# USER INTERFACE MODULE COMPRISING BOTH MENUE AND COMMAND LANGUAGE TECHNIQUES - AS APPLIED IN PHOTOGRAMMETRIC PROGRAM SYSTEMS

L. Molnar
Institute of Photogrammetry
Technical University of Vienna
Austria
A. Köstli
Institute of Photogrammetry
University of Stuttgart
Federal Republic of Germany
Commission IV

## ABSTRACT

A FORTRAN module (DRE-X) is described, comprising two important techniques to interface the user in computer programs: command language, and menues. The user can switch from one of these techniques to the other any time during control input. Both techniques are structured in parallel stages in order to maintain correspondence between them. Menues are adapted to prepare the user for the application of the command language. For users preferring command language, corresponding menues - if requested, - provide a detailled guidance, and an echo of input. Examples are taken from the application of DRE-X in photogrammetric program systems such as SCOP (to create, maintain, and interface digital elevation models), and SORA-MP (Software for Off-line Rectification with Avioplan-Map Projections).

## ZUSAMMENFASSUNG

Der FORTRAN-Modul DRE-X verwirklicht zwei wichtige Methoden der Eingabe von Steuerinformationen: die Kommandosprache und die Menütechnik, wobei die Umschaltung zwischen diesen beiden jederzeit möglich ist. Der Prozeß ist in beiden Fällen in gleiche Stufen gegliedert, wodurch die Zusammenhänge übersichtlich bleiben. Der Benutzer wird durch den Aufbau der Menüs auf die Anwendung der Kommandosprache vorbereitet. Andererseits bilden die Menüs für den Benutzer der Kommandosprache eine wahlweise Führung und ein Echo auf die Eingabe. Die Beispiele für die Anwendung von DRE-X stammen aus photogramm metrischen Programmsystemen SCOP (Aufbau, Laufendhalten und Auswerten von digitalen Höhenmodellen) und SORA-MP (Software for Off-line Rectification with Avioplan-Map Projections).

## INTRODUCTORY NOTES

Methods of computation have always been of great importance in geodesy. Works dealing with means and rational organization of computation are, correspondingly, a major chapter in geodetic literature, and an important contribution to other disciplines, as well. In light of such traditions, and given the central role played by computers in to-days photogrammetry, it is difficult to believe that there is hardly a commission or working group within the ISPRS interested in the problems and solutions dealt with below. It would be our suggestion to the Congress to remedy this problem by creating at least a working group for special questions of software development.

Modules and thoughts represented in this paper came to being in course of work on programs and program systems ORIENT /1/, SCOP /2,3/, TOPIAS /4/. SORA-MP /5/, and others. We are greatful to our colleagues in Stuttgart and

in Vienna as well as to the users of the programs mentioned in different countries for their remarks, ideas, criticism, and the opportunity to learn different computers, manisided working conditions, and applications. Our special thanks are due to the Federal Bureau of Road Construction (BAST) in Cologne, and to its six member organizations. Since 1983, our command language has been accepted as the official one for software development there - a measure aiming at better uniformity of control information input in different programs.

## 1. INTERFACING USERS TO CONTROL PATHS OF PROCESSING

In recent years, the attention raised enormously paid to often luxurious ways of interfacing the user in computer programs. Reasons for this are controversial. On the one hand quality and easyness of communication between humans and computer programs decides to a large extent the effectiveness of applying artificial intelligence. New, highly improved ways of this communication widen, indeed, limitations on program system complexity, while at the same time they make applications rather more convenient. On the other hand, user interfaces are often confused with wrapping paper deciding the appearance of products for sale.

Frequently used ways of control information input are:
- command language (with only a few exceptions, operating systems are controlled by this method)
- line oriented menues of numbered choices
- dialogue (answering questions)
- masks (tables to be filled in, with the cursor jumping automatically to the next field)
- menue tablets (with a hardware cursor: the "mouse"), or their equivalent on an active screen.

Arguments offered in the many disputes and discussions on these methods could be classified as follows.

(1) Aspects of computer environment
- in programs intended for both batch and interactive modes of operation, command language is the only choice of the ways of control input mentioned;
- menues, masks, and all ways similar to them require fast rates of communication between computer and terminal. At rates below (or maybe even equal) 1.200 bauds such methods cannot be used; command language does not have this limitation
- programs intended to be machine independent (not to say terminal independent), cannot apply techniques for which direct addressing of screen position (I/O) would be inevitable. This limitation does not concern command language, menues (if organized correspondingly), and dialogue.

(2) Complexity and relative amount of control input
Dialogue is the choice for processes controlled by just one or two input parameters. Processes requiring the definition of up to some 10 parameters representable in a single menue, are probably best served by the menue techniques - especially if most of these parameters remain unchanged, and if the process is not used all to offen (e.g. reading/writing foreign magnetic tapes). Hardly anyone will control a text editor by line oriented menues. Highly complex systems controlled by hundreds of parameters need, on the one hand, compact and effective ways of control input for experienced

users (i.e. command language), and, on the other hand, detailled help- and/or guidance capabilities for beginners, special situations, etc. Interactive graphics is best served by menue tablets and/or command language input. The conclusion is that complexity and relative amount of control I/O influence choice of the method of control I/O considerably.

(3) Special aspects of the programmer concern complexity and flexibility of the interface. Modules for command languages, menue generators, tablet drivers and the similar can only be adequate when having complex (and therefore tiresome) communication with the calling program. The main advantage of such central modules is, probably, felt in homogeneity and sophistication; economy in source text comes second.

(4) Program users' aspects depend much upon their experience, and character. Detailled guidance pleasing the beginner will annoy the user working with the program on a day-by-day basis. It is important to stress at this point that neither detailled guidance nor sophisticated help functions can replace an adequate manual, not even at the cost of considerable and damaging simplifications of system capabilities. Exceptions to this rule may be some very simple routine procedures.

Considering the character (or just momentary condition) of users one has to differentiate between those with an active attitude in calling up different processes, and those preferring a guidance in their actions. Beyond doubt, command language is the choice of the first type, whereas menues, masks, dialogue etc. suite the passive type better. To put it differently, the active user considers menues as a help function, whereas the passive user sees a guidance in them. Going over from menues to command language is not just a step-up in knowing the process better: it is also a change of attitude.

Simple things, such as the ability to type, influence user preferences, as well.

(5) Selling and bying software products is seldomly business of experienced users. Methods of processing control intended for beginners are better suited for program demonstration than the compact, more effective ways for advanced users. These aspects resulted in overemphasis on spectacular user interfaces some of which play a Bach cantata every time the user succeeds in making the right entry. It is our hope that such options can be turned off in a production environment.

*

The subject matter of this paper, module DRE-X, has been developed to meet the following requirements:

- very high level of control complexity; scientific/mathematic type of application
- suiting both interactive and batch modes of operation as well as both low and high rates of communication
- satisfying the needs of both active and passive, experienced and beginning users
- machine (computer and terminal) independency.

Comparing these points with the different ways of interfacing the user, the conclusion is natural of incorporating in DRE-X both command language and line oriented menues, complete where necessary with dialogue.

## 2. THE MODULE DRE-X

DRE-X is a means of transferring program control information from the user to some program system. To enter control information under DRE-X, the user has the choice between command language (DRE: Directive REcognition), and menue techniques. To switch over from one mode of input to the other, the $X-entry is used; this is the origin of the "-X" extension to "DRE". Partial implementations of DRE-X may contain just command language input (DRE), or just input using menue techniques.

### 2.1 Command language

Processes and their control parameters are named by mnemomics (words) in a command language. To activate a process (such as for instance absolute orientation), the user has to type the name of it (ABSOR), and a series of parameters for it (e.g. XYTOLERANCE=2.5 METER). The command language recognizes the words, ascribes parameters the values entered, and starts the process requested.

Freedom of input is limited by syntactic rules (grammar) as defined by the command language. To maintain a strict syntax requires additional code in the language routines, but it enables better recognition of erroneous entries. In DRE, main lines of syntax are as follows.

Each process (generally some "software tool") is activated by a sentence, termed "directive". A directive consists of its name (e.g. ABSOR), and its parameters (e.g. XYTOLERANCE). Parameters may get one or more values, and so-called attributes (such as METER). The end of a directive (of the sentence) is marked by a semicolon (if not re-defined by the user). A directive may be entered in any number of lines; and any number of directives may be written in one and the same line. Words may be abbreviated. - Different forms of directive ABSOR could be:

```
ABSOR,XYTOLERANCE = 2.5 METER;
ABS,XTL = 2.5 M;
A X M 2.5;
```

In DRE routines, corresponding internal levels take care of directives, parameters, attributes, and values. Furthermore, directives may open (in complex systems) directive segments (tree structure of directives). Entering a question sign at any stage of control input yields a list of mnemonics available at that point (table 1).

Command files with user-written directive procedures on them are an often used feature of DRE. These procedures can be addressed (called up) interactively, and they are then performed with verification of actions from the terminal. Error handling in DRE, and DRE system functions are further points to be at least mentioned.

*

Interface between DRE and the calling program consists exclusively of lists of parameters to four DRE subroutines. The first of these serves purposes of initialization, the second one handles directive names of one directive segment, the third one processes a list of parameters, attributes, and values to some directive, and the fourth one enables program-intern activation of DRE system functions. All mnemonics, corresponding data types etc. are

Example from SCOP I: DRE-internal levels, ? - option (Directive GROUND, REFSYS (1 1 1 1 2 2 2 2) = METER;)

Table 1

| N | Screen | Comments |
|---|---|---|
| 1 | DIR: ? | Prompting DIR: internal level 1 (of directives) |
| | DRE DIRECTIVES (SYSTEM-DIR.SEE$?) | |
| 2 | OPMODE OUTPUT PROJECT MODEL- KAØØ1<br>EDIT- PLOT- CONVERT STOP | } Directives available in the root segment |
| 3 | DIR: MODEL ? | Calling directive segment MODEL |
| | SEGMENT MODEL | |
| | DRE DIRECTIVES (SYSTEM-DIR SEE $ ?) | |
| 4 | GROUND MODDAT HEADER ABSOR TRFORM<br>OUTPUT RETURN | } Directives afailable in segment MODEL |
| 5 | DIR: GROUND ? | Calling directive GROUND (handles ground control) |
| | DRE IN DIRECTIVE GROUND POSSIBLE PARAMETERS: | |
| 6 | REFSYS UNITS FORMAT MULT LIST<br>DELETE RESTORE | } Parameter available in directive GROUND |
| 7 | PAR: REFS, ? | Prompting PAR: internal level 2 (of parameters) |
| | DRE DIRECTIVE:GROUND PARAMETER:REFSYS VAL:DOUBLES ATTRIBUTES: | |
| 8 | METER MM FEET OVERWR ALLSTORE<br>FILE | } attributes available in parameter REFSYS |
| 9 | ATT: MTR( | Prompting ATT: internal level 3 (of attributes) |
| 10 | VAL: 1,1,1 1 2 2 2 2),$VERIFY; | Prompting VAL: internal level 4 (of values) |
| 11 | GROUND, REFSYS (1 1 1 1 2 2 2 2) = METER; | DRE system function $VERIFY; yields this line |
| 12 | PAR:; | semicolon entered: end-of-directive |
| 13 | 2 CONTROL POINTS INPUT 2 STORED | program reaction |
| 14 | DIR: | |

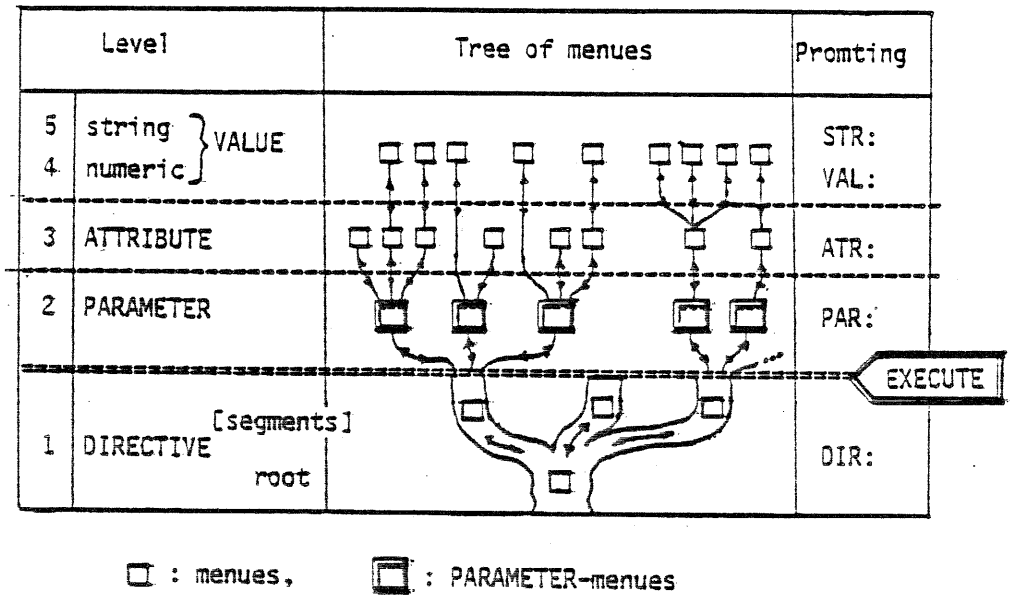
The process from table 1 performed in menue mode

Table 2

| N | SCREEN | Entries | References, comments |
|---|---|---|---|
| 1<br>2<br>3 | root segment: level 1 menue of directives with lines corresponding to (2) x) | 4 (for MODEL ) | |
| 4<br>5 | segment MODEL: level 1 menue of directives with lines corresponding to (4) | 1 (for GROUND) | table 5 |
| 6<br>7 | Directive GROUND: level 2 menue of parameters with lines corresponding to (6) | 1 (for REFSYS) | table 6 without the entries noted at parameter REFSYS |
| 8<br>9 | Directive GROUND, Parameter REFSYS: level 3 menue of attributes with lines corresponding to (8) | 1,V<br>input of values for METER | |
| 10<br><br><br>11 | Directive GROUND, Parameter REFSYS:<br>level 4 menue of values<br>level 3 menue of attributes | 1 1 1 1<br>2 2 2 2<br>$L (re-write menue of values)<br>$E (return to menue of attributes)<br>$E (return to menue of parameters) | Editor type input of values |
| 12<br><br>13 | level 2 menue of parameters<br>GROUND, REFSYS (1 1 1 1 2 2 2 2)=METER;<br>2 CONTROL POINTS INPUT 2 STORED | $E (EXECUTE) | table 6<br>- summary in syntax of DRE<br>- program reaction |
| 14 | level 1 menue of directives (identical with 4) | | table 5 |

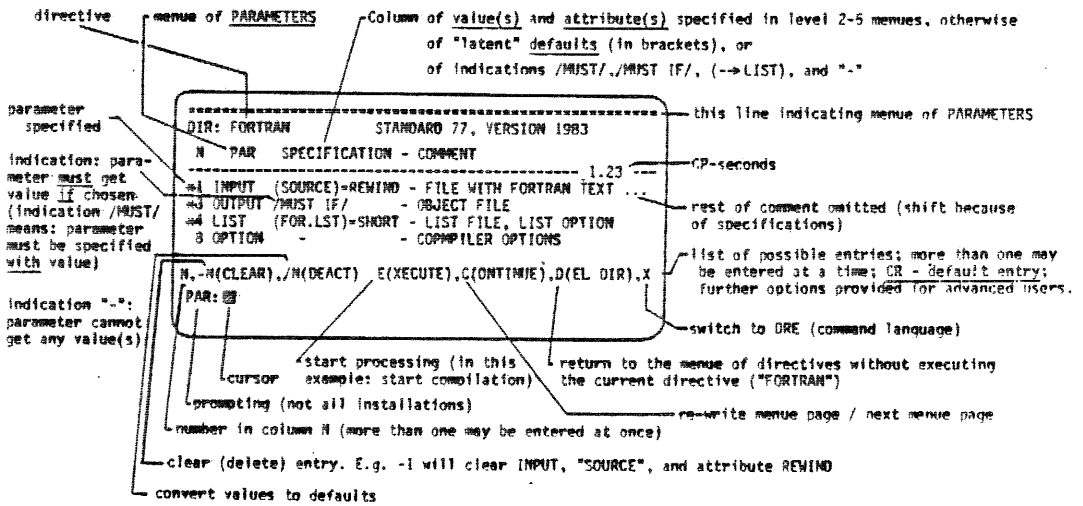
x) Numbers in parenthesis refer to column N of table 1

Table 3




Menue of parameters: explanation of details

Table 4




DRE-X: versions, options

Table 8

| Class of version | Available at this point |
|---|---|
| Languages | English, German |
| Computers | IBM, VAX, PDP-11, CDC, HARRIS, HP-1000/-3000, SIEMENS, UNIVAC |
| FORTRAN standard | FORTRAN 4, FORTRAN 5 |
| Operational systems, terminals | In some cases, differences in OS and/or terminal types must/can be taken into account |
| Options | Command language, menues, VERIFY function |

```
SEGMENT MODEL*   DATA INPUT

  N   DIR            COMMENT
-------------------------------------------------------------  0.68 ----
  1 GROUND       - GROUND CONTROL MANAGEMENT
  2 MODDAT       - DATA INPUT (WINPUT AND ALL OTHER FORMATS
                   EXCEPT FOR KA001)
  3 HEADER       - WINPUT HEADER MANAGEMENT (EDIT/CREATE -
                   A HEADER IS INEVITABLE FOR ABSOR)
  4 ABSOR        - ABSOLUTE ORIENTATION (DETERMINATION OF
                   TRANSFORMATION COEFFICITENTS ONLY)
  5 TRFORM       - DATA TRANSFORMATION (MODEL TO BE ADDED
                   TO DATAFILE)
-------------------------------------------------------------------------
  6 OUTPUT       - LIST,DISPL,MENUEDOK,REPORT=ON/OFF,REWIND
  7 RETURN*      - RETURN TO ROOT SEGMENT

N,C(ONTINUE),X
DRE    1.1  .006.DIR:1
```

Table 5

```
=========================================================================
DIR: GROUND            GROUND CONTROL TO BE INPUT,LISTED,EDITED

  N   PAR     SPECIFICATION -  COMMENT
-------------------------------------------------------------  2.68 ----
 *1 REFSYS   =FILE          - GROUND COORDINATES (REFERENCE SYSTEM)
  8 UNITS     -             - UNITS IN REFERENCE SYSTEM
*12 FORMAT   =FREE          - NECESSARY WHEN FROM 'FILE'
 14 MULT                    - FAKTORS TO REFERENCE SYSTEM
*15 LIST                    - LIST GROUND CONTROL
 20 DELETE                  - DE-ACTIVATE CONTROL POINT(S)
 22 RESTORE                 - RE-ACTIVATE CONTROL POINT(S)

N,-N(CLEAR),/N(DEACT)  E(XECUTE),C(ONTINUE),D(ELETE DIR),X
DRE    3.0  .006.PAR:
```

Table 6

```
DIR: DIRECT    PAR: PROJTYPE    TYPE OF PROJECTION TRANSFORMATION

  N   VAL:          COMMENT
-------------------------------------------------------------------------
  1          2     TYPE OF CENTRAL-PT-COORD.: MAP=1,IMAGE=2
  2      30.23     X-COORD. OF CENTRAL-PT.    (MM)
  3     23.456     Y-COORD. OF CENTRAL-PT.    (MM)
  4       1.23     SCALE FACTOR AT CENTRAL-PT. (1: ... )
  5       1.46     SCALE FACTOR AT PERIPH. PT. (1: ... )
  6      60.12     DISTANCE BETWEEN CENTRAL AND PERIPH. PT. (MM)

N,SL(IST),SE(ND)
DRE    1.3  .003.VAL:5
```

Table 7

Example: SORA-MP

Menue-type input
of values
for parameter PROJTYPE

Entry: 5 (for scale factor)

```
ENTER SCALE FACTOR AT PERIPH. PT. (1: ... )
DRE    1.3  .003.VAL:1.64
```

Entry: scale factor 1.64

```
DIR: DIRECT    PAR: PROJTYPE    TYPE OF PROJECTION TRANSFORMATION

  N   VAL:          COMMENT
-------------------------------------------------------------------------
  1          2     TYPE OF CENTRAL-PT-COORD.: MAP=1,IMAGE=2
  2      30.23     X-COORD. OF CENTRAL-PT.    (MM)
  3     23.456     Y-COORD. OF CENTRAL-PT.    (MM)
  4       1.23     SCALE FACTOR AT CENTRAL-PT. (1: ... )
  5       1.64     SCALE FACTOR AT PERIPH. PT. (1: ... )
  6      60.12     DISTANCE BETWEEN CENTRAL AND PERIPH. PT. (MM)

N,SL(IST),SE(ND)
DRE    1.6  .003.VAL:SE
```

defined in the calling routines (normally in DATA statements). User entries are returned to the calling program coded on array-parameters.

## 2.2 Menues

In describing DRE-X menues, references enclosed in paranthesis to column N of table 1 will be used. The tree of menue tables corresponding to different internal levels (1: directive, 2: parameter, etc. - see table 1) is shown in table 3. On each level, contents of menue tables correspond to those mnemonics listed following a ? entered in DRE. At any of points (1), (3), (5), (7), (9), (10), (12) and (14) of table 1 the possibility is given to switch over to menue techniques (or vice versa), using the $X entry. The process treated in table 1 performed in menue mode, involves major steps as shown in table 2.

DRE-X checks all entries, and puts them on internal buffers (immediate reactions to semantic errors are provided by periodically calling external subroutines to be furnished for each directive). Execution is delayed until the $E(XECUTE) entry under the menue of parameters (table 4). Up to this point, all entries can be changed at will without consequences. Menues of parameters are used, as well, to sumarize entries made in higher level menues. To obtain a summary of entries in syntax of DRE, one can switch over to DRE, call the $VERIFY; system function, and switch back to menues:

```
X$VER;$X
 |  |  └ switching back to menues
 |  └ entry for DRE
 └ switching over from menues to DRE
```

Before execution, $VERIFY; is activated by DRE-X to provide a summary of active input in a protocol file (menues will be written onto this file only when specifically requested).

There are four different ways to input values of parameters in DRE-X. The first two of these are of the type dialogue, the third one serves to input and to edit periodic lists of values (e.g. a list of ground control coordinates), and the fourth way provides comment lines to each element of a list of values. An example of this last ("menue-type") data input is taken from program SORA-MP /5/ (table 7). - Ways of value input can be manipulated in accordance with attributes chosen, or any other characteristics of the actual state of processing.

In DRE-X, user entries are buffered for all levels separately. This allows the user to pre-program his actions (e.g. to choose at once all parameters he wants to deal with). Invisible "tricks" make work of advanced users easier. These capabilities, and switching back and forth between menues and command language, enable the user to jump over entire series of menues.

*

Comments in menues, control information on structuring menue tables, and, as an option, FORTRAN statements to check entries sementically, are passed from the calling program to DRE-X via external subroutines. With DRE-X, a facility program GENME is provided, generating such external subroutines on the basis of data containing the minimum on necessary information in a systematized, relatively convenient form.

The programmer is given capabilities to activate any menue table on the screen. This enables him, on the one hand, to create more guidance if necessary, and on the other hand, to handle errors in control input elegantly. This same capability can be applied to use the editor type input of values as an interactive editor for periodic lists of values within program modules.

Lack of space prevents us from describing here the three ways of handling defaults in DRE-X.

## 3. CONCLUDING REMARKS

As noted earlier, DRE-X is a FORTRAN module to be included in user programs and program systems /2, 3, 4/. The source text of DRE-X is maintained as specially formatted text files containing all different versions and options (table 8). These text files have to be processed by a pre-compiler program (controlled itself by DRE), the output of which is the FORTRAN source to be compiled.

Experience with users confirm expectations (chapter 1). Beginners, such as students using SCOP, prefer menues. With getting better acquainted with the program, menue-type input is becoming more and more tedious, and in combining menues with command language the last one gaines space progressively. At a communication rate of 1.2oo bauds, effectiveness of command language input is greatly better than that of the menue techniques. However, when explaining program capabilities to students or customers, menues prove to be highly advantageous. Advanced users switch over to menues when defining some complicated or seldomly applied directives. The introduction of menues helped advanced users in activating forgotten program options.

## REFERENCES

/1/ Kager, H.: Das interaktive Programmsystem ORIENT im Einsatz. Presented paper, 14th congress of ISP, Commission V, Hamburg 1980

/2/ Kraus, K., Aßmus, E., Köstli, A., Molnar, L., Wild, E.: Digital Elevation Models: Users' Aspects. Works of Institute for Photogrammetry, Stuttgart University, Vol. 8, 1982

/3/ Molnar, L., Aßmus, E., Köstli, A., Wild, E.: Digital Elevation Models: Informatics' Aspects. ISPRS, Symposium of Commission III, Helsinki, 1982

/4/ Haitzmann, H., Kraus, K., Loitsch, J.: A Data Base Towards the Digitally Controlled Production of Orthophotos. Presented paper, 14th congress of ISP, Commission IV, Hamburg 1980

/5/ Jansa, J., Vozikis, E.: The Program SORA-MP (working title). Presented paper, 15th congress of ISPRS, Commission IV, Rio de Janeiro, 1984