NOTA 1536 H

May 1984

Instituut voor Cultuurtechniek en Waterhuishouding
Wageningen

ASPECTEN van INFORMATIEVERWERKING 48

HANDY'84 utilities
user's guide

collected by

J.B.H.M. van Gils

ISN = 210217-01

ASPECTEN van INFORMATIEVERWERKING
48

C O N T E N T S

INTRODUCTION TO HANDY UTILITIES

ABSTRACT

A set of utilities for VAX/VMS users is collected in the HANDY directory.
The utilities supply a variety of more generally used applications.
(operations, algorithms, instruction input, database, ...)
They are developed by workers of the institute while doing project committed work. These applications are not available in other accessible collections.

UTILITIES

The HANDY directory now contains:

a set of command procedures
one main program
subroutines collected in an object module library

SUPPORTING UTILITIES

HANDY utilities call for supporting utilities. From programming possibilities callable by more HANDY utilities there arose supporting utilities that allow a more general application. But mostly the user doesn't want to be charged of these. The utilities merely intended as supports are marked in the summaries.

INFORMATION

Subroutines may used in a program forming a system set. They are described below as a set. Utilities not forming a set that are intended to be called by user programs have a synopsis in this description.
Every object module in the library is supported by its source program.
To every utility a guide is added. Source programs contain comments that are helpful for reading.
Guides, comments, conversations and messages offered by HANDY are in english language. Command procedures are in DCL language, source programs are in fortran-77.
Only the most recent version of a utility is kept available.

HANDY directory

On STAVAX computer the directory is defined in the user's global symbol table by the system login procedure. In a DCL command the term 'HANDY' is meant to be substituted by the directory name "DRBO:[CGLS.90102913".

FILES in HANDY

Command procedures, source programs, guides and the object library are in separate files. The files have the default filetypes as used in the VAX/VMS operating system.
The file of a guide has the name of the utility enlarged with filetype .TXT

INSTRUCTION INPUT

The utilities can be used automatically receiving their instructions from command level or from instruction files (except one command procedure).
When utilities are also intended to receive instructions from terminal they offer the optional use of conversation vertical scrolling on terminal screen.

SUMMARY OF COMMAND PROCEDURES

Command procedures that are intended to be called by user programs:

call	note	description
@'HANDY'ATTRIBUTE		returns the attributes of a file
@'HANDY'BATCH		accepts commands and submits them to a batch job queue
@'HANDY'EXETIME		running and timing programs
@'HANDY'FOR		compiles a fortran program
@'HANDY'FORLINRUN		compiles, links and runs a program
@'HANDY'ORAUFI		executes ORACLE SQL statements
@'HANDY'ORAODL	(1)	loads bulk data into ORACLE
@'HANDY'PRINT	(2)	prints series of ASCII files via the printer connected to the terminal
@'HANDY'PURGE		purges series of files and renames to the lowest possible version
@'HANDY'TAPECOPY	(3)	reads tapes written in formats without record count fields

(1) The command procedure is not equipped for conversational use.
(2) The call '@'HANDY'PRINT printername' switches to conversational mode.
(3) Only equipped for conversational use.

Command procedures for only supporting other HANDY command procedures:

call	note	description
@'HANDY'CLEAR	(2)	purges files and renames to the lowest possible version
@'HANDY'DELETE	(2)	deletes files with saved names
@'HANDY'DT80LA120	(2)	prints ASCII files via a LA120 DECwriter connected to a DT80/1 terminal
@'HANDY'FLN	(1)	gives the most complete filename
@'HANDY'FLNFLN	(1)	extracts filenames from a string of components
@'HANDY'LA120	(2)	prints ASCII files via a LA120 DECwriter connected to a CIT-101 terminal
@'HANDY'MT140L	(2)	prints ASCII files via a MT140L printer connected to a CIT-101 terminal
@'HANDY'NAME	(1)	enlarges truncated names
@'HANDY'PROCEDURE	(2)	executes a command procedure using filenames

(1) The command procedure is not equipped for conversational use.
(2) The command procedure is equipped for conversational use, but another HANDY command procedure calls for this one, so the same operations are implied in another call.

GENERAL GUIDE TO USE HANDY COMMAND PROCEDURES
****

CALL
There are two kinds of procedure calls:

```
   @'HANDY'procedure_name
   @'HANDY'procedure_name parameter1 parameter2 ...
```

Every parameter value is an instruction to the command procedure.
An empty parameter value is defined by ""; empty instructions are replaced by defaults.

INSTRUCTIONS
Normally the command procedures conversationally ask for instructions when no parameters are given at command level. PRINT and TAPECOPY conversationally ask for needed instructions that are empty at command level. An empty answer is given by RETURN only.

SHORT WRITING INSTRUCTIONS
Sometimes a parameter indicates a name which is in a set of names known by the command procedure. In that case the user may type only the unique starting letters.
Series of filenames may be stated in a combined string using + signs, for example PRINT+TEST.TXT+FOR

HELP TEXTS
Mostly, texts of questions are self explaining.
Sometimes a help text is displayed when the instruction can not be interpreted.
Moreover there is a guide to the command procedure.

AUTOMATIC DELETES
Internally used temporary mediums must be deleted, user versions of files may become superfluous. Some procedures automatically purge and delete files, global symbols and logical names. These actions are not reported by the procedure. They are described in the guides. After an abnormal end of the proces a next call of the command procedure mostly deletes the remaining internally used entities only.

PROGRAM INFORMATION
For every procedure there is a source program file and a guide file in the HANDY directory.
Procedures that are intended to be called by user programs have a synopsis in this description.

SUMMARY OF SUBROUTINES

The instruction subroutine set is a part of a system of reading and reporting program instructions that is introduced together with the set.
The ORACLE subroutine set makes programming the access to the ORACLE base somewhat easier and more surveyable.

subroutine name	description
algorithms	
ALGOINOUT	marks a point as being inside or outside an area
ALGOSORT	sorts an integer*2 array and optionally ranks its mate
instruction subroutine set	
HANDYFLD	opens an unformatted file for direct access input or in-/output
HANDYFLI	opens an instruction file
HANDYFLS	opens a sequential ASCII file
HANDYFOLD	opens an existing file for direct access
HANDYINIT	gives initial values for running with HANDY subroutines
HANDYLOOP	generates a series of integers from each set (from, to, step)
HANDYNUMB	reads a number from instruction input
HANDYROW	reads n-byte values from instruction input
HANDYSTRI	reads and reports a character string
HANDYSTRN	reads a character string from sequential input
HANDYTIME	writes a line with text, date and time
HANDYYORN	asks Yes or No
ORACLE subroutine set	
ORABIND	assigns a program value to an ORACLE SQL substitution variable
ORAEXEC	processes an ORACLE SQL statement
ORAFETCH	returns a row of an ORACLE query result
ORALOGOFF	logs on to ORACLE
ORALOGON	logs off from ORACLE
ORASQL	defines an ORACLE SQL statement
subroutines not intended to be called by user programs	
HANDYALFA	adds a character occurring between apostrophes to a n-byte value
HANDYASK	writes a question enlarged to fixed length
HANDYCSTR	writes a string including some pointer
HANDYDECO	reads a number from a string
HANDYERR	writes a FORTRAN run-time error message
HANDYFLN	reads a filename from instruction input
HANDYPAGE	writes a string and optionally pages output
HANDYSKIP	counts the length of instructions in a line
ORACHECK	checks length and datatype of a program defined buffer area connected to ORACLE
ORACURSOR	returns ORACLE Cursor Area data
ORAERROR	writes an extended ORACLE error message
ORAMVI	returns a Missing Value Indicator in a program defined buffer area connected to ORACLE

ORA... subroutines not intended to be called by user programs are described below under the ORACLE SET header.

REFERENCES TO HANDY SUBROUTINES

All subroutines are collected in object library SUBROUTIN. The user will find the library synopsis in this description and the library guide file in the HANDY directory.
For every subroutine there is a source program file and a guide file in the HANDY directory.
Subroutines forming a system set are described below as a set. Subroutines not forming a set that are intended to be called by user programs have a synopsis in this description.

DISPLAYING THE ATTRIBUTES OF A FILE
synopsis of command procedure 'HANDY'ATTRIBUTE

J.B.H.M. van Gils

ABSTRACT
Every file written in FILES-11 format is equipped with an Attribute Control Block. In the block there are attributes containing the specified properties of a file on which a file might be opened when input or output is done.
The command procedure writes a table with the values of all the attributes of a file to the terminal.

OUTPUT
The output of an attribute table is preceeded and followed by control sequences for a LA-120 DECwriter connected to a CIT-101 terminal. With such a hardware combination the table will be displayed and printed.

INSTRUCTIONS
There are two kinds of procedure calls:
```
 @'HANDY'ATTRIBUTE
 @'HANDY'ATTRIBUTE filename  default_filetype  default_owner_id
```
If all parameter values are empty there is conversationally asked for the filename.

PROGRAM INFORMATION
The command procedure is written in DCL.
The guide is in file 'HANDY'ATTRIBUTE.TXT. The source program is in file 'HANDY'ATTRIBUTE.COM.

VAX/VMS INFORMATION
In fortran you may read the attributes with the INQUIRE statement.
In DCL you may reach the attribute values with the DUMP statement and the F$FILE_ATTRIBUTE lexical function.
You can find the item names in table 5-2 of VAX/VMS Guide to Using Command Procedures.

ACCEPTING COMMANDS AND SUBMITTING THEM TO A BATCH JOB QUEUE
synopsis of command procedure 'HANDY'BATCH

J.B.H.M. van Gils

ABSTRACT
Stated DCL commands are collected in a temporary file which is submitted.
The default batch queue on STAVAX computer is SYS$BATCH.

INSTRUCTIONS
There are two kinds of procedure calls:

```
@'HANDY'BATCH
@'HANDY'BATCH  inputfile  queue_name  job_name  cpu_time_limit  print_delete
```

If all parameters are empty they are conversationally be asked for. An empty parameter value is defined by "", an empty answer is defined by a RETURN.
Empty values are replaced by defaults.
A non-empty value of print_delete sends the log-file to the spool printer (SYS$PRINT) and deletes the log-file after printing.

INPUT
When no inputfile is stated there is prompted for data lines.
Any DCL-command line to be submitted to the queued procedure can be added.
The lines don't need a dollar sign in the first position.
Warning: Enter data only after having used the DECK command.

OUTPUT
The entry number of the queued job is displayed on the terminal. Also the status of the entry after submitting is displayed.
By default the logfile (default name 'BATCH.LOG') made by a batch job is added to the (sub)directory of the user.
Messages on terminal report that the batch job has been completed and printed.

PROBLEMS
The batch job logs in with the users LOGIN so use only the SET TERMINAL command in your LOGIN.COM file under the condition:

```
   IF F$MODE .NES. 'BATCH' THEN SET TERMINAL...
```

Don't use the command: @'HANDY'BATCH ... until this batch job is ready.
Nested batch jobs give unpredictable results.
Reading and writing to SYS$COMMAND in your batch job is not allowed.
When your job is aborted the temporary files will be deleted in the next submitted job by this procedure.

CONTROLLING BATCH JOBS
Some commands to control jobs in the batch job queue:

```
   SET TERM/NOBROADCAST          ! avoids receiving messages
   DELETE/ENTRY=nnn queue_name ! deletes a job from a queue before processing
   STOP/ENTRY=nnn queue_name   ! stops processing of a queued job
   SHOW QUEUE/ALL queue_name
```

PROGRAM INFORMATION
The command procedure is written in DCL.
The guide is in file 'HANDY'BATCH.TXT. The source program is in file
'HANDY'BATCH.COM.

REFERENCES

Gils, J.B.H.M. van, 1983. Aspecten van Informatieverwerking, 39.
Stapelsgewijze verwerking op de Staringcomputer.
ICW-nota 1428: pp.11.

RUNNING AND TIMING PROGRAMS
synopsis of command procedure 'HANDY'EXETIME

W. van Doorne

ABSTRACT
The command procedure enables to execute and time RUN commands.
The cpu time is corrected for the time taken by the command procedure.
The cpu time used by executing a DCL command somewhat depends on the rate of occupation of the computer system, which may be a reason to repeat the timing procedure.

RESULT
By default the running time is displayed.
In conversational mode it serves to display the used cpu time as a total of measured executing times of each program repeatedly executed by the RUN command.
The total cpu time (in hundreds of seconds) of all programs runned is stored in a global symbol. When not in conversational mode only one program is run n times at each call of EXETIME.

INSTRUCTIONS
There are two kinds of procedure calls:

```
@'HANDY'EXETIME
@'HANDY'EXETIME  program_name  number_of_runs  global_symbol  noreport
```

If all parameters are empty there is conversationally asked for the program name and the number of runs. An empty parameter value is defined by "", an empty answer is defined by a RETURN. An empty stated number of runs executes one program run.
The program name is the (abbreviated) filespecification as used in the RUN command.

PROGRAM INFORMATION
The command procedure is written in DCL.
The guide is in file 'HANDY'EXETIME.TXT. The source program is in file 'HANDY'EXETIME.COM.

VAX/VMS INFORMATION
An interactive way of measuring the execution time of a DCL-command is obtained by surrounding the command by the DCL-command SHOW PROCESS/ACCOUNTING and taking the difference of the elapsed CPU time.
The logfile of a batch job contains the elapsed CPU time of the total job.

COMPILING A FORTRAN PROGRAM
synopsis of command procedure 'HANDY'FOR

J.B.H.M. van Gils

ABSTRACT
To compile a FORTRAN-77 program mostly no qualifiers are needed. In that case you can use the DCL-command FORTRAN sourcefile+sourcefile...
The user without experience is guided by the conversation in FOR.COM when composing the command string with qualifiers and executing the compilation. Called in a command procedure the names of resulting files are passed.

RESULT
The names of resulting outputfiles of the compilation are displayed and stored in a global symbol. Outputfiles have a default filetype.

INSTRUCTIONS
There are three kinds of procedure calls:

```
@'HANDY'FOR
@'HANDY'FOR input_filename+... string_of_qualifiers
@'HANDY'FOR input_filename+...string_of_qualifiers
```

If all parameters are empty they are conversationally be asked for. An empty parameter value is defined by "", an empty answer is defined by a RETURN.
The default filetype of inputfiles is .FOR

CONVERSATION
For each qualifier there is a prompt. The user may point to one of a set of automatically composed qualifier strings. Or he may continue with defaults only. Moreover there appears help information when no right choice was made.

REMARK
The G_floating datatype is not supported by the STAVAX processor.
The G-floating qualifier is not supported by the command procedure.

PROGRAM INFORMATION
The command procedure is written in DCL.
The guide is in file 'HANDY'FOR.TXT. The source program is in file 'HANDY'FOR.COM.

VAX/VMS INFORMATION
The use of FOR command qualifiers is described in the VAX-11 FORTRAN User's Guide.

COMPILING, LINKING AND RUNNING A PROGRAM
synopsis of command procedure 'HANDY'FORLINRUN

J.B.H.M. van Gils

ABSTRACT
A fortran program is compiled, other object modules and libraries are linked and the executable program is started.
Combinations of libraries can be linked to the users program.
Simple names substitute the strings of libraries on STAVAX installed and containing subroutines of packages publicly accessable by fortran programs.

SUBSTITUTIONS
An object file may be replaced by a substitution name known by this command procedure. Substitution names are replaced by installed library names.

substitution name	description (access to specific information via the system manager)
&DI3DCAL	DI-3000 plotting routines for CALCOMP plotter
&DI3DTEK	DI-3000 plotting routines for TEKTRONIX
&HANDY	library SUBROUTIN in the HANDY directory, set of utilities for VAX/VMS users (conversation etc, guide in file 'HANDY'SUBROUTIN.TXT)
&IMSL-D	International Mathematical and Statistical Library, double precision routines
&IMSL-S	International Mathematical and Statistical Library, single precision routines (ICW nota 1465)
&PLXY-11	Plotter User Library
&ORACLE	ORACLE data base management system: HLI CALL INTERFACE The file SYS$LIBRARY:CRTLIB.OLB is not available on STAVAX.
&SIMPLOT	Plotroutines Calcomp drumplotter
&TEKTRONIX	TEKTRONIX PLOT10 Terminal Control System

RESULT
Files with the used executable program and the used instructions are purged and renamed to version ;1 . Used object files are deleted.
The filename of the executable program is displayed.

INSTRUCTIONS

There are two kinds of procedure calls:

```
 @'HANDY'FORLINRUN
 @'HANDY'FORLINRUN  main_program_file  extra_fortran_files  extra_object_files-
   instruction_file
```

If all parameters are empty they are conversationally be asked for. An empty parameter value is defined by "", an empty answer is defined by a RETURN.
The defaults of filetypes used in VAX/VMS are in force.
Filenames and parts of them may be replaced by global symbols surrounded by apostrophes, substitution names may not be replaced.
A file internally organized as a library containing object modules is given as library_file/LIB
Linking HANDY conversational subroutines program instructions on file (default filetype .INS) can be connected to the run. In all other usages the parameter value is meaningless.

CONVERSATION

Help information is available for:

    using default filetypes
    using object libraries
    using substitution names to link known object libraries
    using an instruction file.

Strings of files are constructed by + and , signs.

PROGRAM INFORMATION

The command procedure is written in DCL. The FOR command implies /CHECK=ALL .
The guide is in file 'HANDY'FORLINRUN.TXT. The source program is in file 'HANDY'FORLINRUN.COM.

PRINTING SERIES OF ASCII FILES VIA THE PRINTER CONNECTED TO THE TERMINAL
synopsis of command procedure 'HANDY'PRINT

J.B.H.M. van Gils

ABSTRACT
Series of file contents are sent to the terminal. Every content is preceeded by a control sequence to connect and to set the printer and followed by a form feed and a control sequence to reset and to disconnect the printer.
Fortran source programs (filetype .FOR) are compiled and the resulting listfile is printed.

PRINTER SETTINGS
Combinations of a CIT-101 terminal with a LA120 DECwriter or a MT140L matrix printer and a DT80/1 terminal with a LA120 DECwriter can be handled.
The printer settings are

```
                   :  6 lines per inch,           66 lines per page
    during execution
     LA120 DECwriter: 10 columns in the left margin, 13.2 characters per inch
     MT140L printer :                no left margin, 12.5 characters per inch
                     or as stated in the printed text
    after  execution:
                      1 column  in the left margin, 10   characters per inch
```

To make photocopies without reducing the printed page:
    max. 80 characters in a printed line
    2 blank lines at the top
    max. 58 lines of text
An aborted print preserves present settings. A legal exit of the command procedure sets them equal to the to the after printing state.

SERIES OF FILES
More filenames may be stated separated by comma's.
The wildcard convention '*' results in a header line with the name of the file and bad paging.
Series of filenames may be stated in a combined string using + signs, for example PRINT+TEST.TXT+FOR

INSTRUCTIONS
There are three kinds of procedure calls:

```
 @'HANDY'PRINT
 @'HANDY'PRINT printer
 @'HANDY'PRINT  printer  list_of_file_spec's  default_filetype-
                default_owner_id  more_fortran_qualifiers
```

If required parameter values are empty they are conversationally asked for.
The printer name (terminal/printer combination) and series of files may be stated in a short writing way. In conversational mode HELP texts are available.
By pressing the CTRL/O key the user may interrupt the printing of one file.

PROGRAM INFORMATION

The command procedure is written in DCL.
The guide is in file 'HANDY'PRINT.TXT. The source program is in file 'HANDY'PRINT.COM.

REFERENCES

Gils, J.B.H.M. van, 1983. Aspecten van Informatieverwerking 40. Printmogelijkheden op de Staringcomputer. ICW-nota 1431; pp.12.

PURGING SERIES OF FILES AND RENAMING TO THE LOWEST POSSIBLE VERSION
synopsis of command procedure 'HANDY'PURGE

J.B.H.M. van Gils

ABSTRACT
A list of file specifications may be stated in a short writing way.
The resulting version is one higher then the highest number of the files of one specification.

SERIES OF FILES
More filenames may be stated separated by comma's.
The wildcard convention "*" results in a header line with the name of the file and bad paging.
Series of filenames may be stated in a combined string using + signs, for example PRINT+TEST.TXT+FOR

INSTRUCTIONS
There are two kinds of procedure calls:

```
 @'HANDY'PURGE
 @'HANDY'PURGE  list_of_file_spec's  default_filetype  default_owner_id
```

If all parameter values are empty there is conversationally asked for the list of file specifications.
In conversational mode a HELP text is available.

PROBLEMS
The command procedure does not report when no file has been found.
Renaming the versions of all files for example with *.* makes the temporary files inaccessible for all the command procedures of HANDY.
To delete them type @'HANDY'DELETE "" version_of_IIIDELETE.TMP .
Renaming all the versions in your highest level directory for example with *.* makes the subdirectories inaccessable for you and for the login procedure.
Then type RENAME [....]*.dir *.dir;1 .

PROGRAM INFORMATION
The command procedure is written in DCL.
The guide is in file 'HANDY'PURGE.TXT. The source program is in file 'HANDY'PURGE.COM.

READING TAPES WRITTEN IN FORMATS WITHOUT RECORD COUNT FIELDS
synopsis of command procedure 'HANDY'TAPECOPY

J.B.H.M. van Gils

ABSTRACT
The command procedure TAPECOPY allocates and mounts your magtape if not already mounted, runs program TAPECOPY to read files from tape and dismounts and, if you want, deallocates the tape.

PROGRAM TAPECOPY
Program SYS$SYSDEVICE:[UTIL.TAPECOPY]TAPECOPY adjusted and installed by L.P. Kamil, skips files or reads files from tape written in 800 or 1600 bpi to disk. Program TAPECOPY always reads sequential from begin of tape, a restart of the program works like a rewind magtape. Every successive mark on the tape is read as an end of file, so labels may be skipped or read as files.
EBCDIC is translated to ASCII.

INPUT
The logical record length in the inputfiles may be fixed, only ASCII records may have variable length when ended with CrLf or LfCr.

OUTPUT
The user defines the names of the outputfiles (the default filetype is .DAT). Information in existing outputfiles is overwritten. The outputfile is sequentially organized with variable record format.

INSTRUCTIONS
There are two kinds of procedure calls:

```
@'HANDY'TAPECOPY
@'HANDY'TAPECOPY  800_or_1600
```

Both, the program and the command procedure, guide the user in conversational mode via the terminal also when the command procedure TAPECOPY is called in a command procedure.

REPORT
When the command procedure is stopped a message shows the situation in which the driver and magtape are left.

PROGRAM INFORMATION
The command procedure is written in DCL.
The guide is in file 'HANDY'TAPECOPY.TXT. The source program is in file 'HANDY'TAPECOPY.COM.

REFERENCES
Gils, J.B.H.M. van, 1983. Aspecten van Informatieverwerking, 42.
Magtape verwerking op de Staringcomputer.
ICW-nota 1452; pp.20+8

EXECUTING ORACLE SQL STATEMENTS
synopsis of command procedure 'HANDY'ORAUFI

J.B.H.M. van Gils

ABSTRACT
SQL statements in a user command procedure can be executed by a ORAUFI call. The stated ORACLE SQL statements are collected in a UFI command file and executed or a stated UFI command file is executed.

INSTRUCTIONS
There are two kinds of procedure calls:
```
@'HANDY'ORAUFI SQL_statement_text  username/password  SQL_worksize_area
               global_symbol  version_save_file
@'HANDY'ORAUFI &UFI_command_file  ""  ""  global_symbol  version_save_file
```
If needed parameters are empty they are conversationally asked for.
An empty parameter value is defined by "". An empty answer is defined by RETURN.
A UFI command filename (default filetype .UFI) is defined by a preceding ampersand (&) sign. All UFI commands from username until EXIT must be in the file.
A stated SQL_statement_text may be or may not be closed by a ; sign.
The SQL work size area (default 3K) is given in Kbytes (*1024 bytes).

OUTPUT
The global symbol in the call returns the name of the logfile. An empty value writes the loggings to SYS$OUTPUT and deletes all files made by the command procedure.
The save file stores the names of all files used by the command procedure.
The save file is a workfile to be used by command procedure 'HANDY'DELETE (deleting files with saved names).

PROGRAM INFORMATION
The command procedure is written in DCL.
The guide is in file 'HANDY'ORAUFI.TXT. The source program is in file 'HANDY'ORAUFI.COM.

LOADING BULK DATA INTO ORACLE
synopsis of command procedure 'HANDY'ORAODL

J.B.H.M. van Gils

ABSTRACT
The ODL program reads bulk data into the ORACLE base.
Mostly bulk data input is found on a sequential ASCII file with file attribute "VAR" (outputfile of editor EDT), where the data are positioned in columns.
When using this input type the procedure ORAODL translates more simply defined ORAODL instructions to the ODL form and executes ODL.
ORAODL has not been equipped for conversational use.

INSTRUCTIONS
There are two kinds of procedure calls:

```
@'HANDY'ORAODL field_instructions  datafile  ORACLE_table  username/password-
               global_symbol  version_save_file
@'HANDY'ORAODL field_instruction_file  datafile  ORACLE_table  username/password-
               global_symbol  version_save_file
```

An empty parameter value is defined by "".
The field instructions may be found in the execute ORAODL command or in a stated file. A field instruction is stated

```
   as:   ORACLE_field_name ( position_from , position_thru )
or as:   ORACLE_field_name ( position )
or as:   ORACLE_field_name ( NULL )
```

The position numbers are the sequence numbers of the characters in the input record, where the value to be stored in the ORACLE field is found. Non-existing numbers, a descending interval and the word "NULL" write a NULL value in the ORACLE field. Spaces may be used at any place in the field instructions.
The field instructions in a file are read from more records till an end_of_file.

OUTPUT
Every inputrecord accepted by ODL inserts an inputrecord in the ORACLE table.
The filenames of the ODL instruction set (filetype .CTL), the logfile made by ODL and ORAODL (filetype .LOG) and the file with the inputrecords rejected by ODL when they do not match the definitions (filetype .BAD), are returned in the global symbol stated at command level.
An empty global symbol name displays the contents of these three files and deletes all files made by the command procedure.
The save file stores the names of all files used by the command procedure.
The save file is a workfile to be used by command procedure 'HANDY'DELETE (deleting files with saved names).

PROGRAM INFORMATION
The command procedure is written in DCL.
The guide is in file 'HANDY'ORAODL.TXT. The source program is in file 'HANDY'ORAODL.COM.

LISTING PARTS OF RECORDS OF SEQUENTIAL ASCII FILES
synopsis of program 'HANDY'LIST

J.B.H.M. van Gils

ABSTRACT
Stated parts of every record in the input are combined to a string.
The output string is written as a number of subsequent records controlled by their maximum length. Both input record and output string may contain at most 2000 characters.

INPUT
Horizontal tabs in the input define the position of the characters in the inputrecord according to their settings in the program.

OUTPUT
Even when the input does not have Fortran Carriage Control the outputrecords does have a FCC character in the first position.
Trailing spaces and nulls in output are removed.
Output may be paged.

PAGES
A page is limited by the maximum form length.
Optionally at most two header records may be added, one with page number and filename and after that one with date and time of the output.

CONTROL CHARACTERS
Non-paged output contains unchanged control characters.
Implied control characters in the output string to be paged also control the division into records. The characters Carriage Return, Line Feed, and Form Feed are processed in the way their names indicate, vertical tabs result in a double line feed.

INSTRUCTION INPUT
Normally instructions can be stated when the program asks for them conversationally (english). Moreover there is a users guide.
Optionally instructions can be read from file. The records read contain the answers to the subsequent questions that the program would have displayed when working conversationally.
Common instructions may be closed by a / followed by comment. This is not allowed when it can be interpreted as a program requested ASCII string.
Instruction parts given once are reused when not changed.

INSTRUCTION OUTPUT
Some more complex instructions are redisplayed by the progam in an interpreted form.
Optionally instructions read from file can be displayed together with the questions in conversational mode they belong to.

PROGRAM INFORMATION
The program is written in fortran-77.
The executable program is in file 'HANDY'LIST.EXE. The guide is in file 'HANDY'LIST.TXT. The source program is in file 'HANDY'LIST.FOR.

OBJECT MODULES TO BE LINKED
synopsis of VAX/VMS library 'HANDY'SUBROUTIN

J.B.H.M. van Gils

ABSTRACT
The library file SUBROUTIN contains the object modules of all HANDY subroutines.

LINK
The user may link HANDY subroutines to his program with the help of command procedure 'HANDY'FORLINRUN or with
DCL-command: LINK program_file,...,'HANDY'SUBROUTIN/LIBRARY
The /LIBRARY qualifier in the link command specifies that the input file is an object-module library that is to be searched to resolve undefined symbols referenced in other input modules. The default file type is .OLB

LIBRARY INFORMATION
The guide is in file 'HANDY'SUBROUTIN.TXT.
Only the most recent version of the library file is kept available.
The guides of the subroutines are in files named with the program name and with filetype TXT.
The source files with file type FOR of the subroutines are written in fortran-77.
They have been compiled with DCL-command: FORTRAN/CHECK=ALL subroutine_name

DIRECTORY
The DCL-command giving a directory of the library file is:
```
    LIBRARY/LIST=SUBROUTIN SUBROUTIN    ! gives directory in file SUBROUTIN.LIS
```

INTRODUCTION TO THE 'INSTRUCTION SUBROUTINE SET'

A system of reading and reporting program instructions is partly preprogrammed.
The concepts are listed below.
An example of using this system may be found in the source file
'HANDY'LIST.FOR .
The flow chart of programming with the 'instruction subroutine set' in program
LIST is found in the appendix.

CONVERSATION OR INSTRUCTIONS FROM FILE

Instructions can be stated when the program asks for them conversationally.
Optionally instructions can be read from file. The read records contain the answers to the subsequent questions that the program would have displayed when working conversationally.
Question texts are enlarged to a fixed length. The conversation scrolls vertically over the screen.

REUSE OF INSTRUCTIONS

Instruction parts given once are reused when not changed. After processing, a series of instructions controls the flow trough the instruction input parts.

EMPTY INSTRUCTIONS

When no instructions, empty instructions or unreadable instructions are read, the subroutines return a default value as set by the calling program.
An empty number or a ? sign given instead of a number surpresses an error message. Doing so a value representing missing data can be entered.
An error message as a consequence of an instruction read from file may cause a program stop.

ADDING COMMENTS

Commonly instructions may be closed by a / followed by comment. This is not allowed when it can be interpreted as a program requested ASCII string.

READABLE FORMATS

When only a one letter instruction must be given (Yes or No), the answer is read in the first position of a one record series. When a one word instruction must be given (a filename) no spaces may preceed the word; a one number instruction, however, may be preceeded by spaces. One word and one number instruction are read in a one record series. A series of values separated by prescribed strings (spaces, comma, return, AND,...) may be given in a series of more records closed by a / sign. When a text string must be given, the full string till end of record is returned to the calling program.

N-BYTE VALUES

A series of numbers and character strings can be read into an array of 2 or 4 byte elements of numeric datatype (n-byte values). A commonly notated number is decoded as a binary value, a string of characters between apostrophes is decoded as a serie of binary values representing n-byte strings. A non n-fold number of characters is enlarged with spaces. The apostrophe sign in a string is decoded from 2 consecutive apostrophes. An unreadable notation, a ? sign or null characters in a character value returns a value representing missing data to the calling program.

INSTRUCTION OUTPUT

Some more complex instructions are redisplayed by the progam in an interpreted form.
Optionally instructions read from file can be displayed together with the questions in conversational mode they belong to. Program instructions included at DCL command level are never reported.
The interactive user is told processing is still going on by the display of a line with text, time and date every 3 minutes.

ARGUMENTS COMMON TO THE 'INSTRUCTION SET SUBROUTINES'

argument	contents
UNITI	Logical Unit Number of the instruction input
	UNITI=ACCEPT conversational instruction input from SYS$INPUT
	UNITI#ACCEPT instruction input from a file
ACCEPT	LUN of SYS$INPUT (default 5)
UNITO	LUN of SYS$OUTPUT (default 6)
REPORT	.TRUE. displays input when input is read from file

MAIN PROGRAM STRUCTURE

The main program mostly can be structured in consecutive parts:
  Initialization
  Instruction input programmed in parts and controlled by instructions
  Processing
  Input of flow control instructions
  Stops and messages
User defined checking, non standard preprocessing and non standard output of instructions by the main program is commonly placed under control of the flow of instruction input.

PROGRAM INFORMATION

All subroutines are collected in object library 'HANDY'SUBROUTIN.
From every subroutine there is a source program file 'HANDY'HANDY---.FOR and a guide file 'HANDY'HANDY---.TXT .

ABSTRACTS OF SUBROUTINES IN THE INSTRUCTION SET

HANDYFLD
Reads the filename, connects a logical unit number to the file and opens it for unformatted direct access input or in-/output. Only to enable opening a new file the dimensions must be stated. The value representing missing data of an existing file is expected in the last 16 bits value of the file. A new file is totally filled with this value. The questions are in english.

HANDYFLI
Reads the filename, connects a logical unit number to the file and opens it for sequential formatted input to read instructions. When instructions are read from file the report instruction is read to make simulate the conversation. The questions are in english.

HANDYFLS
Reads the name of a sequential formatted file for data input or output, connects a logical unit number to that file and opens it. An output file is created as a new version.

HANDYFOLD
Connects a logical unit number to an existing unformatted file and opens it for direct access.

HANDYINIT gives initial values for running with HANDY subroutines and automatically opens an instruction file when existing

HANDYLOOP
Reads series of an initial, terminal and increment loop parameters (from, to, step) and generates subsequent integer*2 values following these loops.

HANDYNUMB
Reads a real or integer number in decimal notation with or without an exponent. An empty number or a ? returns the default value.

HANDYROW
Reads a row of numbers and character values from instruction input and stores them as subsequent n-byte values in an array. The input format is similar to the format in the fortran list directed read statement.
A empty value and a ? are each converted to one value representing missing data.

HANDYSTRI
Reads and optionally reports a character string without displaying a question. The string can be read from sequential input.

HANDYSTRN
Reads and optionally reports a character string optionally with conversation. The string can be read from sequential input.

HANDYTIME
Writes a line with text, date and time after 3 minutes or more from the moment the last textline has been written.

HANDYYORN
Asks Yes or No from instruction input.

INTRODUCTION TO THE 'ORACLE SUBROUTINE SET'

The set of subroutines makes programming the access to the ORACLE base in a fortran program somewhat easier and more surveyable.
The reader is expected to be familiar with the ORACLE language SQL (see ORACLE, 1983a), VAX-11 fortran and the ORACLE Host Language Call Interface HLI (see ORACLE, 1983b).

LIMITATIONS

Using ORACLE version 3 to do research work on a VAX computer makes acceptable to limit the use of possibilities. This means no version 2 calls, no audit, only autocommit and reference only the substitution variables by name.

CONCEPTS

The subroutines are based on the following concepts:

- Cursor Data Area
  The CDA's to be connected to the SQL statements are collected in one program defined array.
- Field buffer
  The data buffer areas in the user program connected to the fields in the SELECT list of an SQL statement are consecutively located in a program defined buffer. The corresponding field lengths, conversion codes and field RETURN code adresses are consecutively located in two byte integer arrays.
- Cursor RETURN code
  The Cursor RETURN code returned by ORA--- subroutines must control the action in the calling program. Only the subroutines ORALOGON and ORALOGOFF force a fortran stop when the result is not successful.
  The code -32767 ('not legal field conversion') has been added.
- Field RETURN code
  The field RETURN code is the code returned by the last operation of subroutine ORAFETCH with a non-zero RETURN code for this field.
  This value may not be changed in the calling program. It is set, changed and used by subroutine ORAERROR.
- Missing Value Indicator
  The occurrence of no value or a null value in SQL is connected to a Missing Value Indicator in ORA--- subroutines. The MVI in a character string is spaces only. The MVI in a numeric field is the largest negative workable binary value in the defined datatype, -127, -32767, -2147483647, or -1.7E38, left justified in the field.

MAIN PROGRAM STRUCTURE

To connect and execute a simple SQL SELECT statement once, the following ORA--- subroutines using HLI modules are used in sequence:

- connecting the SELECT statement and defining the field areas:

ORALOGON	logs on to ORACLE
ORASQL	defines an ORACLE SQL statement

- defining the substitution values:

ORABIND	assigning a program defined value to an ORACLE SQL substitution variable
ORAEXEC	processes an ORACLE SQL statement

- executing the SELECT statement:

ORAFETCH	returns a row of an ORACLE query result
(OCLOSE)	delete a cursor (set free for re-use of the CDA)
ORALOGOFF	logs off from ORACLE

After each HLI call the user program must define a control action depending on the returned cursor RETURN code. The ORACLE information belonging to an issued non zero RETURN code is automatically displayed.

PROGRAM INFORMATION

All subroutines are collected in object library SUBROUTIN.
From every subroutine there is a source program file 'HANDY'ORA---.FOR and a guide file 'HANDY'ORA---.TXT .

LINKING HLI MODULES

On STAVAX computer ORACLE is invoked by DCL-command: @SYS$ORACLE:ORAUSER
ORA--- subroutines call for HLI modules. In the LINK command the object file string 'OLB$:OCLIB/L+ORACLE/L+UPILIB/L+CLIB/L+ORACLE/OPTIONS' must be used.
Linking HLI, HANDY and other libraries may be stated somewhat easier with:

```
@'HANDY'FORLINRUN  main_program_file  extra_fortran_files  &HANDY+&ORACLE...
                   instruction_file
```

REFERENCES

ORACLE Oracle User Manual Volume I version 3.1
(Relational Software Inc., 1983a)

ORACLE Oracle User Manual Volume II version 3.1
(Relational Software Inc., 1983b)

ABSTRACTS OF SUBROUTINES IN THE ORACLE SET
****

------------------ intended to be called by user programs -------------------

ORABIND
   Assignes a program defined value to an ORACLE SQL substitution variable

ORAEXEC
   Processes an ORACLE SQL statement

ORAFETCH
   Returns a row of an ORACLE query result
   The ORAFETCH call returns one row at a time. Each field of the query result is placed into a field area of a program defined buffer identified by a previously executed ORASQL call.
   The arguments of the ORASQL call and the ORAFETCH call are the same.
   Fields that are requested in character string format are left justified and padded with trailing blanks.
   After each fetch the cursor RETURN code is updated. In the field RETURN code only the last occurred non-zero value is stored.
   The ORACLE RETURN codes +2 and +4 do not write an error message.

ORALOGOFF
   Frees all ORACLE resources owned by the program.
   A fortran stop is forced when the result is not successful.

ORALOGON
   Communication is established between ORACLE and the user program. All CDA's to be connected to the specific database are opened and their SQL work areas (SWA) are defined.
   A fortran stop is forced when the result is not successful.

ORASQL
   The SQL statement is passed to ORACLE and associated with an open cursor.
   An output buffer is defined for the fields in the SELECT list.

------------------ for supporting other ORA--- subroutines ------------------

ORACHECK
   Checks length and datatype of a program defined buffer area connected to ORACLE. When using ORA--- subroutines a field length zero is not accepted. When a non-legal field conversion is detected, a message occurs and the cursor RETURN code has the value -32767

ORACURSOR
   Returns ORACLE Cursor Area Data, i.e., rows processed count, number of variables bound, parse error offset, function code and HLI module name as far as they are relevant.

Subroutine ORAERROR
   In 'HANDY'ORA--- subroutines the HLI calls are followed by a call for subroutine ORAERROR, which writes information when the RETURN code is not zero. Only code +2 (null value encountered in any field of fetch) and code +4 (end of fetch) are returned by subroutine ORAFETCH without a message.

ORAMVI
   Returns a Missing Value Indicator in a program defined buffer area connected to ORACLE

MARKING A POINT AS BEING INSIDE OR OUTSIDE AN AREA
synopsis of subroutine 'HANDY'ALGOINOUT

W. van Doorne and J.B.H.M. van Gils

ABSTRACT
A point (Xpoint,Ypoint) is marked as internal point or external point of an area bounded by polygons defined by their vertices. So an area may consist of one or more separate polygons and each of them may contain (nested) enclaves.

INPUT
Each point is defined by its coordinates (X,Y).
When marking, the vertices (Xvertex,Yvertex) must passed trough such that the area is kept at the right hand side.
The subroutine can transform the vertices given in clockwise rotation sequence with the polygons and enclaves closed by special codes.

ALGORITHM
Marking is achieved by calculating the total angular rotation with respect to (Xpoint,Ypoint) when passing through the vertices of the area boundaries in the presupposed sequence. (Xpoint,Ypoint) out of range a priori is marked exterior.

RESULTS
Marking is not accurate when the distance of (Xpoint,Ypoint) to a boundary is less than 0.00001 .
Moreover the area is returned in square units of the coordinates.

CPU-TIME
When a larger number of (Xpoint,Ypoint) is marked an estimation of VAX central processor time (CPU sec.) per marked point is found from

  CPU = c * (number of vertices)

where c varies between 0.001 and 0.002 .

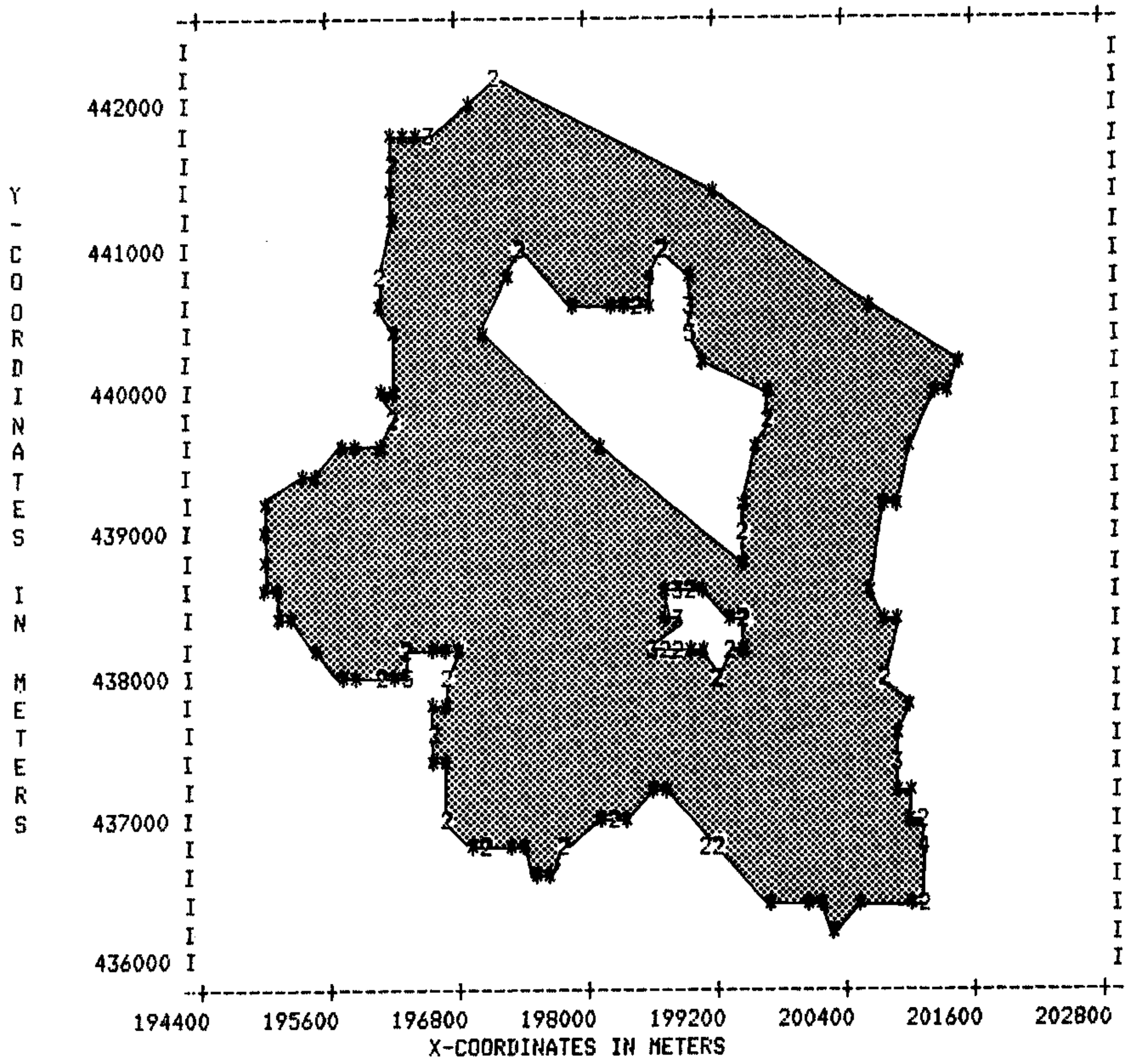
PROGRAM INFORMATION
The program is written as a subroutine in fortran-77.
The object module ALGOINOUT is in library 'HANDY'SUBROUTIN.OLB. The guide is in file 'HANDY'ALGOINOUT.TXT. The source program is in file 'HANDY'ALGOINOUT.FOR.

References

DOORNE, W. van, 1973. Een methode ter bepaling van de inwendige roosterpunten van gedeelten van het platte vlak, zoals te gebruiken bij het samenstellen van een begroeiingskaart in de landschaps-analyse. ICW-nota 760: pp.23+2.
------- personal file.

Example of an area on which marking was applied.

A map of the area of Duiven (Netherlands) showing the rural (shaded) region. The boundaries of rural and urban areas are drawn as polygons. Vertices may coincide in a plot location. The number of coincidences is given in that location.



SORTING AN ARRAY AND OPTIONALLY RANKING ITS MATE
synopsis of subroutine 'HANDY'ALGOSORT

J.B.H.M. van Gils

ABSTRACT
The used procedure, superSHELLsort, is described by Barron and Diehr (see reference). It is an algorithm to sort numbers upwards without using an extra array core.
Values representing missing data are shifted to the end of the array.

DATA
The data to be sorted and the mate data are stored as 2-byte values.

OPTIONS
The array of mate data is optionally ranked.
When sorting alfabetic data and/or sorting downwards the data are transformed previous to the internal sorting operation. Optionally the data may be reset to initial form.
Every 3 minutes a line may be displayed having a text and the clocktime.

PROGRAM INFORMATION
The program is written as a subroutine in fortran-77.
The object module ALGOSORT is in library 'HANDY'SUBROUTIN.OLB. The guide is in file 'HANDY'ALGOSORT.TXT. The source program is in file 'HANDY'ALGOSORT.FOR.
In a DCL command the term 'HANDY' is meant to be substituted by a global symbol.
On STAVAX computer the HANDY directory is defined in the user's global symbol table.

References

Barron, T. and G. Diehr, 1983. Sorting Algorithms for Microcomputers.
BYTE, the small system journal, Vol.8, No.5, page 487-490.

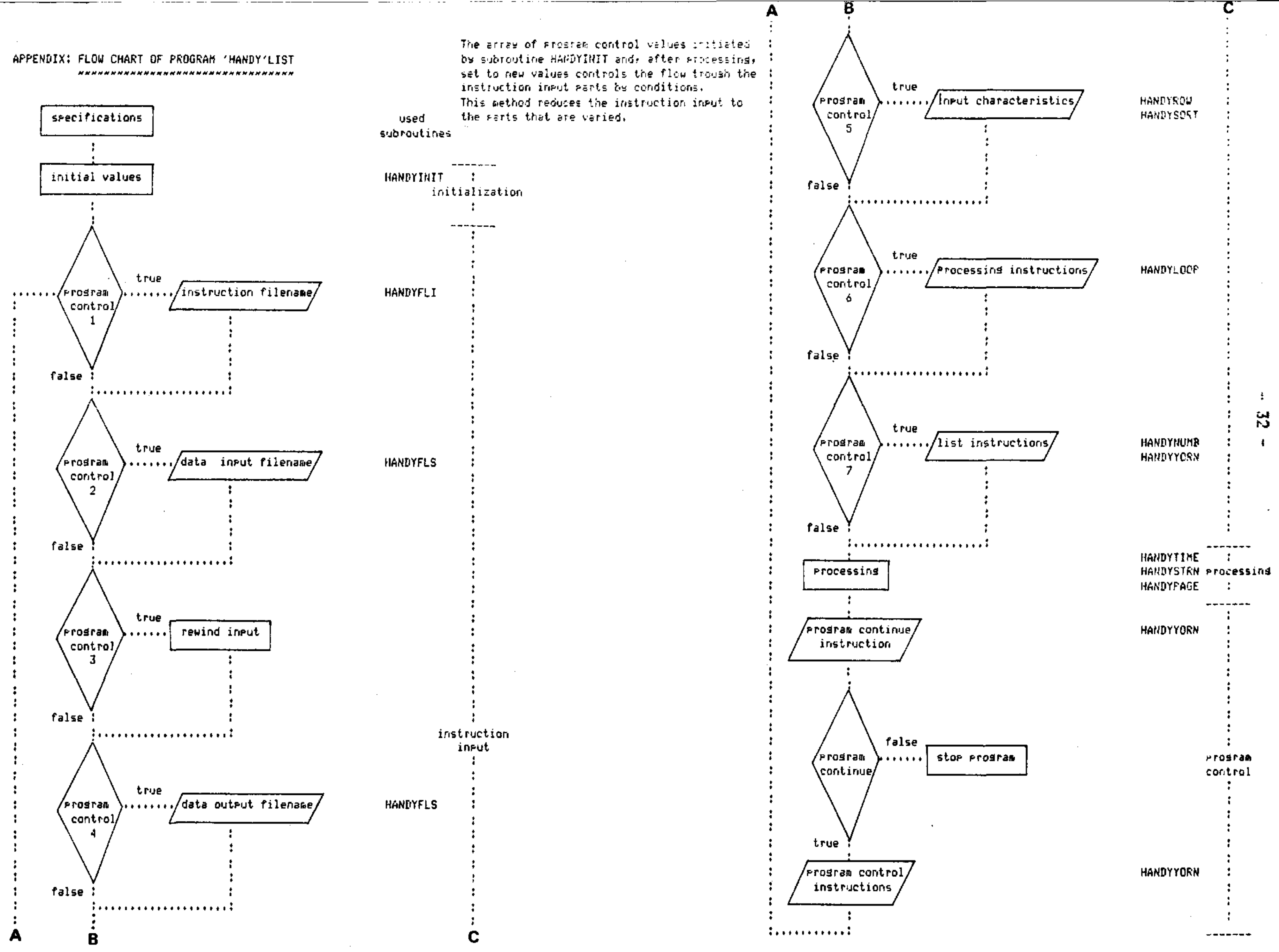
ABSTRACTS OF SUBROUTINES NOT INTENDED TO BE CALLED BY USER PROGRAMS
##

HANDYALFA
   Adds a character occurring between apostrophes to an n-byte value.
   Only a character surrounded by apostrophes is added. The character is added
   by the call after the call in which the character occurs.
HANDYASK
   Writes a question enlarged to fixed length.
   The cursor remains positioned after the written text.
HANDYCSTR
   Writes a string including some pointer.
HANDYDECO
   Reads a number from a string.
   Foregoing spaces, nulls and tabs are not used. A ? is translated to the
   default value. The number of trailing non-decoded characters is returned.
HANDYERR
   Writes a FORTRAN run-time error message belonging to an I/O error occurrence.
   Together with it a message from the calling program and the logical unit
   number used in the I/O statement is written.
HANDYFLN
   Reads a filename, closes the connected unit and sets the Logical Unit Number
   to a new value. When no filetype is in the name a default filetype is added.
HANDYPAGE
   Writes a string in one or more lines and optionally pages output.
   When paging, only printable characters are expected in the string.
HANDYSKIP
   Counts the length of instructions in a line

PROGRAM INFORMATION
All subroutines are collected in object library 'HANDY'SUBROUTIN.
From every subroutine there is a source program file 'HANDY'HANDY---.FOR and a
guide file 'HANDY'HANDY---.TXT .

APPENDIX: FLOW CHART OF PROGRAM 'HANDY'LIST

The array of program control values initiated by subroutine HANDYINIT and, after processing, set to new values controls the flow trough the instruction input parts by conditions.
This method reduces the instruction input to the parts that are varied.


specifications
initial values
program control 1
true
instruction filename
false
program control 2
data input filename
program control 3
rewind input
program control 4
data output filename
A
B
C
used subroutines
HANDYINIT
initialization
HANDYFLI
HANDYFLS
instruction input
HANDYFLS
program control 5
Input characteristics
program control 6
Processing instructions
program control 7
list instructions
Processing
program continue instruction
program continue
stop program
program control instructions
HANDYROW
HANDYSORT
HANDYLOOP
HANDYNUMR
HANDYYORN
HANDYTIME
HANDYSTRN
HANDYPAGE
processing
HANDYYORN
program control
HANDYYORN