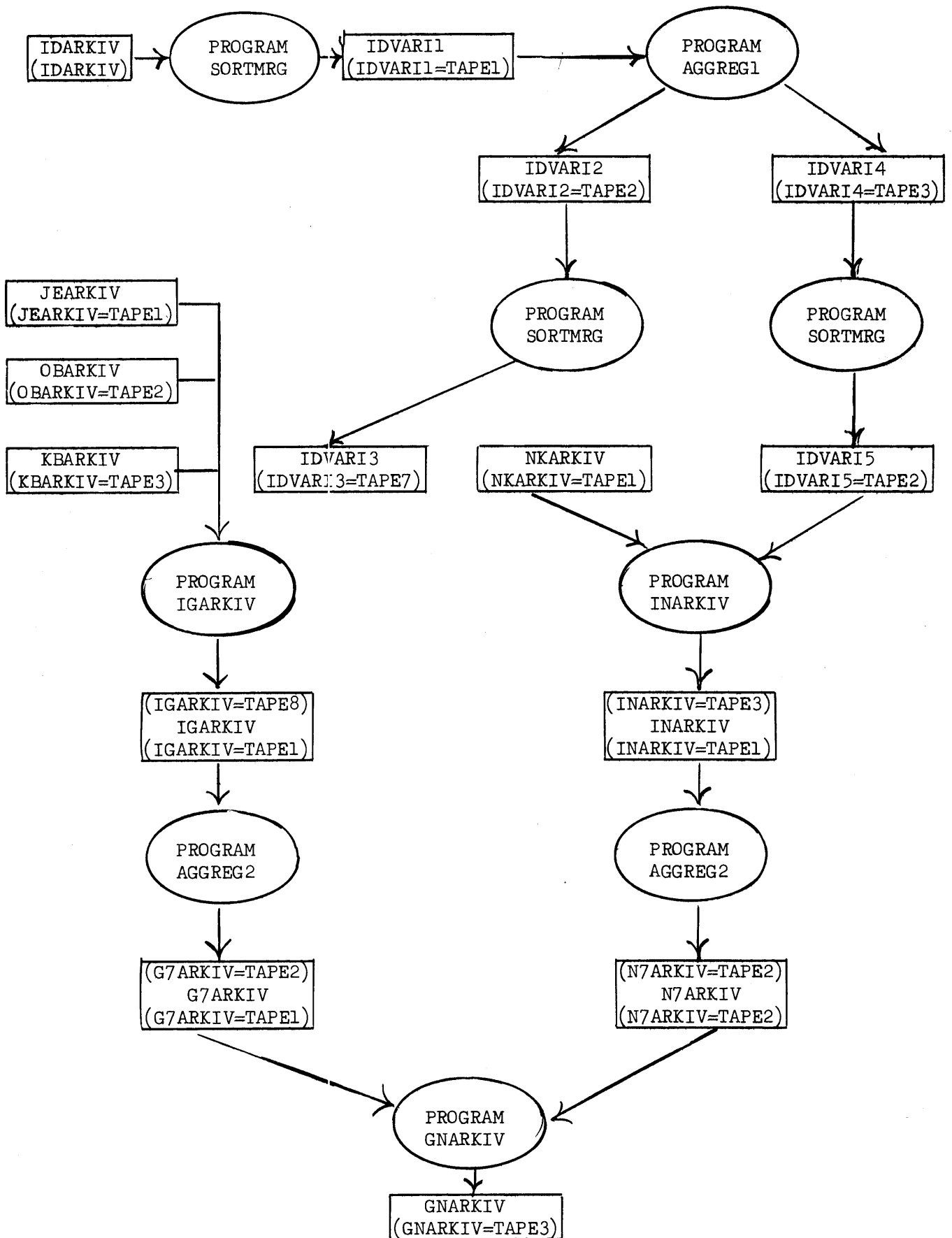
REWIND=0

If specified TAPE1 and TAPE2 will not be rewound

Sequence of logical records for the creation of 'GNARKIV'

1. SCOPE Control Cards
2. SORTMRG. IDARKIV to IDVARI1
3. Program AGGREG1. IDVARI1 to IDVARI2 and IDVARI4
4. SORTMRG. IDVARI2 to IDVARI3
5. Program IGARKIV. JEARKIV, OBARKIV, KBARKIV and IDVARI3 to IGARKIV
6. Program AGGREG2. IGARKIV to G7ARKIV
7. SORTMRG. IDVARI4 to IDVARI5
8. Program INARKIV. NKARKIV and IDVARI5 to INARKIV
9. Program AGGREG2. INARKIV to N7ARKIV
10. Program GNARKIV. G7ARKIV and N7ARKIV to GNARKIV

Flow Chart of the Creation of the 'GN-ARCHIVE'.



SCOPE 3.3. Control Card Deck creating Standard Version of the 'GNARKIV'.

```

XXXXX,CM100000,T5000,NT1. (SPECIFICATION OF USER'S JOBNAME)
ACCOUNT,X,Y,Z. (SPECIFICATION OF USER'S ACCOUNT CARD)
BIGTXT. GENERATION OF "GNARKIV"
RFL,100.
LIMIT,1000.
RPACK,XXX,E,XXX. (SPECIFICATION OF USER'S RPACK)
COMMENT. CREATION OF IDVARI1, IDVARI2, IDVARI3 AND IDVARI4
RFL,1000.
REWIND,IDVARI1.
RFL,50000.
SORTMRG. SORTING OF IDARKIV TO IDVARI1
RFL,35000.
UPDATE,P=PGARKIV,D. AGGREG1 TO COMPILE FILE
RFL,47000.
FTN,I=COMPILE,R=0.
RFL,77000.
REDUCE.
LGO,IDVARI1,IDVARI2,IDVARI4. CREATION OF IDVARI2 AND IDVARI4
RFL,1000.
RETURN,LGO,IDVARI1,TAPE4.
REWIND,IDVARI3.
RFL,50000.
SORTMRG. SORTING OF IDVARI2 TO IDVARI3
RFL,1000.
RETURN,IDVARI2.
RFL,100.
COMMENT. CREATION OF IGARKIV
RFL,2000.
REQUEST,IGARKIV,PK,XXX. (SPECIFICATION OF USER'S RPACK)
ATTACH,RANDOM,RECAURANDOMIO,CY=2.
RFL,35000.
UPDATE,P=PGARKIV,D. IGARKIV TO COMPILE FILE
RFL,47000.
FTN,I=COMPILE,R=0.
RFL,100000.
LOAD,RANDOM.
REDUCE.
LGO,JEARKIV,OBARKIV,KBARKIV,IDVARI3,IGARKIV. CREATION OF IGARKIV
RFL,10000.
COPYSBF,TAPE9.
RFL,1000.
RETURN,LGO,TAPE4,TAPE5,TAPE6,TAPE9,IDVARI3.
RFL,100.
COMMENT. AGGREGATION OF IGARKIV TO G7ARKIV
RFL,35000.
UPDATE,P=PGARKIV,D.AGGREG2 TO COMPILE FILE
RFL,47000.
FTN,I=COMPILE,R=0.
REDUCE.
LGO,IGARKIV,G7ARKIV. CREATION OG G7ARKIV
RFL,10000.
COPYSBF,TAPE3.
REMOVE,IGARKIV.
RETURN,LGO,TAPE3.

```

```

RFL,1000.
REWIND,IDVAR15.
RFL,50000.
SORTMRG. SORTING OF IDVAR14 TO IDVAR15
RFL,1000.
RETURN,IDVAR14.
RFL,100.
COMMENT. CREATION OF INARKIV
RFL,2000.
REWIND,RANDOM.
RFL,35000.
UPDATE,P=PGARKIV,D. INARKIV TO COMPILE FILE
RFL,47000.
FTN,I=COMPILE,R=0.
RFL,100000.
LOAD,RANDOM.
REDUCE.
LGO,NKARKIV,IDVAR15,INARKIV. CREATION OF INARKIV
RFL,10000.
COPYSBF,TAPE4.
RFL,1000.
RETURN,LGO,IDVAR15,TAPE4,TAPE5.
RFL,100.
COMMENT. AGGREGATION OF INARKIV TO N7ARKIV
RFL,35000.
UPDATE,P=PGARKIV,D. AGGREG2 TO COMPILE FILE
RFL,47000.
FTN,I=COMPILE,R=0.
REDUCE.
LGO,INARKIV,N7ARKIV. CREATION OF N7ARKIV
RFL,10000.
COPYSBF,TAPE3.
RETURN,LGO,INARKIV,TAPE3.
RFL,100.
COMMENT. CREATION OF GNARKIV
REQUEST,GNARKIV,PK,XXX. (SPECIFICATION OF USER'S RPACK)
RFL,35000.
UPDATE,P=PGARKIV,D. GNARKIV TO COMPILE FILE
RFL,47000.
FTN,I=COMPILE,R=0.
REDUCE.
LGO,G7ARKIV,N7ARKIV,GNARKIV. CREATION OF GNARKIV
RFL,10000.
COPYSBF,TAPE4.
7/8/9
SORT(1,1,250)
FILE(IDARKIV,S,B,,R,N)
FILE(IDVAR11,0,B,,R,N)
KEY(A,C,161,4)
RECORD(I,F,250)
END
7/8/9
*IDENT KONTROL
*COMPILE AGGREG1
7/8/9
SORT(1,1,250)
FILE(IDVAR12,S,B,,R,N)
FILE(IDVAR13,0,B,,R,N)
KEY(A,C,161,4)
RECORD(I,F,250)
END
7/8/9

```



```

*IDENT KONTROL
*INSERT IGARKIV.8
    INTEGER A(320),AA(320),B(128),BB(128),C(832),CC(832)
*INSERT IGARKIV.9
    INTEGER REGISTA(2,1500),REGISTB(2,1120),REGISTC(2,1655)
*INSERT IGARKIV.10
    IJE09KB=1
*INSERT IGARKIV.11
    PRUSA=5
    PRUSB=2
    PRUSC=13
*INSERT IGARKIV.12
    IA=259
    IB=110
    IC=796
*INSERT IGARKIV.13
    NA=1500
    NB=1120
    NC=1655
*INSERT IGARKIV.14
    IDARKIV=2
*INSERT JERAND.10
    9 FORMAT(2I7,2X,3R10,3X,11I7,14(/,18I7),/,I7)
*INSERT OBRAND.10
    14 FORMAT(6(18I7,/),2I7)
*INSERT KBRAND.10
    18 FORMAT(44(18I7,/),4I7)
*INSERT IJE09KB.10
    9 FORMAT(2I7,2X,3R10,3X,11I7,14(/,18I7),/,I7)
    14 FORMAT(6(18I7,/),2I7)
    18 FORMAT(44(18I7,/),4I7)
7/8/9
*IDENT KONTROL
*INSERT AGGREG2.6
    INTEGER VARA(520),VARB(520)
*INSERT AGGREG2.7
    VARAVT=520
*INSERT AGGREG2.8
    2 FORMAT(70X,8I7,14(/,18I7),/,I7,/,7X,17I7,/,5(18I7,/),2I7,/,
    A7X,10I7,49X,/,126X,/,126X,/,10I7,56X,/,126X,/,119X,I7,/,
    B9I7,63X,/,126X,/,112X,2I7,/,8I7,70X,/,126X,/,105X,3I7,/,
    C7I7,77X,/,126X,/,98X,4I7,/,6I7,84X,/,126X,/,91X,5I7,/,
    D5I7,91X,/,126X,/,84X,6I7,/,4I7,98X,/,126X,/,77X,7I7,/,
    E3I7,105X,/,126X,/,70X,8I7,/,2I7,112X,/,126X,/,63X,9I7,/,
    F1I7,119X,/,126X,/,56X,10I7,/,126X,/,126X,/,49X,10I7,7X,/,
    G126X,/,126X,/,42X,10I7,14X,/,126X,/,126X,/,35X,10I7,21X,/,
    H126X,/,126X,/,28X)
*INSERT AGGREG2.9
    10 FORMAT(28(18I7,/),16I7)
7/8/9
SORT(1,1,71)
FILE(IDVARI4,S,B,,R,N)
FILE(IDVARI5,0,B,,R,N)
KEY(A,C,37,4)
RECORD(I,U,71)
END
7/8/9

```

```

*IDENT KONTROL
*INSERT INARKIV.4
    INTEGER D(128)
*INSERT INARKIV.5
    INTEGER REGISTD(2,278)
*INSERT INARKIV.6
    PRUSD=2
*INSERT INARKIV.7
    ID=126
*INSERT INARKIV.8
    ND=278
*INSERT NKRAND.6
    9 FORMAT(6(18I7,/),18I7)
*INSERT INK.6
    9 FORMAT(6(18I7,/),18I7)
7/8/9
*IDENT KONTROL
*INSERT AGGREG2.6
    INTEGER VARA(123),VARB(123)
*INSERT AGGREG2.7
    VARANT=123
*INSERT AGGREG2.8
    2 FORMAT(21X,15I7,6(/,18I7))
*INSERT AGGREG2.9
    10 FORMAT(6(18I7,/),15I7)
*INSERT AGGREG2.10
    INARKIV=1
7/8/9
*IDENT KONTROL
*INSERT GNARKIV.4
    INTEGER VARA(520),VARB(123)
*INSERT GNARKIV.5
    1 FORMAT(28(18I7,/),16I7)
*INSERT GNARKIV.6
    2 FORMAT(6(18I7,/),15I7)
6/7/8/9

```

(It is assumed, that PGARKIV, JEARKIV, OBARKIV, KBARKIV, NKARKIV and IDARKIV are stored on the RPACK)

NOTICE: DO NEVER RETURN A FILE STORED ON A RPACK DISC. ALL FILES ON THE RPACK CAN BE LOST.