EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

PS/CO/Note 85-4
18.4.1985

THE INFO FILE SYSTEM

J. Cupérus

Geneva, Switzerland

1. INTRODUCTION

The INFO file system is a program written in NORD-PASCAL, working with the SINTRAN III operating system on the NORD-100 computers. With this program, the user can create, update and search tables (composed of records with fixed-format fields) and associated text files for free-format text. A previous version of this program has been in use for almost 5 years as support for our real-time data-bases for working sets, OB names and data for TREES, ALARMS and SETUP.

The system is very easy to use. The user must know enough PED commands to enter and edit data or text but, apart from this, he is completely guided by the system with multiple-choice questions to which he answers with the first letter of his choice. In addition, he can type '?' to get more information.

Little will be told here about the detailed interactions with INFO (which are best learned with hands-on experience) except for some notes in the next paragraph. In the following paragraphs, the different operating modes of INFO are described.

2. SOME NOTES ON THE USE OF INFO

- As for PED, you must specify the terminal type number after logging in.
- Data may be in upper or lower case but commands must be in uppercase.
- Call INFO by typing (INFO)TABLE.
- To multiple-choice questions, answer with the first letter of your choice (+ return). '?' gives help and 'Q' or 'QUIT' gets you out of the current section.
- INFO will ask for missing parameters, if any. If you know the answers to several questions in advance, you can type them on one line, separated by commas.
- The editor is compatible with PED.
- Multi-line update is terminated by typing CTRL-L (CONTROL-L).
- You can get out of many questions without action by typing CTRL-L.
- After escape, the SINTRAN III command CONTINUE will bring you back in INFO at more or less the right spot.
- A textfield is limited to 21 lines of 80 characters.

3. CREATE

In this mode, the user specified a NAME and INFO creates a file NAME:TABL for the fixed-format information. If he desires to store text, then a file NAME:TEXT is also created. Both these files can be contiguous (when created with n pages) or indexed (when created with 0 pages).

None of these files may exist before their creation by INFO and deletion is not possible in INFO, for safety reasons. After creation, you pass automatically in the MODIFYHEAD mode.

4. MODIFYHEAD

In this mode, the fields of the table are specified or modified. When the table contains already records of data, the allowed modifications are rather trivial in order not to make the data unreadable, but then another mode, REWRITE, can help.

Up to 50 fields can be specified. Fields can be added, changed, deleted, exchanged and so forth. For each field, the user must specify, if appropriate:

TYPE this specifies the nature of the data and determines also how it will be coded. Possible types are :

STRING : (coded 2 characters per 16 bit word);

INTEGER, BINARY, OCTAL, HEXADECIMAL : (1 word);

DATE : in format YY/MM/DD (packed in 1 word);

CODEWORDS : a list of words coded into a list of integers with the help of a file of codewords (only one such file per table). A list of length 1 is perfectly acceptable).

Types 1,2,3 or 4 : lists of positive integers which are packed in one word according to specifications supplied by the user, e.g. for packing a CAMAC C/N/A address in one word;

TEXT : in fact a pointer to the associated text file.

CHARACTERS-IN-FIELD : the maximum number of characters in a field of type STRING. In general, the place reserved for the field in printouts or displays on the screen. The size is limited to 110 characters.

STATISTICS : possible modes are : NONE, SUM, MEAN, BOTH. These statistics are printed after a data printout.

NUMBER-OF-DECIMALS : number of decimals to be printed out for reals or statistics.

KEYFIELD : if the field is not of type TEXT, it can be labelled keyfield 1....6. In mode ORDER, the records are ordered according to the information in the keyfields, with keyfield 1 having the highest priority. If no keyfield is specified, then the records stay in the original sequence.

TITLE : maximum 16 characters. Is used for prompting user input and for column heading in printouts. If CHARACTERS-IN-FIELD < 16 then only part of TITLE is printed as column header.

If types 1...4 are used, the user must specify for each of these types the number of components and, for each component :

- the number of bits leftshift (decreasing for each component),
- the maximum value of the integer,
- the number of characters in a printout (CHARS=3 may print .../019/...).

Some general information is also requested :

- table description of maximum 1 line,
- name of associated codeword file, if any.

The textfields must come after all other fields. Keyfield labelling must be without gaps, starting from 1. The user can leave the MODIFY-HEAD mode only when the heading is correctly specified. This heading is stored as the first page (of 1024 words) in the file NAME:TABL. The following pages will contain lines of data, entered by the user in the UPDATE mode.

4.1 Codeword file

A codeword file is an INFO table with just one field of type STRING. The order of the string records never changes because the number of the record is the code number for the string (no keyfield !). An example of a codeword file is (DATA-BASE)PROPERTY-LIST:TABL with the EM properties.

5. UPDATE

In this mode, the user can append, change and delete records. Usually, the records up to a certain number are ordered according to the keyfields. New records are appended at the end and can later be merged with the already ordered records in mode ORDER. Some simple commands allow the user to find and list records.

The keyfields of a new record must be filled before it can be appended. The filling of other fields is optional. The fields to be updated can be selected with the SELECT command. To change the keyfield of an ordered line (REWRITE), it is deleted (leaving a blank line until the next ordering) and rewritten at the end of the file. Unfilled fields are either blank (for string) or -32767 (for integers or lists) or -9.999 E-99 (for reals) or 0 (for all other types).

Input is checked against type and field size and wrong data must be corrected before they are accepted. When the record is filled, it is automatically written to disc. The prompting message reminds the user what to fill in.

It is also possible to append or change whole blocks of data :

JOIN : selected blocks of records from a compatible INFO table can be appended. Is useful to combining two tables into one and also for recovering lost or deleted records from an archive table.

BLOCKINPUT : a flat file with compatible structure can be appended. This can be used for converting existing data structures into INFO format. By means of mode REWRITE, the data can then be converted to another representation.

BLOCKCHANGE : for a given field and between lower and upper record limits, substitute one character string for another (works with all data types).

6. ORDER

After an ORDER operation, all the records in the table are ordered according to the keyfields. This operation is quite efficient for merging a number of records at the end of the table with the bulk of already ordered records but is rather slow if all the records must be reordered according to a new set of keyfields. A temporary file NAME:SORT (max. 10 pages long) is written in the user space to help recovery (with a new ORDER) if the system crashes during an ORDER operation.

7. REWRITE

In this mode, the user can modify the heading by deleting, adding and changing the field specifications. The program checks whether the new heading is compatible with the old one (e.g. changing an INTEGER into a STRING is possible but changing a DATE into HEX makes no sense) and then rewrites the whole file, under a new name, according to the new specifications. Conversion errors, due to wrong data, are listed. The old file is left unchanged.

Conversion is done by expressing the data in string form and re-converting the strings according to the new heading. This makes it impossible to convert INTEGER, BINARY, OCTAL and HEX into each other but these are exactly the conversions which are possible in the MODIFYHEAD mode (because the code does not change by this type change !).

8. SEARCH

This is one of the most used modes. The main operations are :

LIST : list a sequence of records or display one record in a more elaborate form, including text, if any.

FIND : list the records which contain in a specified field : a specified string in any position or an abbreviation, in SINTRAN style. If field 0 is specified, a tree-structured search is entered with as many levels as there are keyfields; this works only if there are not more than 20 possible branchings at each mode, e.g. not more than 20 diffierent entries in keyfield 1, not more than 400 in keyfield 2 and so on.

DUMP : write the data to a file NAME:DATA (or NAME:TABL in case of a SUBFILE dump) in a form useable by other programs.

Four modes are possible :

- BLOCKDUMP dumps the records, one after the other, without heading and without spaces between the records. Record zero is special and contains, in all positions, the number of data records.

- APPENDBLOCKDUMP works as BLOCKDUMP but the new records are appended to previous dumps (with compatible record structure !). The record zero contains now the new total number of data records.

- DICTIONARY-DUMP prepares files for fast access, by RT programs, according to the keyfields, which must be in front of the record in proper order. More about this dump in next paragraph.

- SUBFILE dumps in normal INFO file format. The name of this file must start with SUB- for safety reasons. It can be renamed afterwards if you like.

PRINT : print the data in one of 2 formats :

- one or two lines per record according to available space.

- a new line for each field.

SELECT : select the fieldset for display, printout or dump.

NEXTFILE : close the files NAME : xxxx and open a new set of files.

CRITERIA: up to 6 search criteria can be specified, each involving a particular field. The criteria are linked with AND/OR, with AND taking precedence. A search can be initiated, which writes all records which satisfy the criteria on the SUBTABLE. All previous operations are possible on the SUBTABLE in addition to NEWSEQ and ORDER which are possible on the SUBTABLE only.

NEWSEQ : select fields of the SUBTABLE in any order and reformat the table.

ORDER : the SUBTABLE can be ordered according to any sequence of fields. This order operation is more efficient than that which keeps the main table ordered because it must not safeguard the data in case of a computer crash.

Searches are done in a serial way : the whole file is scanned page by page. The main overhead in this procedure is getting the pages from disc, one by one. TEXT is in a separate file which is not searched.

9. DICTIONARY FILES

Real-time programs must have fast access to well-defined records. They do this through ICCI routines which can read (on the TREES computer) the special files prepared by the DICTIONARY DUMP (see preceding paragraph).

The dictionary dump starts with 1 or more index pages (of 1024 words), followed by data pages with records of information. If a record does not fit completely at the end of the page, the space is left blank and the record

begins on the next page. The first positions of the first index page contain:

Pos. 0 : number of records
Pos. 1 : words per record
Pos. 2 : records per page
Pos. 3 : start of last record on page
Pos. 4 : last page in file (starting from 0)
Pos. 5 : length of key in words
Pos. 6 : number of index pages
Pos. 7 : position of start of last key on index
Pos. 8, 9 : 0 (spare positions).

Starting at Pos. 10 follows a contiguous list of the keys of the last line of each data page.

The key search is correct only for integer and string keys but these are the only keys which matter in practice (a record is not normally identified with a real or octal key !).

10. MODE-FILE APPLICATION PROGRAMS

INFO is normally an interactive system which prompts the user but it can also be driven by mode file. Before writing the mode file, go through the whole sequence interactively and note your answers. In the answers to multiple-choice questions, only the first letter is significant but the rest is accepted as a comment. Answers may be written on one line, separated by commas. Some questions require a carriage return only (to indicate : no change). This requires as many empty lines in the mode file as there are carriage returns. Appendix 1 gives an example of a mode file application program.

11. PASCAL APPLICATION PROGRAMS

The INFO files can be read by background NORD-PASCAL application programs (AP). The AP is compiled in 2-Bank mode with a perform macro :

```
PERFORM,(INFO)DBPROG,DBPROG,....
```

The AP must call 2 include files :

- (DATABASE)PAS-DB-RECORDS with data base record structures. Make sure that the tables you are interested in are described in this file.

- (INFO)PAS-DB-INTERFACE with data base access routines (see Appendix 2).

The AP has about 55 K-words of global + stack space left. You can now open several tables. For each table you must declare a buffer (for index, codewords and working space) which takes 8.5 K of space. This means that the AP can have a maximum of 6 tables open at the same time and less if the AP needs much data space of its own. Appendix 3 shows an example of an AP doing a trivial job just to get an idea of how it works.

If you need to have access to a table through a secondary index then you must do some preparations (by mode file) before entering the AP :

- call INFO and SEARCH the table with a criterium which is always satisfied,

- ORDER the resulting subfile according to the secondary index,

- DUMP the subfile; this is the table you need, it has the same record description as the original table but with different keys.

After execution of the AP, delete the subfile.

12. OUTLOOK FOR THE FUTURE

A home-made system has the advantage that it can be tailored to fit the applications. In the long run, however, it can be no match for the powerful new data base systems which are commercially developed. NORSK DATA has for the moment only a network data-base SIBAS (and IR which is a document retrieval system). SIBAS is attractive where efficient access to large data sets (more than 10000 entries per table) is necessary and when these data have a stable structure as is the case in many commercial applications. In our case, the data sets are small to medium and we can afford the occasional inefficiencies of a relational data base. Our continuously changing and expanding data structure is also much better captured by a relational structure than by a network or hierarchical structure.

When and if NORSK DATA implements a modern relational data base such as ORACLE (and provided it has a decent PASCAL interface), we shall probably abandon INFO in favour of it. Our tables are at present compatible with the relational model and we should keep it that way for future extensions. Conversion of our data to the relational data base should be fairly straightforward. Our mode-file APs will have to be rewritten (perhaps as PASCAL programs but probably as mode files) in the data base interface language (probably some variant of SQL). Our PASCAL APs will have to be modified for the new interface but this is often a minor part of these programs.

In the meantime we can live very well with INFO for several more years and new applications should use it preferably so as not to fragment our data in incompatible sets. IR should only be used when its superior text capabilities clearly outweigh the disadvantages of its inflexible and closed structure. Burying of data in NODAL programs should also be avoided as much as possible.

Distribution
Operation Group
Controls Group

/ww

APPENDIX 1: Example of a Mode-file Data-base Application Program

```
@ENTER DATA-BASE,XXXXX,,25
@CC (DATA-BASE)BAT-DB-RTDBASE  J.CUPERUS   LAST MODIFIED: 17 APR 1985
@CC CALL MODFILES AND PROGRAMS TO CONSTRUCT THE RT DATA BASE FILES.
@(INFO)TABLE
ORDER, (DATA-BASE)EQM-DATA-BASE
QUIT
@TIME
@CC ********************************************************************************
@CC DUMP DICTIONARIES
@CC ********************************************************************************
@DELETE-FILE (DATA-BASE)SETUP-INFO:DATA
@CREATE-FILE (DATA-BASE)SETUP-INFO:DATA,,,
@(INFO)TABLE
ORDER, (DATA-BASE)SETUP-INFO
SEARCH, DUMP, DICTIONARY, (DATA-BASE)SETUP-INFO, QUIT
@CC
@DELETE-FILE (DATA-BASE)ALARM-MES-DICT:DATA
@CREATE-FILE (DATA-BASE)ALARM-MES-DICT:DATA,,,
@(INFO)TABLE
ORDER, (DATA-BASE)AL-MES-DICT
SEARCH, DUMP, DICTIONARY, (DATA-BASE)AL-MES-DICT, QUIT
@TIME
@CC ********************************************************************************
@CC CONSTRUCT OB DICTIONARY
@CC ********************************************************************************
@DELETE-FILE (DATA-BASE)OB-DICT-FILE:DATA
@CREATE-FILE (DATA-BASE)OB-DICT-FILE:DATA,,,
@(INFO)TABLE
SEARCH, (DATA-BASE)EQM-DATA-BASE
CRITERIA, INCLUDED FOR FIELD, 1, COMPARE, 0, GREATER, STOP, SEARCH
NEWSEQ, 1/12/3/4/5
ORDER, 1
DUMP, DICTIONARY, (DATA-BASE)OB-DICT-FILE:DATA
QUIT, QUIT, QUIT
@TIME
@CC ********************************************************************************
@CC START PROGRAM DB-EQINFO
@CC ********************************************************************************
@(DATA-BASE)DB-EQINFO
@TIME
@CC ********************************************************************************
@CC START PROGRAM DB-WSET
@CC ********************************************************************************
@DELETE-FILE (PUBLIC)WSPRINT:SYMB
@CREATE-FILE (PUBLIC)WSPRINT:SYMB,,,
@DELETE-FILE (PUBLIC)WSORT1:DATA
@DELETE-FILE (PUBLIC)WSORT2:DATA
@DELETE-FILE (PUBLIC)WSORT3:DATA
@CREATE-FILE (PUBLIC)WSORT1:DATA,,,
@CREATE-FILE (PUBLIC)WSORT2:DATA,,,
@CREATE-FILE (PUBLIC)WSORT3:DATA,,,
@COPY-FILE REF-WS-LIST:DATA  WS-LIST:DATA
@(DATA-BASE)DB-WSET
@DELETE-FILE (PUBLIC)WSORT1:DATA
@DELETE-FILE (PUBLIC)WSORT2:DATA
@DELETE-FILE (PUBLIC)WSORT3:DATA
```

APPENDIX 2: Data-base Interface Procedures for PASCAL Application Programs
==

EXTRACT FROM FILE (INFO)PAS-DB-INTERFACE:SYMB

```
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

INFO INTERFACE ROUTINES FOR USE BY PASCAL APPLICATION PROGRAMS

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@


THESE ROUTINES CAN BE USED WITHOUT KNOWING THE FUNCTIONING OF THE INFO PROGRAM.
THE TYPES USED ARE:

ISET=SET OF 0..127
INSTANT=RECORD TICK,SEC,MIN,HOUR,DAY,MONTH,YEAR: INTEGER END
QSTRING=PACKED ARRAY[1..64] OF CHAR
FILEBUF=PRIVATE TYPE, ABOUT 8.5 KWORDS LONG, FOR DATA-BASE OPERATIONS ON A FILE.
DBREC=VARIANT TYPE FOR ONE DATA-BASE RECORD (DEFINED IN FILE PAS-DB-RECORDS).

VARIABLE NOW:INSTANT IS DEFINED AND CAN BE USED BY THE BACKGROUND PROGRAM.  *)


PROCEDURE OPENINFO(VAR FBUF:FILEBUF; FILNAME:QSTRING; OPENMODE:INTEGER;
                   VAR NROFRECORDS:INTEGER);
(***************************************************************************

FILNAME MUST BE WITHOUT TYPE.  OPENMODE SHOULD NORMALLY BE 1 (READ-ONLY).
OPEN THE FILE AND READ THE FILE HEADING INTO FBUF.
INITIALISE THE CODEWORDS IF ANY.  FILL THE PAGE SEARCH INDEX.
PRINT SOME INFORMATION ABOUT THE FILE.  RETURN THE NUMBER OF RECORDS FOUND IN
THE FILE.  IF AN ERROR IS DETECTED, PRINT MESSAGE AND STOP PROGRAM.  *)


PROCEDURE CLOSINFO(VAR FBUF:FILEBUF);
(*************************************

CLOSE THE FILE CONNECTED TO FBUF.  *)


PROCEDURE GETRECORD(VAR FBUF:FILEBUF; RECNO:INTEGER; VAR REC:DBREC;
                    VAR STATUS: INTEGER);
(*********************************************************************

RETURN THE RECORD RECNO.  STATUS=1 IF RECNO OUTSIDE RANGE ELSE STATUS=0 *)


PROCEDURE FINDRECORD(VAR FBUF:FILEBUF; VAR REC:DBREC; VAR RECNO,STATUS:INTEGER);
(*******************************************************************************

KEYFIELDS OF REC MUST BE FILLED IN BEFORE ENTERING.  OTHER FIELDS ARE IGNORED.
SEARCH THE FILE FOR A RECORD WHICH MATCHES KEY.
IF FOUND, RETURN COMPLETE RECORD AND STATUS=0.
IF NOT FOUND:
     RETURN RECORD WITH NEXT HIGHER KEY AND STATUS=1 OR, IF THERE IS NO SUCH
     RECORD, RETURN LAST RECORD IN FILE AND STATUS=2.
IN EACH CASE, RETURN RECNO OF THE RECORD.     *)
```

```
PROCEDURE WRITEFIELD(VAR FBUF:FILEBUF; VAR REC:DBREC; FIELDNO:INTEGER;
                     VAR OUT:TEXT);
(*************************************************************************

PRINT THE FIELD FIELDNO OF REC.  THE FIELDLENGTH IS DETERMINED BY NROFCHAR IN
THE FILE HEADER IN FBUF. *)


PROCEDURE DISPLAYRECORD(VAR FBUF:FILEBUF;  RECNO:INTEGER; FIELDSET:ISET;
                        VAR OUT:TEXT);
(*************************************************************************

PRINT THE CONTENTS OF EACH FIELD IN FIELDSET, PRECEDED BY TILE, ON A NEW LINE.*)


PROCEDURE PRINTRECORD(VAR FBUF:FILEBUF;  RECNO,LINELENGTH:INTEGER;
                      FIELDSET:ISET;  VAR OUT:TEXT);
(*************************************************************************

PRINT THE FIELDS IN FIELDSET ON ONE LINE.  FIELD 0 IS THE RECORD NUMBER.
LINELENGTH IS THE AVAILABLE SPACE (80 CHAR FOR A4 AND MAX 132 CHAR FOR LINE
PRINTER.  SOME FIELDS MAY BE PRINTED ON 2 LINES TO MAKE THE LINE FIT IN THE
AVAILABLE SPACE.  RECNO=0 PRINTS THE TITLES FOLLOWED BY A BLANK LINE. *)


PROCEDURE GETDATE(VAR NOW:INSTANT);
(**********************************

RETURN PRESENT TIME AND DATE IN RECORD NOW. *)


PROCEDURE WRITEDATE(NOW:INSTANT; VAR OUT:TEXT);
(**********************************************

WRITE DATE AND TIME TO FILE OUT IN FORMAT:  '1985/FEB/05  15:05:22 '  *)
```

APPENDIX 3: Example of a PASCAL Data-base Application Program

```
(*$B2*)
(* (DATA-BASE)PAS-DB-APPLIC:SYMB;
J.CUPERUS;       STARTED APR 1985;       LAST UPDATED: 16 APR 1985;
COMPILE AND DUMP ON (DATA-BASE)DB-APPLIC WITH:
PERFORM,(INFO)DBPROG,DBPROG,DB-APPLIC,DATA-BASE,PASSWORD    *)

(*****************************************************************************
PROGRAM DB-APPLIC
*****************************************************************************)

PROGRAM DBAPPLIC(OUTPUT);
(************************

RATHER TRIVIAL EXAMPLE OF A DATA-BASE APPLICATION PROGRAM.
OPEN TABLE EQ-INFO.  FIND THE FIRST EQUIPMENT FOR FEC=16 AND EM=8.
PRINT THE NAME/FEC/EM/EQ OF ALL EQUIPMENT IN THIS EM. *)

(*$L-*)
$INCLUDE (DATA-BASE)PAS-DB-RECORDS;     (*DEFINES, AMONG OTHERS, RECORD EQM*)
(*$L+*)

VAR  EQMBUF: FILEBUF;                   (*FILE BUFFER FOR EQM-DATA-BASE*)
     XEQM: DBREC;                       (*VARIANT DATA-BASE RECORD*)
     FIELDSET: ISET;                    (*SET OF FIELDS TO BE PRINTED*)
     NROFRECORDS: INTEGER;              (*NUMBER OF RECORDS IN TABLE*)
     RECNO: INTEGER;                    (*RECORD NUMBER*)
     COCO: INTEGER;                     (*COMPLEMENT CODE*)

(*$L-*)
$INCLUDE (INFO)PAS-DB-INTERFACE;        (*DATA-BASE INTERFACE PROCEDURES*)
(*$L+*)


BEGIN  (*MAIN PROGRAM*)
(**********************)

GETDATE(NOW);
WRITE('PROGRAM DB-APPLIC  ');   WRITEDATE(NOW, OUTPUT);
WRITELN;  WRITELN;
OPENINFO(EQMBUF, '(DATA-BASE)EQM-DATA-BASE', 1, NROFRECORDS);
FIELDSET:=[1,3,4,5];
WITH XEQM.EQ DO BEGIN
     FECNO:=16;  EMNO:=8;  EQNO:=1;
     FINDRECORD(EQMBUF, XEQM, RECNO, COCO);
     IF (COCO<=1) AND (FECNO=16) AND (EMNO=8) THEN BEGIN
          WRITELN;
          PRINTRECORD(EQMBUF, 0, 80, FIELDSET, OUTPUT);   (*COLUMN HEADINGS*)
          REPEAT
               PRINTRECORD(EQMBUF, RECNO, 80, FIELDSET, OUTPUT);
               RECNO:=SUCC(RECNO);
               GETRECORD(EQMBUF, RECNO, XEQM, COCO);
          UNTIL (COCO<>0) OR (FECNO<>16) OR (EMNO<>8);
          END
     ELSE WRITELN('NO EQUIPMENT FOUND FOR FECNO=16/EMNO=8');
     END; (*WITH XEQM.EQ*)
CLOSINFO(EQMBUF);
END.
```

APPENDIX 4: INFO-related Files

(INFO)PAS-DB-DECLAR:SYMB	INFO PROGRAM DECLARATIONS
(INFO)PAS-DB-COMMON:SYMB	INFO GENERAL PURPOSE PROCEDURES
(INFO)PAS-DB-SEARCH:SYMB	INFO SEARCH PROCEDURES
(INFO)PAS-DB-UPDATE:SYMB	INFO UPDATE PROCEDURES
(INFO)PAS-DB-MAIN:SYMB	INFO MAIN INTERACTIVE PROGRAM
(INFO)PAS-DB-INTERFACE:SYMB	INFO INTERFACE PROCEDURES FOR APPLICATIONS
(INFO)NPL-DB-LIB:SYMB	INFO NPL LIBRARY PROCEDURES
(DATA-BASE)PAS-DB-RECORDS:SYMB	DATA-BASE RECORD DESCRIPTIONS
(INFO)DBPROG:MCRO	PERFORM MACRO FOR DB PROGRAM COMPILATIONS
(INFO)DB-MAIN:PROG	INFO PROGRAM FOR TESTS
(INFO)TABLE:PROG	INFO INTERACTIVE PROGRAM
(INFO)HELP:TABL	TABLE FILE FOR HELP FACILITY IN INFO
(INFO)HELP:TEXT	CORRESPONDING TEXT FILE

GROUPE OP

ADORNI, V.....	
BACON, E.	G. ADRIAN
BARIBAUD, G.	D. ALLEN
BENINCASA, G.P.	G. AZZONI
BOBBIO, P.	S. BAIRD
BOONROY, P.	L. BLANC
BUNACIU, B.	N. BLAZIANU
BURLA, P.	H. BOBILLIER
CAILLIAU, R.	J. BOILLOT
CASALEGNO, L.	M. BOUTHEON
CHERY, C.	B. CANARD
CLOYE, J.-J.	J.C. CENDRE
CRUTCHER, A.	E. CHERIX
CUISINIER, G.	V. CHOHAN
CUPERUS, J.	P. COLLET
DAEMS, G.	J. COMTE
DANEELS, A.	G. CYVOCT
DEBORDES, R.	C. DANGOISSE
DEHAVAY, C.	M. DAMIANI
DESMARIS, M.F.....	E. DURIEU
DI MAIO, F.	T. ERIKSSON
DORENBOS, T.	B. FRAMMERY
GAGNAIRE, A.	D. GUEUGNON
GAYRAUD, Ch.	L. HENNY
GAYRAUD, D.	F. HOEKEMEIJER
GIUDICI, F.	R. HOH
GUICHARD, D.	G. JUBIN
HEINZE, W.	J. KUCZEROWSKI
HEYMANS, P.	F. LENARDON
JACQUEMET, F.	R. LEY
JANKOWSKI J.	B. L'HUILLIER
KIRK, M.	D. MANGLUNKI
KNAFOU, J.	R. MARTIN
KNOTT, G.	G. MARTINI
KUIPER, B.	M. MARTINI
LARSEN, H.	J.L. MARY
LELAIZANT, M.	S. MAURY
LEWIS, J.	A. NICOUD
LOCK, H.	J. OTTAVIANI
MALANDAIN, E.	E. OVALLE
MARTUCCI, P.	S. PASINELLI
MERARD, L.	M. PERFETTI
de METZ-NOBLAT, N....	J.P. POTIER
NAVRATIL, J.	K. PRIESTNALL
PERRIOLLAT, F.	Y. RENAUD
PERROTT, R.	L. RINOLFI
PHILIPPE, J.	I. ROBINSON
POTDEVIN, P.	G. ROSSET
REDARD, J.	M. RUETTE
REMMER, W.	C. SAULNIER
SCHENKELS, P.	P. SMITH
SERRE, Ch.	Ch. STEINBACH
SHERING, G.	G. TRANQUILLE
SICARD, C.H.	A. VALVINI
SIGAUD, E.	B. VANDORPE
SKAREK, P.	M. VAN ROOIJ
SUPPIN, Ch.	H. VON ARX
TROTTEREAU, E.	H. VESTERGAARD
VIGNES, M.	
VOGT-NILSEN, N.	
VO-DUY, Il	
WILKINSON, W.	