MEN - Menu and Dialog Software
Version 1.3

H. Kroiss

IPP III / 192 Juli 1993

MAX-PLANCK-INSTITUT FÜR PLASMAPHYSIK
85748 GARCHING BEI MÜNCHEN
IPP III / 192

Abstract

The MEN Menu and Dialog Software is an easy way to implement dialog and menu techniques into FORTRAN user programs.

The MEN Software consists of four parts of support:

I. MEN GENERATE - Menu File and Program Generator
II. MEN DIALOG - Standard Dialog Program
III. MEN SHELL - UNIX shell dialog functions
IV. MEN FORTRAN - FORTRAN Programming Routines

All Routines are designed for use on standard VT200/300/400 compatible terminals or windows and running on all UNIX and VHS operating systems.

The MEN software package should only be used by programs, for which any WINDOW software (mouse driven) is too difficult to implement.

Die nachstehende Arbeit wurde im Rahmen des Vertrages zwischen dem Max-Planck-Institut für Plasmaphysik und der Europäischen Atomgemeinschaft über die Zusammenarbeit auf dem Gebiete der Plasmaphysik durchgeführt.
M E N - Menu and Dialog Software

Version 1.3

H. Kroiss 4.7.1993

Abstract

The MEN Menu and Dialog Software is an easy way to implement dialog and menu techniques into FORTRAN user programs.

The MEN Software consists of four parts of support:

I. MEN_GENERATE - Menu File and Program Generator
II. MEN_DIALOG - Standard Dialog Program
III. MEN_SHELL - UNIX shell dialog functions
IV. MEN_FORTRAN - FORTRAN Programming Routines

All Routines are designed for use on standard VT200/300/400 compatible terminals or windows and running on all UNIX and VMS operating systems.

The MEN software package should only be used by programs, for which any WINDOW software (mouse driven) is too difficult to implement.
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Overview</td>
<td>1</td>
</tr>
<tr>
<td>B. MEN_GENERATE - Menu File and Program Generator</td>
<td>9</td>
</tr>
<tr>
<td>C. MEN_DIALOG - Standard Dialog Program</td>
<td>18</td>
</tr>
<tr>
<td>D. MEN_SHELL - Programs for UNIX shell script usage</td>
<td></td>
</tr>
<tr>
<td>D.1 Program &quot;men_get_i&quot;</td>
<td>22</td>
</tr>
<tr>
<td>D.2 Program &quot;men_get_r&quot;</td>
<td>23</td>
</tr>
<tr>
<td>D.3 Program &quot;men_get_c&quot;</td>
<td>24</td>
</tr>
<tr>
<td>D.4 Program &quot;men_set_i&quot;</td>
<td>25</td>
</tr>
<tr>
<td>D.5 Program &quot;men_set_r&quot;</td>
<td>26</td>
</tr>
<tr>
<td>D.6 Program &quot;men_set_c&quot;</td>
<td>27</td>
</tr>
<tr>
<td>E. MEN_CONVERT - convert menu descriptor files</td>
<td>28</td>
</tr>
<tr>
<td>F. MEN_FORTRAN - Programming Interface Routines</td>
<td></td>
</tr>
<tr>
<td>F.1 Basic menu Functions</td>
<td></td>
</tr>
<tr>
<td>F.1.1 Routine &quot;MEN_OPEN&quot;</td>
<td>30</td>
</tr>
<tr>
<td>F.1.2 Routine &quot;MEN_DEFINE&quot;</td>
<td>31</td>
</tr>
<tr>
<td>F.1.3 Routine &quot;MEN_INFO&quot;</td>
<td>33</td>
</tr>
<tr>
<td>F.1.4 Routine &quot;MEN_MODIFY&quot;</td>
<td>35</td>
</tr>
<tr>
<td>F.1.5 Routine &quot;MEN_SET&quot;</td>
<td>37</td>
</tr>
<tr>
<td>F.1.6 Routine &quot;MEN_GET&quot;</td>
<td>39</td>
</tr>
<tr>
<td>F.1.7 Routine &quot;MEN_READ&quot;</td>
<td>41</td>
</tr>
<tr>
<td>F.1.8 Routine &quot;MEN_WRITE&quot;</td>
<td>42</td>
</tr>
<tr>
<td>F.1.9 Routine &quot;MEN_SAVE&quot;</td>
<td>44</td>
</tr>
<tr>
<td>F.2 Advanced menu routines</td>
<td></td>
</tr>
<tr>
<td>F.2.1 Routine &quot;MEN_REFRESH&quot;</td>
<td>45</td>
</tr>
<tr>
<td>F.2.2 Routine &quot;MEN_SETID&quot;</td>
<td>46</td>
</tr>
<tr>
<td>F.2.3 Routine &quot;MEN_GETID&quot;</td>
<td>47</td>
</tr>
<tr>
<td>F.2.4 Routine &quot;MEN_ID&quot;</td>
<td>48</td>
</tr>
<tr>
<td>F.2.5 Routine &quot;MEN_SETBGRD&quot;</td>
<td>49</td>
</tr>
<tr>
<td>F.2.6 Routine &quot;MEN_GETBGRD&quot;</td>
<td>51</td>
</tr>
<tr>
<td>F.2.7 Routine &quot;MEN_WRITEBGRD&quot;</td>
<td>52</td>
</tr>
<tr>
<td>F.2.8 Routine &quot;MEN_SETHELP&quot;</td>
<td>53</td>
</tr>
<tr>
<td>F.2.9 Routine &quot;MEN_GETHELP&quot;</td>
<td>54</td>
</tr>
<tr>
<td>F.2.10 Routine &quot;MEN_SETRANGEI&quot;</td>
<td>55</td>
</tr>
<tr>
<td>F.2.11 Routine &quot;MEN_GETRANGEI&quot;</td>
<td>57</td>
</tr>
<tr>
<td>F.2.12 Routine &quot;MEN_SETRANGER&quot;</td>
<td>58</td>
</tr>
<tr>
<td>F.2.13 Routine &quot;MEN_GETRANGER&quot;</td>
<td>59</td>
</tr>
<tr>
<td>F.2.14 Routine &quot;MEN_CLEAR&quot;</td>
<td>60</td>
</tr>
<tr>
<td>F.2.15 Routine &quot;MEN_MOVE&quot;</td>
<td>61</td>
</tr>
<tr>
<td>F.2.16 Routine &quot;MEN_NEXT&quot;</td>
<td>62</td>
</tr>
<tr>
<td>F.2.17 Routine &quot;MEN_SETTOGGLE&quot;</td>
<td>63</td>
</tr>
<tr>
<td>F.2.18 Routine &quot;MEN_TOGGLE&quot;</td>
<td>65</td>
</tr>
<tr>
<td>F.2.19 Routine &quot;MEN_SAVRES&quot;</td>
<td>67</td>
</tr>
</tbody>
</table>

**Note:** The contents are a list of sections and routines with page numbers.
## CONTENTS (cont.)

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>F.3</td>
<td>Routines to perform simple dialog functions</td>
<td></td>
</tr>
<tr>
<td>F.3.1</td>
<td>Routine &quot;MEN_GETI&quot;</td>
<td>71</td>
</tr>
<tr>
<td>F.3.2</td>
<td>Routine &quot;MEN_GETR&quot;</td>
<td>73</td>
</tr>
<tr>
<td>F.3.3</td>
<td>Routine &quot;MEN_GETC&quot;</td>
<td>75</td>
</tr>
<tr>
<td>F.3.4</td>
<td>Routine &quot;MEN_SETI&quot;</td>
<td>77</td>
</tr>
<tr>
<td>F.3.5</td>
<td>Routine &quot;MEN_SETR&quot;</td>
<td>79</td>
</tr>
<tr>
<td>F.3.6</td>
<td>Routine &quot;MEN_SETC&quot;</td>
<td>81</td>
</tr>
<tr>
<td>F.4</td>
<td>Routines to perform terminal I/O functions</td>
<td></td>
</tr>
<tr>
<td>F.4.1</td>
<td>Routine &quot;VT_CLEARSCREEN&quot;</td>
<td>83</td>
</tr>
<tr>
<td>F.4.2</td>
<td>Routine &quot;VT_RESETSCREEN&quot;</td>
<td>84</td>
</tr>
<tr>
<td>F.4.3</td>
<td>Routine &quot;VT_SETREGION&quot;</td>
<td>85</td>
</tr>
<tr>
<td>F.4.4</td>
<td>Routine &quot;VT_CLEARREGION&quot;</td>
<td>86</td>
</tr>
<tr>
<td>F.4.5</td>
<td>Routine &quot;VT_SETCURSOR&quot;</td>
<td>87</td>
</tr>
<tr>
<td>F.4.6</td>
<td>Routine &quot;VT_READKEY&quot;</td>
<td>88</td>
</tr>
<tr>
<td>F.4.7</td>
<td>Routine &quot;VT_SOFTKEY&quot;</td>
<td>90</td>
</tr>
<tr>
<td>F.4.8</td>
<td>Routine &quot;VT_READSTRING&quot;</td>
<td>92</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.</td>
<td>SUMMARY of MEN Fortran Interface Routines</td>
<td>94</td>
</tr>
<tr>
<td>H.</td>
<td>Example programs for MEN subroutine package</td>
<td></td>
</tr>
<tr>
<td>H.1</td>
<td><code>men_example1.f</code> - Using PRINT for background</td>
<td>98</td>
</tr>
<tr>
<td>H.2</td>
<td><code>men_example2.f</code> - Using MEN_SETBGRD for background</td>
<td>100</td>
</tr>
<tr>
<td>H.3</td>
<td><code>men_example3.f</code> - Using a menu descriptor file</td>
<td>102</td>
</tr>
<tr>
<td>H.4</td>
<td><code>men_example4.f</code> - Performing noninteractive dialog</td>
<td>104</td>
</tr>
<tr>
<td>H.5</td>
<td><code>men_example5.f</code> - Calculator using softkeys</td>
<td>105</td>
</tr>
<tr>
<td>H.6</td>
<td><code>men_example6.f</code> - Using MEN_TOGGLE on input fields</td>
<td>108</td>
</tr>
<tr>
<td>H.7</td>
<td><code>men_example7.f</code> - Simple parameter input</td>
<td>110</td>
</tr>
<tr>
<td>H.8</td>
<td><code>men_example8.f</code> - Shell scripts using menu files</td>
<td>110</td>
</tr>
</tbody>
</table>
Overview

The Menu Software Package is a very easy way to create and modify screenfields, manipulate the terminal (e.g. CLEAR), define softkeys and store dialog information into files. This information may then be used to get dialog values into permanently running (noninteractive) programs. A further important advantage of the M E N software is the independence of computer operating systems and I/O devices.

A.1 What is a menu?

In this context a menu means a set of structured information about I/O operations. This information consists of two basic types:

1. Display-oriented information (like screen position, display attributes and background)
2. Value-oriented information (like actual values, format, range and help)

Each menu will be displayed as a frame of up to 24 lines with up to 80 characters (rows) each.

Menu-Display:

<table>
<thead>
<tr>
<th>LINE</th>
<th>Menu Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Background-Text Dynamic Field</td>
</tr>
<tr>
<td>.</td>
<td>V</td>
</tr>
<tr>
<td>.</td>
<td>This is Factor 1: 1234.E3</td>
</tr>
<tr>
<td>24</td>
<td></td>
</tr>
</tbody>
</table>

The display of a menu consists of two distinct parts of screen information:

1. Static display: That means text information, which will be not modified during a menu session. This information will be called 'Background' in the further.

2. Dynamic display: This is information (called "Fields"), which may be dynamically modified during a menu session.

For each menu up to 99 dynamic fields may be specified. If more than 99 fields are required, a further menu (or sub-menu) should be created. Note, that the user program may only work with one menu at a time.
A.2 Dynamic menu fields

To specify a dynamic menu field the following definitions are necessary:

1. An identifier (ID), which specifies the number of menu field (range 1..99)

2. The name of the menu field

3. The line number on screen (1..24)

4. The row number on screen (1..80)

5. The length of menu field (1..80)

6. The format of menu field:
   'C' for CHARACTER*n field
   'I' for INTEGER field
   'R' for REAL field (floating point)
   'F' for REAL field (fixed point)

7. The I/O direction:
   'I' for Input
   'O' for Output
   'B' for Bidirectional

8. The field attributes:
   'N' for normal display
   'I' for increased intensity display
   'B' for blinking display
   'R' for reverse display
   'U' for underlining display

9. The data buffer for the menu field values

10. A character line for help messages

11. The range (LOW/HIGH) specification for INTEGER and REAL values

A.3 Menu Mask File (.msk)

A menu mask file will be needed if you want to generate a menu descriptor file or a FORTRAN program with the MEN_GENERATE utility. In the menu mask file the menu background and the position of desired dynamic fields should be specified. It is recommended that menu mask files be given the file extension ".*msk".
A. Overview

To create a menu mask file you may use any editor program and insert the desired text. The following syntactical rules are to obey:

1. Only the first 24 lines will be used by the MEN_GENERATE program.

2. Dynamic fields may be defined by the "$" character. The number of "$" characters specify the maximum length of the menu field.

3. Control characters are not allowed in the background text (exception: TAB's)

Example of a simple menu mask file:

$ edt men_example.msk

```
This is a menu mask example

Factor-1: $$$$$$$  Factor-2: $$$$$$$$$
Date: $$$$$$$$$  Time: $$$$$$$$
Note: $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Note: It is strongly recommended to define only 23 lines, because the 24th line will be often used for softkey input purposes (for example: MEN_DIALOG).

A.4 Menu Descriptor File (.men)

The menu descriptor file contains all information about a specified menu, including field values, background and display parameters. The " .men" files are binary files and may be displayed and modified by the standard MEN_DIALOG program, by the user program or shell scripts (csh).

A menu descriptor file may be created at the first time either by the MEN_GENERATE program or by an user FORTRAN program (during a MEN_SAVE call).

A menu descriptor file is needed, if the information about a specified menu should be saved on disk for later usage (last state) or for noninteractive procedures.
A.5 FORTRAN programming interface routines

The FORTRAN menu subroutines are divided into 4 classes of functions:

- Basic menu functions:
  - This is a small set of most often used routines.

- Advanced menu functions:
  - Used only for special purposes.

- Non-interactive menu functions:
  - Used only for menufile interactions.

- Terminal I/O functions (Prefix "VT "):  
  - Used to perform operations on terminal screen.

All menu operations will be performed over an internal COMMON, called menu database, on which the actual state of a given menu is stored. Most of the MEN routines work with this menu database and not directly with the terminal screen. For example, to enter an INTEGER value from the screen and work with it, the following two steps are necessary:

1. Read value from screen into menu database:
   CALL MEN_READ (ID)

2. Assign the value to an own variable (MYINT):
   CALL MEN_GET (ID,MYINT,RVAL,IVAL)

Naming conventions:

- MEN routines, performing Terminal I/O functions, are named "MEN_READxx" or "MEN_WRITExx".
- Assignment routines are named "MEN_GETxx" or "MEN_SETxx".
- Generally routines with prefix "MEN_" need always a defined menu database.
- The routines with prefix "VT ", are independent of menu structures and therefore may be also used for general purposes.

A.6 Non-interactive dialog with menu files

In many cases the user want to get input parameters from a file instead of terminal input. Batch jobs and permanently running programs (like automatic shot analysis) does not allow an interactive terminal input, because the program execution will be interrupted during this kind of input. For these applications the MEN software offers very easy routines to get (and set) values from (into) an existing menu file. Therefore the user must not program own structured file I/O routines by FORTRAN OPEN,READ,WRITE,FORMAT and CLOSE statements.
For example, to get an INTEGER value from an existing menu file, only two statements are necessary (see MEN_EXAMPLE4):

```fortran
CALL MEN_OPEN (FNAME, ERR)
CALL MEN_GETI (ID, IVAL, STAT)
```

Furthermore shell scripts may also use parameters contained in a menu descriptor file in a non-interactive fashion. (see MEN_EXAMPLE8)

### A.7 Error Handling

The MEN_FORTRAN subroutines have the error handling included and write an appropriate error message on terminal. After an error has occurred the program execution will be stopped, because all possible errors are programming errors and therefore it makes no sense to continue the program. Exceptions of this rule: MEN_OPEN, MEN_ID and all non-interactive routines (MEN_GETI, MEN_SETI etc.).

### A.8 Menu Applications

The most important applications should be described below:

- **Application 1:**
  
  In many cases the user wants to get a lot of parameter values by terminal input. The number of parameters and the format of the values is fixed, that means they must not be changed during program execution.

  In this case the user should
  - create a menu mask file with any text editor,
  - call MEN_GENERATE to specify menu fields and
  - generate a menu descriptor file.

  After these steps the user may work with the existing menu descriptor file by calling MEN_OPEN (see MEN_EXAMPLE3). If the user does not want to operate with a menu descriptor file, he should create a FORTRAN subroutine with program MEN_GENERATE, which generates the desired menu and may be included (by call) in the user program.

- **Application 2:**

  A batch or permanently running job (daemon) needs parameter values from a file. The modification or input of the values should be done by another program.

  In this case it is recommended to
  - create a menu mask file with any text editor,
  - call MEN_GENERATE to specify menu fields,
  - generate a menu descriptor file and
  - include the non-interactive MEN routines (MEN_GETI, MEN_GETR etc.) in the job (see MEN_EXAMPLE4).

  To modify parameter values the standard program MEN_DIALOG may be used.
Application 3:

Procedures which want to create a menu at runtime or have program conditions on which the menu structure should be dynamically changed, must use the FORTRAN menu subroutine package, which allow all desired changes (DEFINE,CLEAR, MOVE, MODIFY) on an existing menu at runtime. In this case the creation of a menu descriptor file makes no sense, but the MEN GENERATE function 'PROGRAM' may help to implement the desired function.

Application 4:

A system administrator want to perform dialog with a parameter file by using shell scripts. In this case the interactive dialog part may be done with the MEN_DIALOG program (Display and modification of current values). The shell script may then read and write these values by appropriate MEN programs (see MEN_EXAMPLES).

A.9 Implementation Notes

The MEN dialog and menu software is written in standard FORTRAN77/90 (95%) and "C" (5%). Therefore it is very easy to implement the software on new UNIX systems (currently running under VAX/VMS, ULTRIX, OSF1).

The MEN software consists of:

1. MEN_FORTTRAN - Fortran subroutine package
2. MEN_GENERATE - Menu and program generator program
3. MEN_DIALOG - Menu dialog program
4. MEN_SHELL - Menu utilities for shell scripts
5. MEN documentation set:
   - Online help (GOPHER)
   - Manual (MEN MANUAL.TXT)
   - Examples: FORTRAN-Programs and menu descriptor files

The interactive part of the MEN software is currently supported on the following I/O devices:

1. Terminals: DEC VT200/VT300/VT400 series
   TEKTRONIX 4107/4207

2. X-WINDOW devices: DEC VXT2000 series
   TEKTRONIX XP358

3. Workstations: DEC MIPS stations
   DEC VAX stations (DECwindow, VWS)
   DEC ALPHA stations

4. MacIntosh
Software Installation:

a) VAX/VMS:
   udslib:men77.0lb - MEN_FORTRAN subroutines
   udsexe: - executable programs
   uds/hlp: - Example programs

b) UNIX:
   /usr/lib/libmen77.a - FORTRAN77 subroutines
   /usr/lib/libmen90.a - FORTRAN90 subroutines
   /usr/bin - Executable programs
   /usr/examples - Example programs
   /usr/doc - MEN documentation

Program Linkage:

a) VAX/VMS:
   $ LINK <program>
   program will be linked with default libraries.

B) UNIX:
   $ f77 <program>.o -lmen77 -o <program>

A.10 Operating Instructions
-------------------------

For usage of the MEN_GENERATE and MEN_DIALOG programs
it is necessary to understand the concepts of "Function
Keys" and "Data Fields".

A.10.1 Function Keys

   o Keyword Functions

   At the bottom of each menu page, a line containing 8
   keywords will appear. They correspond to 8 central
   function keys at the top of your terminal keyboard as
   follows:

   [<kwdr>] [<kwdr>] [<kwdr>] [<kwdr>] [<kwdr>] [<kwdr>] [<kwdr>]
   [ F7 ] [ F8 ] [ F9 ] [ F10 ] [ F11 ] [ F12 ] [ F13 ] [ F14 ]

   NOTE: The leftmost key, F6, is not used.

   Pressing one of these function keys will cause programs
   to perform a function indicated by the associated
   keyword <kwdr>.
   Depending on the menu page and the previous functions
   selected, the keywords displayed - and the possible
   functions - may change.
o Arrow Keys

In addition to the keyword functions, the arrow keys may be used to move the current field position (see section A.10.2) to another field. Usually they are marked clearly as:

```
<-    ^    v    ->
left  up   down  right
```

Pressing an arrow key moves the current field to the next field in the indicated general direction.

A.10.2 Data Fields

Certain areas on the displayed menu page contain data which may change during a program session. These areas are known as "data fields". Data fields can be "entry" or "display" fields. Display fields contain information which can be changed by the program without rewriting the entire page. Entry fields can also be modified by the user, usually by typing the new value directly into the field, and in some cases by pressing a special "TOGGLE" function key to step through a set of predefined values.

To modify an entry field, it must first be selected as the "current field". The current field is displayed in inverted form (dark text on a light background). Pressing an arrow key selects the next field in the general direction of the pressed key as the new current field. The contents of the current field may be changed in two ways:

If "TOGGLE" appears in the function keyword list, pressing the associated key will choose the next value from a predetermined set of possible values for the current field. If the keyword "ENTER" appears, pressing the corresponding key will allow you to type in directly a new value for the current field.
B. MEN_GENERATE - Menu File and Program Generator

Version 1.3

H. Kroiss 14.8.93

Abstract

MEN GENERATE is an utility program to create and modify a menu file in a very easy way. The menu file may be used by FORTRAN programs to perform interactive or non-interactive dialog and menu functions. Furthermore a FORTRAN subroutine may be generated which may then be included in own programs to create the menu function desired.

CONTENTS

B.1 Introduction

B.1.1 Starting MEN_GENERATE
B.1.2 Interactive Mode Menu Pages
B.1.3 Error Messages
B.1.4 Help Messages

B.2 CREATE a new menu descriptor file

B.2.1 SPECIFY a dynamic menu field
B.2.2 SHOW menu fields already defined
B.2.3 BUILD a new menu descriptor file or FORTRAN subroutine

B.3 MODIFY an existing menu descriptor file

B.3.1 CHANGE an existing dynamic menu field
B.3.2 INSERT a new dynamic menu field
B. MEN_GENERATE - Menu File and Program Generator

B.1 Introduction

MEN_GENERATE - the Menu File and Program Generator- is a program available on all VAX/VMS and UNIX systems. It may be run from any terminal or X-Window display supporting VT200/300/400 mode.

MEN_GENERATE may be used to

- create a new menu descriptor file;
- display an existing menu descriptor file;
- modify an existing menu descriptor file;
- modify the parameters of a single dynamic field;
- generate a FORTRAN subroutine performing all operations to create and display the menu desired.

The program MEN_GENERATE is largely self-explanatory by means of function keys and help messages, but it is nevertheless recommended first to read chapter A (Overview) for better understanding of the operations performed by MEN_GENERATE.

B.1.1 Starting MEN_GENERATE

MEN_GENERATE may be started as follows:

* Log on to the desired terminal
* After the $ prompt, type: men_generate

B.1.2 Interactive Menu Pages

When started, MEN_GENERATE will display the first of a series of menu pages on the screen. Each menu page displays a differing aspect of menu modification actions, and allows you to press function keys and/or type in parameters to control what will be displayed on the screen.

The following differing main menu pages are available:

CREATE  Create a menu descriptor file or FORTRAN subroutine for the first time. For this purpose a menu mask file should already exist. When pressing the CREATE key, the name of this mask file will be requested.

MODIFY  Allows you to display and modify an existing menu descriptor file. When pressing the MODIFY key, the name of the desired menu descriptor file will be requested.

EXIT  Stop MEN_GENERATE and exit to VMS/UNIX level.

B.1.3 Error Messages

Occasionally you may make a mistake in operating the MEN_GENERATE program, in particular if you attempt to find a file which is not available. In such cases, a line of text will appear straight above the function keyword line explaining what error occurred.
B. MEN_GENERATE — Menu File and Program Generator

B.1.4 Help messages

Depending on the current menu page and the menu field selected a help message will appear straight above the function keyword line.

B.2 CREATE — Create a new menu descriptor file or FORTRAN subroutine

This menu page will be entered immediately after having issued the "CREATE" command and input of an existing menu mask file. The menu mask selected will be displayed on the screen.

While in the CREATE menu, you may

- Specify the available dynamic menu fields selected by using the cursor keys;
- Show menu fields already defined;
- Build a menu descriptor file or FORTRAN subroutine;
- Stop MEN_GENERATE operations and exit to VMS/UNIX level.

Function Keys:

^ (Arrow Keys)

< - | - > Change the current menu data field by moving the current field to a differing number.

SPECIFY [*] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]

Enter the "SPECIFY" menu page to define the parameters needed for a dynamic menu field.

SHOW [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]

Display dynamic menu fields already defined with the corresponding field-ID and field-position.

BUILD [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]

Build a new menu descriptor file or a FORTRAN subroutine.

EXIT [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]

Exit to VMS/UNIX level without modifications.

When a dynamic field have been already specified the "$" characters will be substituted by the field's default value. It is not necessary to specify all fields available, but non-specified fields will not be included into FORTRAN programs or menu descriptor files! The SPECIFY function may be called more than once to redefine a menu field. After specification of all dynamic fields required, the BUILD function may be called to generate menu file or FORTRAN program.
B.2.1 SPECIFY - Specify a dynamic menu field for the first time

After entering the SPECIFY key, the field selected is defined by initial values. Now you may change one or more parameters of selected field in any order. The SPECIFY menu page shows the field number previously selected, the corresponding field screen position (Line, Row and Length) and the associated background line. The following parameters may be modified:

- **Field name:**
  This name may be used to reference (select) the field by name instead of ID-Number.
  
  Used function key: ENTER

- **Field format:**
  Specify menu field format desired:

  a) INTEGER - for INTEGER*4 field
  b) REAL fixed - for REAL field
  c) REAL floating - for REAL field
  d) CHARACTER*n - for CHARACTER field

  Used function key: TOGGLE

- **Field I/O direction:**
  Specify allowed I/O operation on terminal screen:

  a) Input - only input allowed on screen field
  b) Output - only output allowed on screen field
  c) Bidirect - input and output allowed on screen field

  Used function key: TOGGLE

- **Field attribute:**
  Specify screen attribute desired for screen display:

  a) None - for normal display
  b) Reverse - for reverse display
  c) Intensity - for increased intensity display
  d) Underline - for underlining display
  e) Blinking - for blinking display

  Used function key: TOGGLE

- **Help for current menu field:**
  A special help message may be defined for each menu field. (up to 72 characters)

  Used function key: ENTER

- **Default value for current menu field:**
  This value will be used as initial value on menu programs. Note: The range of the default value will be not checked on legality by MEN_GENERATE!

  Used function key: ENTER
o Field range (LOW):
  This value may only be set on REAL or INTEGER fields and
  specify the lowest value allowed in this field.

o Field range (HIGH):
  This value may only be set in REAL or INTEGER fields and
  specify the highest value allowed in this field.
  Note: Range checking is only enabled in case that the
  HIGH value exceeds the LOW value!

Function Keys:

  ^ (Arrow Keys)
  Move the current menu input field to the next
  field in the direction of the arrow key pressed.

ENTER [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
Modify the value of the current field by typing
in the value directly. (Does only appear when the
current field may be modified directly).

TOGGLE [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
Change the value of the current field to the next
item in a limited list of choices. (Does only appear
when the current field has a toggle-set of values).

RETURN [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
Return to previous menu page.

In case that the user does not want to specify the field by now,
but wants to include (reserve) the field definition for later
usage, the RETURN function key should be pressed without
modification on field parameters.

B.2.2 SHOW - Display menu fields already defined

The SHOW function may be used to get information about
the dynamic fields currently defined, the corresponding
field screen position and the field identification number.

The fields already defined are displayed in 'Blinking' mode
and show the specified field ID-Number.

The SHOW function should be used before changing a field
ID-Number or inserting a new menu field to avoid conflicts
with existing menu fields!

Function Keys:

CONTINUE [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
Return to the previous menu page.
B.2.3 BUILD - Create a new menu descriptor file or FORTRAN subroutine

The BUILD function may be called from the menu pages CREATE and MODIFY and allows the user to:

1. Create a new menu descriptor file
2. Generate a FORTRAN subroutine for all menu functions and definitions required
3. Perform action 1 and 2 together

In case that not all available dynamic fields were defined, a WARNING message will be appear on terminal screen. You may ignore the warning but undefined fields will not be included into menu file or FORTRAN subroutine output. If you forgot to define a field, then press RETURN function key!

For all BUILD functions possible the name of an appropriate diskfile must be entered. The file names should be defined by the standard file name conventions:

  o <name>.f - for FORTRAN programs
  o <name>.men - for menu descriptor files

NOTE: If you want to create a FORTRAN subroutine with a filename already existing on disk, the function 'Program' will fail (existing file not superseded)!

Function Keys:

**MenuFile**

[*] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
Stores all menu definitions performed on the current MEN_GENERATE session into specified menu descriptor file. MEN_GENERATE will then be stopped.

**Program**

[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
Creates a FORTRAN subroutine with all menu functions and definitions made by the current MEN_GENERATE session. MEN_GENERATE will then be stopped.

**BOTH**

[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
Execute 'MenuFile' and 'Program' function together. MEN_GENERATE will then be stopped.

**RETURN**

[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
Return to previous menu page.

**EXIT**

[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
Exit to VMS/UNIX level without modifications.
B.3 MODIFY - Modify an existing menu descriptor file

This menu page is entered immediately after issuing the
"MODIFY" command and input of an existing menu descriptor
file. The content of the menu file selected will be shown
on screen.

While in the MODIFY menu, the user may

- Redefine the dynamic menu fields available selected by
  using the cursor keys;
- Show menu fields already defined;
- Insert a new dynamic menu field into current menu;
- Build a new menu descriptor file or FORTRAN subroutine;
- Stop MEN_GENERATE operations and exit to VMS/UNIX level.

Function Keys:

- (Arrow Keys)
- Change the current menu data field by moving the
  current field to a different number.

CHANGE [*] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
Enter the "CHANGE" menu page to redefine the
parameters for a given dynamic menu field.

SHOW [ ] [*] [ ] [ ] [ ] [ ] [ ] [ ]
Display dynamic menu fields already defined with
the corresponding field-ID and field-position.

INSERT [ ] [ ] [ ] [ ] [ ] [ ] [ ]
Insert (Add) a new dynamic menu field to the
currently used menu descriptor file.

BUILD [ ] [ ] [ ] [ ] [ ] [ ] [ ]
Build a new menu descriptor file or a FORTRAN
subroutine.

EXIT [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
Exit to VMS/UNIX level without modifications.

The CHANGE function may be called more than once to redefine
a menu field.
When all modifications desired have been done, the BUILD
function may be called to generate menu file or a FORTRAN
program.

NOTE: The SHOW function is explained in Chapter B.2.2;
for BUILD function in detail see Chapter B.2.3.
B.3.1 CHANGE — Change an existing dynamic menu field

After entering the CHANGE key, all definitions and values of the selected field will be displayed in the CHANGE menu page. Now you may redefine one or more parameters of the selected field in any order. The function of the CHANGE key is very similar to the MODIFY key (see Chapter B.2.1), but the current field number, the corresponding field screen position (Line, Row and Length) and the associated background line may also be modified in addition.

Modifications may be made on the following parameters:

- Field number:
The field number is used to reference a menu field by menu programs. The new field number entered should be free for use (not yet defined), because all definitions of this field will be overwritten! Therefore be careful while changing a field number. (Note: use the SHOW function to see which fields are already defined). The legal range for field numbers is 1..99.

  Used function key: ENTER

- Field line:
Change the screen line number of current field.
The legal range of line numbers is 1..24.

  Used function key: ENTER

- Field row:
Change the screen row number of current field.
The legal range of row numbers is 1..80.

  Used function key: ENTER

- Field length:
Change the display length of current field.
The legal range of field length is 1..80 depending on the row position of the field (LENGTH <= 80-ROW).

  Used function key: ENTER

- Field background:
Change the background line of current field.

  Used function key: ENTER

- Other field parameters:
  All parameters described in MODIFY function (see Chapter B.2.1) may also be changed.
Function Keys:

(Arrow Keys)

<-|->    Move the current menu input field to the next field in the direction of the arrow key pressed.

v  ENTER    [*][ ][ ][ ] [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
Version 1.3

H. Kroiss 14.8.93

Abstract

MEN_DIALOG is used to display and modify menu files in a simple way. MEN_DIALOG shows the menu previously created as well as all defined menu parameters, help messages and dynamic fields. The values of the dynamic fields may be changed and written back to the menu descriptor file.

CONTENTS

C.1 Introduction
C.2 Starting MEN_DIALOG
C.3 Selection of menu descriptor files available
C.4 Menu Dialog Functions
   - ENTER a value in a dynamic menu field
   - HELP for a dynamic menu field
   - RANGE of a dynamic menu field
   - INFORMATION about a dynamic menu field
   - PRINT the current menu page
   - QUIT - exit the dialog menu without modifications
   - SAVE - exit the dialog menu and save modifications
C.1 Introduction

MEN_DIALOG - the standard Menu Dialog Program - is a program available on all VAX/VMS and UNIX systems. It may be run from any terminal or X-Window display supporting VT200/300/400 mode.

MEN_DIALOG may be used to

- list which menu files are available in the current default directory;
- select the menu descriptor file desired;
- display the contents of the menu descriptor file;
- modify the values of a dynamic field;
- display the predefined help message of a dynamic field;
- show the specified INTEGER or REAL range of a dynamic field;
- give information about field definitions;
- print the current menu page including field values;
- save all modifications made on the current menu in the selected menu file.

The program MEN_DIALOG is largely self-explanatory by means of function keys and help messages, but it is nevertheless recommended first to read chapter A (Overview) for better understanding of the operations performed by MEN_DIALOG.

C.2 Starting MEN_DIALOG

MEN_DIALOG may be started in two ways:

a) Specifying a single menu descriptor file:

* Log on to the terminal desired
* After the $ prompt, type: men_dialog <menu-file>

MEN_DIALOG will enter the dialog menu page for the specified menu descriptor file directly.

b) Menu descriptor file specification omitted:

* Log on to the desired terminal
* After the $ prompt, type: men_dialog

In this case MEN_DIALOG first enters a menu page to select one of the menu descriptor files available (see chapter C.3).
C.3 Selection of menu descriptor files available

If MEN_DIALOG was called without the file argument, the current user directory will be scanned for existing ".men" files. If any ".men" files found, MEN_DIALOG will show the files (without file extension) on a separate menu page. The user may then choose the menu descriptor file to be displayed or modified by moving the cursor to the desired filename and pressing the SELECT function key.

If no menu file is found, the error message "MEN_DIALOG - No menu files found in current directory!" will appear on the terminal screen.

Function Keys:

^ (Arrow Keys)  
<-|-> Change the current menu data field by moving the current field to a differing number.

**SELECT**  
[*] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] Enter the dialog menu page to show and modify the menu descriptor file selected.

**EXIT**  
[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [*] Stop MEN_DIALOG and exit to VMS/UNIX level.

C.4 Menu Dialog Functions

This menu page is entered after issuing the "SELECT" command or immediately when MEN_DIALOG is called with a valid menu descriptor file specification. The contents of the selected menu descriptor file will be displayed on screen.

The user may then select one of the dynamic fields available in the current menu file by usage of the arrow keys.

The following operations may be performed on the selected field:

^ (Arrow Keys)  
<-|-> Change the current menu data field by moving the current field to a different number.
ENTER
Modify the value of the current field by typing in the value desired. When the field has 'range checking' enabled, only values lying in the specified range will be accepted.

HELP
Display the predefined help message of the current field on the terminal screen.

RANGE
Show the specified INTEGER or REAL range of the current field on the terminal screen.

INFO
Give information about definitions made on the current field:
- Field identification number (ID)
- Field name
- Field I/O direction
- Field format

PRINT
Print the contents (background and field values) of the current menu descriptor file on a selected printer. The name of the printer must be specified.
Note: The print operation performed depends on the operating system used!

QUIT
Return to SELECT menu page or VMS/UNIX level without modifications on menu file.

SAVE
Return to SELECT menu page or VMS/UNIX level and save all modifications in the selected menu descriptor file.

c) Errors:
- SYNTAX error
- Field identification number is too long
- Field identification number is not defined
- Illegal field number (Range 1..100)
For UNIX shell programming a set of programs are available to perform non-interactive read or write operations on existing menu descriptor files.

D.1 Unix Shell Function: men_get_i

a) Description:

The "men_get_i" program is used to get an INTEGER value from a selected field of an existing menufile. The operation of the "men_get_i" program is the same as the MEN_GETI subroutine, but may be performed by a UNIX shell script. If the specified field does not have INTEGER format, then "men_get_i" tries to convert the given field value to INTEGER format. If this is not possible, "men_get_i" returns a fatal error.

b) Command definition:

```
$ men_get_i <menufile> <id>
or:
$ men_get_i <menufile> <fieldname>
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>menufile</td>
<td>Name of desired menufile</td>
</tr>
<tr>
<td>id</td>
<td>Identification number to reference the field desired. Range: 1..99</td>
</tr>
<tr>
<td>fieldname</td>
<td>Name of desired field</td>
</tr>
</tbody>
</table>

c) Example: see MEN_EXAMPLE8

d) Errors:

After calling 'men_get_i', the result value may contain the following messages:

* MEN_GET_I - Syntax error !
* MEN_GET_I - field identification number is missing !
* MEN_GET_I - Error on opening specified menu file !
* MEN_GET_I - Error on reading integer field: STAT

<table>
<thead>
<tr>
<th>STAT</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>conversion of field format not possible</td>
</tr>
<tr>
<td>5</td>
<td>only input allowed on field</td>
</tr>
<tr>
<td>6</td>
<td>specified field is not defined</td>
</tr>
<tr>
<td>7</td>
<td>illegal field number (Range 1..100)</td>
</tr>
</tbody>
</table>
D.2 Unix Shell Function: men_get_r

a) Description:

The "men_get_r" program is used to get a REAL value from a selected field of an existing menufile. The operation of the "men_get_r" program is the same as the MEN_GETR subroutine, but may be performed by a UNIX shell script. If the specified field does not have REAL format, then "men_get_r" tries to convert the given field value to REAL format. If this is not possible, "men_get_r" returns a fatal error.

b) Command definition:

```
$ men_get_r <menufile> <id>
or:
$ men_get_r <menufile> <fieldname>
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>menufile</td>
<td>Name of desired menufile</td>
</tr>
<tr>
<td>id</td>
<td>Identification number to reference the field</td>
</tr>
<tr>
<td>fieldname</td>
<td>Name of desired field</td>
</tr>
</tbody>
</table>

c) Example:

```
#!/bin/csh
# example for a c-shell script:
#
# Get a REAL value from menu file "men_example.men"
# field number 2 (Name: Field-2) by shell script:
# -----------------------------------------------
set RVAL = `men_get_r men_example.men 2`
# or:
set RVAL = `men_get_r men_example.men Field-2`
#
echo "Current real value of field 2: $RVAL"
exit $status
```

d) Errors:

After calling 'men_get_r', the result value may contain the following messages:

- *MEN_GET_R* - Syntax error!
- *MEN_GET_R* - field identification number is missing!
- *MEN_GET_R* - Error on opening specified menu file!
- *MEN_GET_R* - Error on reading REAL field: STAT

<table>
<thead>
<tr>
<th>STAT</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>conversion of field format not possible</td>
</tr>
<tr>
<td>5</td>
<td>only input allowed on field</td>
</tr>
<tr>
<td>6</td>
<td>specified field is not defined</td>
</tr>
<tr>
<td>7</td>
<td>illegal field number (Range 1..100)</td>
</tr>
</tbody>
</table>
D.3 Unix Shell Function: men_get_c

a) Description:

The "men_get_c" program is used to get a CHARACTER*n value from a selected field of an existing menufile.
The operation of the "men_get_c" program is the same as the MEN_GETC subroutine, but may be performed by
a UNIX shell script. If the specified field does not have CHARACTER*n format, then "men_get_c" returns a
fatal error.

b) Command definition:

```
$ men_get_c <menufile> <id>
or:
$ men_get_c <menufile> <fieldname>
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>menufile</td>
<td>Name of desired menufile</td>
</tr>
<tr>
<td>id</td>
<td>Identification number to reference the field defined. Range: 1..99</td>
</tr>
<tr>
<td>fieldname</td>
<td>Name of desired field</td>
</tr>
</tbody>
</table>

c) Example:

```
#!/bin/csh
# example for a c-shell script:
#
# Get a CHARACTER*n value from menu file "men_example.men"
# field number 3 (Name: Field-3) by shell script:
# --------------------------------------------------------
set CVAL = `men_get_c men_example.men 3`
# or:
set CVAL = `men_get_c men_example.men Field-3`
# echo "Current CHARACTER value of field 3: $CVAL"
exit $status
```

d) Errors:

After calling 'men_get_c', the result value may contain the following messages:

* MEN_GET_C - Syntax error !  
* MEN_GET_C - field identification number is missing !  
* MEN_GET_C - Error on opening specified menu file !  
* MEN_GET_C - Error on reading CHARACTER field: STAT

<table>
<thead>
<tr>
<th>STAT</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>conversion of field format not possible</td>
</tr>
<tr>
<td>5</td>
<td>only input allowed on field</td>
</tr>
<tr>
<td>6</td>
<td>specified field is not defined</td>
</tr>
<tr>
<td>7</td>
<td>illegal field number (Range 1..100)</td>
</tr>
</tbody>
</table>
D.4 Unix Shell Function: men_set_i

a) Description:

The "men_set_i" program is used to set an INTEGER value into a selected field of an existing menufile. The operation of the "men_set_i" program is the same as the MEN_SETI routine, but may be performed by a UNIX shell script. If the specified field does not have INTEGER format, then "men_set_i" changes the field format to INTEGER.

b) Command definition:

$ men_set_i <menufile> <id> <IVAL>
or:
$ men_set_i <menufile> <fieldname> <IVAL>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>menufile</td>
<td>Name of desired menufile</td>
</tr>
<tr>
<td>id</td>
<td>Identification number to reference the field desired. Range: 1..99</td>
</tr>
<tr>
<td>fieldname</td>
<td>Name of desired field</td>
</tr>
<tr>
<td>IVAL</td>
<td>INTEGER value to set</td>
</tr>
</tbody>
</table>

c) Example:

```bash
#!/bin/csh
# example for a c-shell script:
#
# Set an INTEGER value into menu file "men_example.men"
# field number 1 (Name: Field-1) by shell script:
#
@ IVAL = 1234
men_set_i men_example.men 1 $IVAL
or:
men_set_i men_example.men Field-1 $IVAL
exit $ival
```

d) Errors:

After calling 'men_set_i', the result value may contain the following messages:

* MEN_SET_I - Syntax error!
* MEN_SET_I - field identification number is missing!
* MEN_SET_I - Error on opening specified menu file!
* MEN_SET_I - INTEGER value is missing or illegal!
* MEN_SET_I - Error on writing to integer field: STAT

<table>
<thead>
<tr>
<th>STAT</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>conversion of field format not possible</td>
</tr>
<tr>
<td>5</td>
<td>only input allowed on field</td>
</tr>
<tr>
<td>6</td>
<td>specified field is not defined</td>
</tr>
<tr>
<td>7</td>
<td>illegal field number (Range 1..100)</td>
</tr>
</tbody>
</table>
D.5 Unix Shell Function: men_set_r

a) Description:

The "men_set_r" program is used to set a REAL value into a selected field of an existing menufile. The operation of the "men_set_r" program is the same as the MEN_SETR routine, but may be performed by a UNIX shell script. If the specified field does not have REAL format, then "men_set_r" changes the field format to REAL.

b) Command definition:

$ men_set_r <menufile> <id> <RVAL>
or:
$ men_set_r <menufile> <fieldname> <RVAL>

Parameter | Meaning
-----------|------------------
menufile   | Name of desired menufile
id         | Identification number to reference the field desired. Range: 1..99
fieldname  | Name of desired field
RVAL       | REAL value to set

c) Example:

```csh
#!/bin/csh
e# example for a c-shell script:
e#
e# Set a REAL value into menu file "men_example.men"
e#
e# --------
set RVAL = "12.34E-2"
men_set_r men_example.men 2 $RVAL
#
or:
men_set_r men_example.men Field-2 $RVAL
exit $status
```

d) Errors:

After calling 'men_set_r', the result value may contain the following messages:

* MEN_SETR - Syntax error !
* MEN_SETR - field identification number is missing !
* MEN_SETR - Error on opening specified menu file !
* MEN_SETR - REAL value is missing or illegal !
* MEN_SETR - Error on writing to REAL field: STAT

STAT | Meaning
-----|------------------
 4   | conversion of field format not possible
 5   | only input allowed on field
 6   | specified field is not defined
 7   | illegal field number (Range 1..100)
D.6 Unix Shell Function: men_set_c

a) Description:

The "men_set_c" program is used to set a CHARACTER value into a selected field of an existing menufile. The operation of the "men_set_c" program is the same as the MEN_SETC routine, but may be performed by a UNIX shell script. If the specified field does not have CHARACTER format, then "men_set_c" changes the field format to CHARACTER.

b) Command definition:

```
$ men_set_c <menufile> <id> <CVAL>
or:
$ men_set_c <menufile> <fieldname> <CVAL>
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>menufile</td>
<td>Name of desired menufile</td>
</tr>
<tr>
<td>id</td>
<td>Identification number to reference the field desired. Range: 1..99</td>
</tr>
<tr>
<td>fieldname</td>
<td>Name of desired field</td>
</tr>
<tr>
<td>CVAL</td>
<td>CHARACTER value to set</td>
</tr>
</tbody>
</table>

c) Example:

```
#!/bin/csh
# example for a c-shell script:
# Set a CHARACTER value into menu file "men_example.men"
# field number 3 (Name: Field-3) by shell script:
# set CVAL = "This is a new text"
men_set_c men_example.men 3 $CVAL
# or:
men_set_c men_example.men Field-3 $CVAL
exit $status
```

d) Errors:

After calling 'men_set_c', the result value may contain the following messages:

* MEN_SET_C - Syntax error !
* MEN_SET_C - field identification number is missing !
* MEN_SET_C - Error on opening specified menu file !
* MEN_SET_C - CHARACTER value is missing or illegal !
* MEN_SET_C - Error on writing to CHARACTER field: STAT

<table>
<thead>
<tr>
<th>STAT</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>conversion of field format not possible</td>
</tr>
<tr>
<td>5</td>
<td>only input allowed on field</td>
</tr>
<tr>
<td>6</td>
<td>specified field is not defined</td>
</tr>
<tr>
<td>7</td>
<td>illegal field number (Range 1..100)</td>
</tr>
</tbody>
</table>
E. Program MEN_CONVERT

a) Description:

Menu descriptor files are structured binary files as already explained in chapter A.4. Therefore it is necessary to convert existing menu descriptor files (.men) after transferring to a computer system of a different type (processor).

The following format conversions are currently supported:

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSF</td>
<td>VMS</td>
<td>Conversion from OSF to VMS</td>
</tr>
<tr>
<td>OSF</td>
<td>SUN</td>
<td>Conversion from OSF to SUN/SOLARIS</td>
</tr>
<tr>
<td>VMS</td>
<td>OSF</td>
<td>Conversion from VMS to OSF</td>
</tr>
<tr>
<td>VMS</td>
<td>SUN</td>
<td>Conversion from VMS to SUN/SOLARIS</td>
</tr>
<tr>
<td>SUN</td>
<td>OSF</td>
<td>Conversion from SUN/SOLARIS to OSF</td>
</tr>
<tr>
<td>SUN</td>
<td>VMS</td>
<td>Conversion from SUN/SOLARIS to VMS</td>
</tr>
</tbody>
</table>

Calling MEN_CONVERT without arguments a help message will be appear on the terminal screen.

NOTE: 1) The DEC/OSF and DEC/ULTRIX computer systems have the identical format, therefore conversion of binary files must not be performed.

2) The SUN/SOLARIS and IBM/AIX have identical format.

2) MEN_CONVERT is not available on VAX/VMS !

b) Command definition:

```
$ men_convert <menufile> <source-OS> <destination-OS>
```

Parameter | Meaning
-----------|--------------------------------------------------

**menufile**  
Name of desired menu descriptor file to convert.

**source-OS**  
Format of the current menu descriptor file which should be converted.  
Format names: VMS, OSF, SUN, AIX

**destination-OS**  
Desired format of menu descriptor file after conversion.  
Format names: VMS, OSF, SUN, AIX
c) Example:

We assume that the menu descriptor file "men_example.men" has been created on a VAX/VMS computer and should be used on a ALPHA/OSF1 workstation. In this case, we must first transfer the menu file in binary mode (FTP etc.) to the OSF system and then call MEN_CONVERT to convert the VMS file into OSF format:

```
$ men_convert men_example.men VMS OSF
```

After that file "men_example.men" has the correct OSF format and may be used by MEN-Utility programs or user programs.

d) Errors:

After calling 'men_convert', the following error messages may occur:

- *MEN_CONVERT* - Syntax error!
- *MEN_CONVERT* - specified menu descriptor file not found!
- *MEN_CONVERT* - specified file has an illegal format!
- *MEN_CONVERT* - unknown SOURCE operating system!
- *MEN_CONVERT* - unknown DESTINATION operating system!

B) Parameters:

```
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER</td>
<td>Identification number of menu descriptor file</td>
<td>1..99</td>
</tr>
<tr>
<td>NAME</td>
<td>CHAR(32)</td>
<td>Name of menu descriptor file</td>
<td>'test'</td>
</tr>
<tr>
<td>LIN</td>
<td>INTEGER</td>
<td>Line number of screen field</td>
<td>1..20</td>
</tr>
<tr>
<td>ROW</td>
<td>INTEGER</td>
<td>Row number of screen field</td>
<td>1..20</td>
</tr>
</tbody>
</table>
```

Remarks:

- If MEN_OPEN is called after a previous MEN_OPEN call, the current menu area page will be overwritten with generated area.
- MEN_OPEN should be called after a successful MENU_CLOSE call.


A) Description:

The MEN_OPEN routine is for opening a previously saved menu descriptor file named FNAME. If FNAME = ' ' then a local menu database will be created on the first call to MEN_OPEN and initialized with default values. If ERR = true, then the specified menu descriptor file was not found in the current default directory. MEN OPEN must be called before any other MEN routine may be used.

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>FNAME</td>
<td>CHARACTER*80 (input)</td>
<td>name of menu descriptor file to be opened. If FNAME = ' ' local menu data base will be initialized.</td>
</tr>
<tr>
<td>ERR</td>
<td>LOGICAL</td>
<td>If FNAME not found, ERR = .TRUE.</td>
</tr>
</tbody>
</table>

C) Definition of subroutine:

SUBROUTINE MEN_OPEN (FNAME, ERR)

D) Example:

    LOGICAL ERR
    C open an existing menu descriptor file:
    CALL MEN_OPEN ('example.men',ERR)
    IF (ERR) PRINT *, 'Menu file not found'

    C create a new menu data base:
    CALL MEN_OPEN (' ',ERR)

E) Errors:

An error occurs if a file is explicitly specified (FNAME is not blank) does not exist.

F) Remarks:

If MEN_OPEN is called after a previous MEN OPEN call the current menu data base will be overwritten with default values.
A) Description:

The MEN_DEFINE routine specifies an entry in the current menu database for a dynamic screen field. This field may then be used for input and output operations.

To identify the field by further operations, the user must define an identification number in the range 1..99. Furthermore an appropriate name (or '') should be specified for the field. To reference a field by other MEN routines only the identification number must then be specified.

A field is completely defined by the screen position (line/row), the field length, the format, the I/O direction and a display attribute.

Each field must be first defined by MEN_DEFINE before other MEN routines (like MEN_SET, MEN_READ etc.) may reference it. MEN DEFINE may be also called to redefine a previously defined field.

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER</td>
<td>Identification number to reference the field. Range: 1..99</td>
</tr>
<tr>
<td></td>
<td>(input)</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>CHARACTER*22</td>
<td>Name of field. If not needed, define NAME = ''</td>
</tr>
<tr>
<td></td>
<td>(input)</td>
<td></td>
</tr>
<tr>
<td>LIN</td>
<td>INTEGER</td>
<td>Line number of screen field. Range: 1..24</td>
</tr>
<tr>
<td></td>
<td>(input)</td>
<td></td>
</tr>
<tr>
<td>ROW</td>
<td>INTEGER</td>
<td>Starting row number of screen field. Range: 1..80</td>
</tr>
<tr>
<td></td>
<td>(input)</td>
<td></td>
</tr>
<tr>
<td>LNG</td>
<td>INTEGER</td>
<td>Length of screen field (characters). Range: 1..80</td>
</tr>
<tr>
<td></td>
<td>(input)</td>
<td></td>
</tr>
</tbody>
</table>
C) Definition of subroutine:

SUBROUTINE MEN_DEFINE (ID, NAME, LIN, ROW, LNG, FMT, IO, ATT)

D) Example:

LOGICAL ERR

C 1. Create a menu database:
CALL MEN_OPEN (', ', ERR)

C 2. Define a dynamic screen field
CALL MEN_DEFINE ( 1, 'FLD1', 5, 6, 10, 'I', 'B', 'N')

E) Errors:

If any error occurs, MEN_DEFINE will display the appropriate error message on the terminal and abort the program.

F) Remarks:

A menu database must have been created by MEN_OPEN before the first call to MEN_DEFINE performed.
**A) Description:**

The MEN_INFO routine gives information about a specified menu field. The information returned consists of:

- the screen position (line/row),
- the field length,
- the format,
- the I/O direction and
- the display attribute.

Furthermore MEN_INFO may be used to test if a selected field (ID) is already defined or not (see example).

**B) Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER</td>
<td>Identification number to reference the field. Range: 1..99</td>
</tr>
<tr>
<td>(input)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>CHARACTER*22</td>
<td>Name of selected field.</td>
</tr>
<tr>
<td>(output)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LIN</td>
<td>INTEGER</td>
<td>Line number of selected field.</td>
</tr>
<tr>
<td>(output)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROW</td>
<td>INTEGER</td>
<td>Starting row number of selected field.</td>
</tr>
<tr>
<td>(output)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LNG</td>
<td>INTEGER</td>
<td>Length of selected field (characters).</td>
</tr>
<tr>
<td>(output)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**C) Definition of subroutine:**

```plaintext
SUBROUTINE MEN_INFO (ID, NAME, LIN, ROW, LNG, FMT, IO, ATT)
```

**D) Example:**

```plaintext
INTEGER ID,LIN,ROW,LNG
CHARACTER FMT,IO,ATT
CHARACTER*22 NAME

CALL MEN_INFO (1, NAME, LIN, ROW, LNG, FMT, IO, ATT)
```

**C)** Get information about field 1:

```plaintext
CALL MEN_INFO (1, NAME, LIN, ROW, LNG, FMT, IO, ATT)
IF (FMT .EQ. ' ') PRINT *, 'Field 1 is not defined'
```

**E) Errors:**

If any error occurs, MEN_INFO will display the appropriate error message on the terminal and abort the program.

**F) Remarks:**

A menu database must have been created by MEN_OPEN before the first call to MEN_INFO performed.
A) Description:

The MEN MODIFY routine allows to change the parameters set by a previously performed MEN_DEFINE operation. The following options are available:

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>'FMT'</td>
<td>CVAL</td>
<td>change the format of specified field</td>
</tr>
<tr>
<td>'I/O'</td>
<td>CVAL</td>
<td>change the I/O direction of specified field</td>
</tr>
<tr>
<td>'ATT'</td>
<td>CVAL</td>
<td>change the display attribute of specified field</td>
</tr>
<tr>
<td>'ROW'</td>
<td>IVAL</td>
<td>change the row number of specified field</td>
</tr>
<tr>
<td>'LIN'</td>
<td>IVAL</td>
<td>change the line number of specified field</td>
</tr>
<tr>
<td>'LEN'</td>
<td>IVAL</td>
<td>change the length of specified field</td>
</tr>
</tbody>
</table>

Depending on the selected option, either the parameter IVAL or CVAL will be used to set the desired value.

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER (input)</td>
<td>Identification number to reference the field. Range: 1..99</td>
</tr>
<tr>
<td>OPT</td>
<td>CHARACTER*3 (input)</td>
<td>Desired option (see above)</td>
</tr>
<tr>
<td>IVAL</td>
<td>INTEGER (input)</td>
<td>INTEGER value to modify: (LINE, ROW or LENGTH)</td>
</tr>
<tr>
<td>CVVAL</td>
<td>CHARACTER (input)</td>
<td>CHARACTER value to modify: (FMT, IO or ATT)</td>
</tr>
</tbody>
</table>
C) Definition of subroutine:

SUBROUTINE MEN_MODIFY (ID, OPT, IVAL, CVAL)

D) Example:

```
INTEGER ID,IVAL
CHARACTER CVAL
CHARACTER*3 OPT

C Modify the length of field #3 to 10 characters:
C---------------------------------------------------------
IVAL = 10
CALL MEN_MODIFY (3, 'LEN', IVAL, CVAL)

C Modify the line position of field #3 to line 16:
C---------------------------------------------------------------
CALL MEN_MODIFY (3, 'LIN', 16, CVAL)

C Modify the format of field #2 to REAL:
C---------------------------------------------
CVAL = 'R'
CALL MEN_MODIFY (2, 'FMT', IVAL, CVAL)

C Modify the display attribute of field #2 to BLINKING:
C---------------------------------------------------------------
CALL MEN_MODIFY (2, 'ATT', IVAL, 'B')
```

E) Errors:

If any error occurs, MEN_MODIFY will display the appropriate error message on the terminal and abort the program.

F) Remarks:

A menu database must have been created by MEN_OPEN and the field must be defined before calling MEN_MODIFY.
A) Description:

The MEN_SET routine is used to set a value of any possible format (INTEGER, REAL or CHARACTER*n) into a selected menu database field.

Depending on the previously defined field format (by MENDEFINE parameter 'FMT'), the corresponding parameter (IVAL,IVAL or CVAL) will be used to set the value.

<table>
<thead>
<tr>
<th>Field Format</th>
<th>Used Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>'I'</td>
<td>IVAL</td>
<td>content of IVAL will be set</td>
</tr>
<tr>
<td>'R'</td>
<td>RVAL</td>
<td>content of RVAL will be set</td>
</tr>
<tr>
<td>'F'</td>
<td>RVAL</td>
<td>content of RVAL will be set</td>
</tr>
<tr>
<td>'C'</td>
<td>CVAL</td>
<td>content of CVAL will be set</td>
</tr>
</tbody>
</table>

The unused parameters will be ignored by MEN_SET, but must be present for FORTRAN calling standard.

**NOTE:** MEN_SET will only save the desired value in the menu database, but will not display the value on screen!

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER (input)</td>
<td>Identification number to reference the desired field. Range: 1..99</td>
</tr>
<tr>
<td>IVAL</td>
<td>INTEGER (input)</td>
<td>Integer value to set into database if FMT = 'I'; otherwise ignored.</td>
</tr>
<tr>
<td>RVAL</td>
<td>REAL (input)</td>
<td>Real value to set into database if FMT = 'R' or 'F'; otherwise ignored.</td>
</tr>
<tr>
<td>CVAL</td>
<td>CHARACTER*n (input)</td>
<td>Character value to set into database if FMT = 'C'; otherwise ignored.</td>
</tr>
</tbody>
</table>
C) Definition of subroutine:

```
SUBROUTINE MEN_SET (ID, IVAL, RVAL, CVAL)
```

D) Example:

```
INTEGER ID, IVAL, I
REAL RVAL
CHARACTER*80 CVAL

C Set the value 123 into menu database, field 1:
C ---------------------------------------------
CALL MEN_SET (1, 123, RVAL, CVAL)

C Set the value 1.23E3 into menu database, field 2:
C -------------------------------------------------
RVAL = 1.23E3
CALL MEN_SET (2, IVAL, RVAL, CVAL)

C Set the value 'Hallo' into menu database, field 3:
C -------------------------------------------------
CVAL = 'Hallo'
CALL MEN_SET (3, IVAL, RVAL, CVAL)

C If you want to display the new values on screen:
C ------------------------------------------------
DO 10 I=1,3
   CALL MEN_WRITE (I)
10 CONTINUE
```

E) Errors:

If any error occurs, MEN_SET will display the appropriate error message on the terminal and abort the program.

F) Remarks:

A menu database must have been created by MEN_OPEN and the field must be defined before calling MEN_SET.
The MEN_GET routine is used to get a value of any possible format (INTEGER, REAL or CHARACTER*n) from a selected menu database field.

Depending on the previously defined field format (by MEN_DEFINE parameter 'FMT'), the corresponding parameter (IVAL, RVAL or CVAL) will be used to get the value.

<table>
<thead>
<tr>
<th>Field Format</th>
<th>Used Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>'I'</td>
<td>IVAL</td>
<td>returns an INTEGER value</td>
</tr>
<tr>
<td>'R'</td>
<td>RVAL</td>
<td>returns a REAL value</td>
</tr>
<tr>
<td>'F'</td>
<td>RVAL</td>
<td>returns a REAL value</td>
</tr>
<tr>
<td>'C'</td>
<td>CVAL</td>
<td>returns a CHARACTER*n value</td>
</tr>
</tbody>
</table>

The unused parameters will be ignored by MEN_GET, but must be present for FORTRAN calling standard.

NOTE: MEN_GET will only return the desired value from the menu database, but will not perform a read function from the terminal screen field!

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER</td>
<td>Identification number to reference the desired field. Range: 1..99</td>
</tr>
<tr>
<td>IVAL</td>
<td>INTEGER</td>
<td>Integer variable to get value from menu database, if FMT = 'I'; otherwise ignored.</td>
</tr>
<tr>
<td>RVAL</td>
<td>REAL</td>
<td>Real variable to get value from menu database, if FMT = 'R' or 'F'; otherwise ignored.</td>
</tr>
<tr>
<td>CVAL</td>
<td>CHARACTER*n</td>
<td>Character variable to get value from menu database, if FMT = 'C'; otherwise ignored.</td>
</tr>
</tbody>
</table>
C) Definition of subroutine:

SUBROUTINE MEN_GET (ID, IVAL, RVAL, CVAL)

D) Example:

INTEGRER ID, IVAL
REAL RVAL
CHARACTER*80 CVAL

C Get the current value from menu database, field 1:
CALL MEN_GET (1, IVAL, RVAL, CVAL)

C Depending on the format of field #1, IVAL, RVAL or
C CVAL contains the current value of screen field #1

E) Errors:

If any error occurs, MEN_GET will display the appropriate
error message on the terminal and abort the program.

F) Remarks:

A menu database must have been created by MEN_OPEN and the field
must be defined before calling MEN_GET.
A) Description:

The MEN_READ routine is used to read a value from a specified terminal screen field into the menu database. Depending on the previously defined field format (by MEN DEFINE parameter 'FMT'), INTEGER, REAL or CHARACTER*n values will be read from the screen field. If any illegal value was entered (for example: 123xy for an INTEGER field), the screen field will be cleared and new input requested. The input is finished, when a "RETURN" was entered or the maximum field length is reached. Function keys or Control characters will be ignored during MEN_READ input.

NOTE: MEN_READ will only read input from terminal screen field and save the value in the corresponding menu database field.

B) Parameter | Type | Meaning

ID INTEGER Identification number to reference the desired field. Range: 1..99

C) Definition of subroutine:

SUBROUTINE MEN_READ (ID)

D) Example:

INTEGER ID, IVAL
REAL RVAL
CHARACTER*80 CVAL

CALL MEN_READ (1)
CALL MEN_GET (1, IVAL, RVAL, CVAL)
PRINT *,'Current value of field 1:',IVAL

E) Errors:

If any error occurs, MEN_READ will display the appropriate error message on the terminal and abort the program.

F) Remarks:

A menu database must have been created by MEN_OPEN and the field must be defined before calling MEN_READ.
A) Description:

The MEN_WRITE routine displays a value from a specified menu database field on terminal screen. Depending on the previously defined field format (by MEN_DEFINE parameter 'FMT'), INTEGER, REAL or CHARACTER*n values will be displayed on terminal screen field. The output dimension of the values will be evaluated to fit into the specified field length. Therefore it is possible on output, that characters or digits may be truncated if their length would exceed the field width!

MEN_WRITE does only display fields, which have been defined as OUTPUT or BIDIRECTIONAL (see MEN_DEFINE parameter 'IO').

If MEN_WRITE will be called with argument 0 (ID=0), then all menu fields will be displayed, which are defined and have the OUTPUT or BIDIRECTIONAL qualifier (MEN_DEFINE parameter 'IO').

NOTE: MEN_WRITE will only display the current menu database values of the selected field.

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER</td>
<td>Identification number to reference the desired field. Range: 1..99</td>
</tr>
<tr>
<td></td>
<td>(input)</td>
<td>If ID=0, then all defined fields will be displayed.</td>
</tr>
</tbody>
</table>

C) Definition of subroutine:

SUBROUTINE MEN_WRITE (ID)
D) Example:

```fortran
INTEGER ID, IVAL
REAL RVAL
CHARACTER*80 CVAL
```

We assume, that field #1 is defined as REAL field.

Set and Display a new value on terminal screen field #1:

```fortran
CALL MEN_SET (1, IVAL, 1.97E-4, CVAL)
CALL MEN_WRITE (1)
```

E) Errors:

A) If any error occurs, MEN_WRITE will display the appropriate error message on the terminal and abort the program.

F) Remarks:

A menu database must have been created by MEN_OPEN and the field must be defined before calling MEN_WRITE.
A) Description:

The MEN_SAVE routine is used to save the current menu definitions and field values into a disk file named FNAME.

WARNING: If a file with name FNAME already exist on disk, the file will be overwritten by the MEN_SAVE call!

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>FNAME</td>
<td>CHARACTER*80 (input)</td>
<td>Name of menu descriptor file to be created.</td>
</tr>
</tbody>
</table>

C) Definition of subroutine:

SUBROUTINE MEN_SAVE (FNAME)

D) Example:

C Create menu descriptor file "example.men"
C and save current menu database:
C
CALL MEN_SAVE ('example.men')

E) Errors:

If any error occurs, MEN_SAVE will display the appropriate error message on the terminal and abort the program.

F) Remarks:

A menu database must have been created by MEN_OPEN before the first call to MEN_SAVE performed.
A) Description:

The MEN_REFRESH routine first clears the screen and sets the cursor to the 1st line. Then the background lines will be displayed (if any available) and all defined dynamic screen fields contents will be written to screen. MEN_REFRESH will be most often used to refresh the whole terminal screen after any (non MEN) terminal output have been made.

NOTE: MEN_REFRESH does only display fields, which have been defined as OUTPUT or BIDIRECTIONAL (see MEN_DEFINE parameter 'IO').

B) Parameters: none

C) Definition of subroutine:

SUBROUTINE MEN_REFRESH

D) Example:

LOGICAL ERR

C Read menu file 'example.men' and display menu on screen:
C
CALL MEN_OPEN ('example.men', ERR)
CALL MEN_REFRESH

E) Errors:

If any error occurs, MEN_REFRESH will display the appropriate error message on the terminal and abort the program.

F) Remarks:

A menu database must have been created by MEN_OPEN and the fields must be defined before calling MEN_REFRESH.
A) Description:

The MEN_SETID routine marks the specified field as the first field for possible input. On a following display function like MEN_WRITE or MEN_REFRESH the marked field will be displayed in 'reverse' representation and the cursor will be positioned to the start of the field.

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER</td>
<td>Identification number to reference the desired field. Range: 1..99</td>
</tr>
</tbody>
</table>

C) Definition of subroutine:

SUBROUTINE MEN_SETID (ID)

D) Example:

C We assume, that field #1 is defined and have the BIDIRECTIONAL qualifier.
C Mark and Display screen field #1:
C
CALL MEN_SETID (1)
CALL MEN_WRITE (1)

E) Errors:

If the specified field is explicitly declared as OUTPUT field (MEN_DEFINE parameter 'IO'), function MEN_SETID will display an error message on the terminal and abort.

F) Remarks:

A menu database must have been created by MEN_OPEN and the field must be defined before calling MEN_SETID.
The MEN_GETID routine returns the number of an input field, on which was made the last input operation in the current menu. This number will be saved on menu descriptor files and therefore the user may continue input operations on the same menu position (field) on which the last input has been performed.

Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>FID</td>
<td>INTEGER</td>
<td>Current field identification number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(output)</td>
</tr>
</tbody>
</table>

Definition of subroutine:

SUBROUTINE MEN_GETID (FID)

Example:

```
INTEGER FID
LOGICAL ERR
C Read menu file 'example.men' and get marked start field:
CALL MEN_OPEN ('example.men', ERR)
CALL MEN_GETID (FID)
PRINT *, 'Last used input field:', FID
```

Errors:

If any error occurs, MEN_GETID will display the appropriate error message on the terminal and abort the program.

Remarks:

A menu database must have been created by MEN_OPEN and any input field must be defined before calling MEN_GETID.
A) Description:

The MEN_ID function searches in the current menu database for the desired field named "NAME". If the field was found, the corresponding field identification number will be returned as function result of MEN_ID. If no field with the specified name was found, MEN_ID will deliver a zero value.

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>CHARACTER*22</td>
<td>Name of desired menu field</td>
</tr>
</tbody>
</table>

C) Definition of function:

FUNCTION MEN_ID (NAME)

D) Example:

```plaintext
INTEGER FID, MEN_ID
CHARACTER*22 NAME
```

C We assume, that menu field 5 exist and have the name 'FIELD-5'. To read a value from this field, the field may be addressed either by ID or by name:

```plaintext
C 1. Read value by ID:
   CALL MEN_READ (5)
```

C 2. Read value by NAME:

```plaintext
   CALL MEN_READ (MEN_ID('FIELD-5'))
```

E) Errors:

If the specified field name was not found, a zero field ID will be returned as function result.

F) Remarks:

A menu database must have been created by MEN_OPEN and any input field must be defined before calling MEN_ID.
The MEN_SETBGRD routine sets a line of 'background' characters into the specified line buffer of menu database.

Menu background means a static text, which is organized into 24 lines of 80 characters. The text lines may be set or modified by the MEN_SETBGRD routine.

There are two ways to create a menu background:

1. Write menu background by using own FORTRAN WRITE or PRINT statements.

2. Using MEN_SETBGRD routine followed by a display output routine like MEN_WRITEBGRD or MEN_REFRESH.

The advantage of the 2nd method is that the background text may be saved in a menu descriptor file (or buffer), no problems with cursor positioning will occur and the background lines may be modified dynamically. Therefore it is recommended to use the MEN_SETBGRD routine.

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIN</td>
<td>INTEGER</td>
<td>Line number for background text</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(input) Range: 1..24</td>
</tr>
<tr>
<td>BLINE</td>
<td>CHARACTER*80</td>
<td>Desired background text for line number LIN</td>
</tr>
</tbody>
</table>

C) Definition of subroutine:

SUBROUTINE MEN_SETBGRD (LIN, BLINE)
D) Example:

```
INTEGER LIN
CHARACTER*80 BLINE(24)

C Define background lines 1-3:
C ----------------------
BLINE(1) = 'This is background line 1'
BLINE(2) = 'This is background line 2'
BLINE(1) = 'This is background line 3'

C Set background lines 1-3 into menu database:
C ---------------------------------------------
DO 10 LIN=1,3
   CALL MEN_SETBGRD (LIN, BLINE(LIN))
10 CONTINUE
```

E) Errors:

If any error occurs, MEN_SETBGRD will display the appropriate error message on the terminal and abort the program.

F) Remarks:

A menu database must have been created by MEN_OPEN before the first call to MEN_SETBGRD performed.
The MEN_GETBGRD routine gets a specified line of 'background' characters from menu database.
Menu background means a static text, which is organized into 24 lines of 80 characters. The currently defined background text lines may be get by the MEN_GETBGRD routine.
NOTE: if a specified background line was not previously defined by a MEN_SETBGRD call, a blank string will be returned!

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIN</td>
<td>INTEGER</td>
<td>Line number for background text</td>
</tr>
<tr>
<td></td>
<td>(input)</td>
<td>Range: 1..24</td>
</tr>
<tr>
<td>BLINE</td>
<td>CHARACTER*80</td>
<td>Background text returned for</td>
</tr>
<tr>
<td></td>
<td>(output)</td>
<td>line number LIN</td>
</tr>
</tbody>
</table>

C) Definition of subroutine:

SUBROUTINE MEN_GETBGRD (LIN, BLINE)

D) Example:

INTEGER    LIN
CHARACTER*80 BLINE

C Get the current background text line #1:
CALL MEN_GETBGRD (1, BLINE)
PRINT *, 'Current background line:'
PRINT *, BLINE

C Now we can modify the background line:
BLINE = 'This is a new background line #1'
CALL MEN_SETBGRD (1,BLINE)

E) Errors:
If any error occurs, MEN_GETBGRD will display the appropriate error message on the terminal and abort the program.

F) Remarks:
A menu database must have been created by MEN_OPEN before the first call to MEN_GETBGRD performed.
A) Description:

The MEN_WRITEBGRD routine displays all 'background' lines defined in the current menu database on terminal screen.

NOTE: Because MEN_WRITEBGRD overwrites the whole menu screen, previous displayed dynamic screen fields are lost. Therefore the order of calling display output routines is to obey:

1. display background by MEN_WRITEBGRD
2. display dynamic fields by MEN_WRITE (in this order)

or use MEN_REFRESH (do both functions).

B) Parameters: none

C) Definition of subroutine:

SUBROUTINE MEN_WRITEBGRD

D) Example:

C We assume, all needed background lines are defined
C by MEN_SETBGRD calls. To display the menu background:
C
CALL MEN_WRITEBGRD

E) Errors:

If any error occurs, MEN_WRITEBGRD will display an error message on the terminal and abort the program.

F) Remarks:

A menu database must have been created by MEN_OPEN before the first call to MEN_WRITEBGRD performed.
A) Description:

The MEN_SETHELP routine sets a 'HELP' message line for the specified field into menu database. The help message may consist of up to 80 characters and may contain any explanation for the selected field (like notes, input options etc.). This information will be saved in the current menu database and may be used for own HELP functions or by the MEN_DIALOG program.

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER</td>
<td>Identification number to reference the field. Range: 1..99</td>
</tr>
<tr>
<td></td>
<td>(input)</td>
<td></td>
</tr>
<tr>
<td>HLINE</td>
<td>CHARACTER*80</td>
<td>Desired help message for field number ID</td>
</tr>
<tr>
<td></td>
<td>(input)</td>
<td></td>
</tr>
</tbody>
</table>

C) Definition of subroutine:

SUBROUTINE MEN_SETHELP (ID, HLINE)

D) Example:

```
INTEGER ID
CHARACTER*80 HLINE

C Define a HELP message for field #1:
C ----------------------------------
HLINE = 'This is a HELP message for field 1'
CALL MEN_SETHELP (1, HLINE)
C or:
CALL MEN_SETHELP (1, 'This is a HELP message for field 1')
```

E) Errors:

If any error occurs, MEN_SETHELP will display the appropriate error message on the terminal and abort the program.

F) Remarks:

A menu database must have been created by MEN_OPEN and the field must be defined before calling MEN_SETHELP.
**Name:** MEN_GETHELP  **Library:** libmen77.a  **Class:** MEN

**Purpose:** Get the help message for specified menu field.

**Created:** 27.2.93  **Revised:**  **Version:** 1.2

**Author(s):** H. Kroiss  **Phone:** 1308

**Type of program:**  [ ] Main  [*] Subroutine  [ ] Function

**System:**  [*] UNIX  [*] VAX/VMS  [ ]

**Documentation:**  [*] Help  [*] Manual  [ ]

---

**A) Description:**

The MEN_GETHELP routine gets a 'HELP' message line for the specified field from the menu database. The help message may consist of up to 80 characters and may contain any explanation for the selected field (like notes, input options etc.). If the help message was not previously defined by a MEN_SETHELP call, a blank string will be returned.

**B) Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER</td>
<td>Identification number to reference the field. Range: 1..99</td>
</tr>
<tr>
<td>HLINE</td>
<td>CHARACTER*80</td>
<td>Help message returned for field number ID</td>
</tr>
</tbody>
</table>

**C) Definition of subroutine:**

```plaintext
SUBROUTINE MEN_GETHELP (ID, HLINE)
```

**D) Example:**

```plaintext
INTEGER ID
CHARACTER*80 HLINE

C Get the current HELP message for field #1:
CALL MEN_GETHELP (1, HLINE)
PRINT *, 'Help message for field 1:
PRINT *, HLINE
```

**E) Errors:**

If any error occurs, MEN_GETHELP will display the appropriate error message on the terminal and abort the program.

**F) Remarks:**

A menu database must have been created by MEN_OPEN and the field must be defined before calling MEN_GETHELP.
A) Description:

The MEN_SETRANGEI routine defines a range of legal INTEGER values for the specified field. This range will be checked by the MEN_READ routine during input of an INTEGER value. If the entered INTEGER number is less than the ILOW value or greater than the IHIGH value, the input will not be accepted and new input will be requested. The range checking may be disabled by specifying the ILOW value greater than the IHIGH value. For example:

```
ILOW = 1
IHIGH = 0
CALL MEN_SETRANGEI (ID, ILOW, IHIGH)
```

Note: Range checking is disabled by default.

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER</td>
<td>Identification number to reference the field. Range: 1..99</td>
</tr>
<tr>
<td>ILOW</td>
<td>INTEGER</td>
<td>Lowest INTEGER input value allowed for specified field.</td>
</tr>
<tr>
<td>IHIGH</td>
<td>INTEGER</td>
<td>Highest INTEGER input value allowed for specified field.</td>
</tr>
</tbody>
</table>

C) Definition of subroutine:

```
SUBROUTINE MEN_SETRANGEI (ID, ILOW, IHIGH)
```
D) Example:

```fortran
INTEGER ID, ILOW, IHIGH

C Define an input range for field #1:
C
ILLOW = -200
IHIGH = 1024
CALL MEN_SETRANGEI (1, ILOW, IHIGH)
```

E) Errors:
If any error occurs, MEN_SETRANGEI will display the appropriate error message on the terminal and abort the program.

F) Remarks:
A menu database must have been created by MEN_OPEN and the field must be defined before calling MEN_SETRANGEI.
Name: MEN_GETRANGEI  Library: libmen77.a  Class: MEN
Title: Get the actual INTEGER range of specified field
Created: 27.2.93  Revised:  Version: 1.2
Author(s): H. Kroiss  Phone: 1308
Type of program:  Main  [*] Subroutine  [*] Function
System:  [*] UNIX  [*] VAX/VMS
Documentation: [*] Help  [*] Manual

A) Description:

The MEN_GETRANGEI routine returns the currently defined INTEGER range values of the specified field.

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER</td>
<td>Identification number to reference the field. Range: 1..99</td>
</tr>
<tr>
<td></td>
<td>(input)</td>
<td></td>
</tr>
<tr>
<td>ILOW</td>
<td>INTEGER</td>
<td>Returns the actual low range value for specified field.</td>
</tr>
<tr>
<td></td>
<td>(output)</td>
<td></td>
</tr>
<tr>
<td>IHIGH</td>
<td>INTEGER</td>
<td>Returns the actual high range value for specified field.</td>
</tr>
<tr>
<td></td>
<td>(output)</td>
<td></td>
</tr>
</tbody>
</table>

C) Definition of subroutine:

```
SUBROUTINE MEN_GETRANGEI (ID, ILOW, IHIGH)
```

D) Example:

```
INTEGER ID, ILOW, IHIGH
C Get the currently defined input range of field #1:
C --------------------------------------------------
CALL MEN_GETRANGEI (1, ILOW, IHIGH)
PRINT *, 'Low value:', ILOW, ' High value:', IHIGH
```

E) Errors:

If any error occurs, MEN_GETRANGEI will display the appropriate error message on the terminal and abort the program.

F) Remarks:

A menu database must have been created by MEN_OPEN and the field must be defined before calling MEN_GETRANGEI.
A) Description:

The MEN_SETRANGER routine defines a range of legal REAL values for the specified field.
This range will be checked by the MEN_READ routine during input of an REAL value. If the entered REAL number is less than the RLOW value or greater than the RHIGH value, the input will not be accepted and new input will be requested.
The range checking may be disabled by specifying the RLOW value greater than the RHIGH value. For example:

```
RLOW = 1.0
RHIGH = 0.0
CALL MEN_SETRANGER (ID, RLOW, RHIGH)
```

Note: Range checking is disabled by default.

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER (input)</td>
<td>Identification number to reference the field. Range: 1..99</td>
</tr>
<tr>
<td>RLOW</td>
<td>REAL (input)</td>
<td>Lowest REAL input value allowed for specified field.</td>
</tr>
<tr>
<td>RHIGH</td>
<td>REAL (input)</td>
<td>Highest REAL input value allowed for specified field.</td>
</tr>
</tbody>
</table>

C) Definition of subroutine:

```
SUBROUTINE MEN_SETRANGER (ID, RLOW, RHIGH)
```
Example:

```plaintext
INTEGER ID
REAL RLOW, RHIGH

C Define an input range for field #2:
RLOW = 0.0
RHIGH = 1.23E3
CALL MEN_SETRANGER (2, RLOW, RHIGH)
```

Errors:

If any error occurs, MEN_SETRANGER will display the appropriate error message on the terminal and abort the program.

Remarks:

A menu database must have been created by MEN_OPEN and the field must be defined before calling MEN_SETRANGER.
A) Description:
The MEN_GETRANGER routine returns the currently defined REAL range values of the specified field.

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER</td>
<td>Identification number to reference the field. Range: 1..99</td>
</tr>
<tr>
<td></td>
<td>(input)</td>
<td></td>
</tr>
<tr>
<td>RLOW</td>
<td>REAL</td>
<td>Returns the actual low range value for specified field.</td>
</tr>
<tr>
<td></td>
<td>(output)</td>
<td></td>
</tr>
<tr>
<td>RHIGH</td>
<td>REAL</td>
<td>Returns the actual high range value for specified field.</td>
</tr>
<tr>
<td></td>
<td>(output)</td>
<td></td>
</tr>
</tbody>
</table>

C) Definition of subroutine:

SUBROUTINE MEN_GETRANGER (ID, RLOW, RHIGH)

D) Example:

```
INTEGER ID
REAL RLOW, RHIGH
...
CALL MEN_GETRANGER (2, RLOW, RHIGH)
PRINT *, 'Low value: ', RLOW, ' High value: ', RHIGH
```

E) Errors:
If any error occurs, MEN_GETRANGER will display the appropriate error message on the terminal and abort the program.

F) Remarks:
A menu database must have been created by MEN_OPEN and the field must be defined before calling MEN_GETRANGER.
### Name: MEN_CLEAR  
**Library:** libmen77.a  
**Class:** MEN

### Purpose:
Clears a previous defined menu field

### Created: 27.2.93  
**Revised:**  
**Version:** 1.2

### Author(s): H. Kroiss  
**Phone:** 1308

### Type of program:
- [ ] Main  
- [ * ] Subroutine  
- [ ] Function

### System:
- [ * ] UNIX  
- [ * ] VAX/VMS  
- [ ]

### Documentation:
- [ * ] Help  
- [ ]

#### A) Description:

The MEN_CLEAR routine clears a previous defined menu field in the menu database. All definitions will be erased and the field will be declared as not defined.

If the specified menu field was not defined, no error will occur during MEN_CLEAR call.

#### B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER</td>
<td>Identification number to reference (input)</td>
</tr>
</tbody>
</table>
<pre><code>                         |                                               | the field. Range: 1..99                       |
</code></pre>

#### C) Definition of subroutine:

```c
SUBROUTINE MEN_CLEAR (ID)
```

#### D) Example:

```c
C We assume, that menu field 5 was previous defined:
C
CALL MEN_CLEAR (5)
C Now menu field 5 does no longer exist !
```

#### E) Errors:

If any error occurs, MEN_CLEAR will display the appropriate error message on the terminal and abort the program.

#### F) Remarks:

A menu database must have been created by MEN_OPEN.
A) Description:

The MEN_MOVE routine transfers all specified information of menu field number "ID" to menu field number "FID". This is an easy way to change dynamically menu field numbers.

NOTE: 1. An existing menu field (DSTID) may be overwritten by the MEN_MOVE operation !

2. The content of the old menu field (SRCID) will be cleared !

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRCID</td>
<td>INTEGER (input)</td>
<td>Source menu field identification number. Range: 1 .. 99</td>
</tr>
<tr>
<td>DSTID</td>
<td>INTEGER (input)</td>
<td>Destination menu field identification number. Range: 1 .. 99</td>
</tr>
</tbody>
</table>

C) Definition of subroutine:

SUBROUTINE MEN_MOVE (SRCID, DSTID)

D) Example:

C Content of field number 5 should be moved to field 10:

C

CALL MEN_MOVE (5, 10)

E) Errors:

If any error occurs, MEN_MOVE will display the appropriate error message on the terminal and abort the program.

F) Remarks:

A menu database must have been created by MEN_OPEN and the field must be defined before calling MEN_MOVE.
Name: MEN_NEXT  Library: libmen77.a  Class: MEN

Purpose: Find the next input field and move the cursor to it

Created: 27.2.93  Revised:  Version: 1.2

Author(s): H. Kroiss  Phone: 1308

Type of program:  [ ] Main  [*] Subroutine  [ ] Function

System:  [*] UNIX  [*] VAX/VMS  [ ]

Documentation:  [*] Help  [*] Manual  [ ]

**A) Description:**

The MEN_NEXT routine searches the next geographical input field depending on the last input field position and the given cursor direction.

The found field will be displayed in reverse mode and the corresponding field number will be returned (FID).

On beginning of a menu program the starting input field is set to zero by default. Therefore the first field enabled for input (I/O-direction parameter "I" or "B") will be selected by the MEN_NEXT call. If you want to start with another input field, the desired field number should be set by MEN_SETID.

The cursor key direction selected by VT_READKEY or VT_SOFTKEY may be used directly as input for the KEY parameter on MEN_NEXT routine (see example).

**B) Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY</td>
<td>INTEGER</td>
<td>Desired cursor direction:</td>
</tr>
<tr>
<td></td>
<td>(input)</td>
<td>9 = Cursor UP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 = Cursor DOWN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11 = Cursor LEFT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12 = Cursor RIGHT</td>
</tr>
<tr>
<td>FID</td>
<td>INTEGER</td>
<td>Next input field found depending on current field and selected KEY.</td>
</tr>
<tr>
<td></td>
<td>(output)</td>
<td>Zero, if no input field is defined</td>
</tr>
</tbody>
</table>

**C) Definition of subroutine:**

```
SUBROUTINE MEN_NEXT (KEY, FID)
```
D) Example:

A typical application of the MEN_NEXT routine is the selection and reading operation of input fields:

```
INTEGER KEY, FID
CHARACTER*8 KEYNAME(8)

KEYNAME(1) = 'ENTER'
KEYNAME(2) = '
KEYNAME(3) = '
KEYNAME(4) = '
KEYNAME(5) = '
KEYNAME(6) = '
KEYNAME(7) = '
KEYNAME(8) = 'EXECUTE'
```

50 CALL VT_SOFTKEY (KEYNAME, KEY)
GOTO (51,50,50,50,50,50,50,99,52,52,52,52) KEY
51 CALL MEN_READ (FID)
GOTO 50
52 CALL MEN_NEXT (KEY, FID)
GOTO 50
99 CONTINUE

C Input operations finished!
C ----------------------------------
```

E) Errors:

If any error occurs, MEN_NEXT will display the appropriate error message on the terminal and abort the program.

F) Remarks:

A menu database must have been created by MEN_OPEN and any input field must be defined before calling MEN_NEXT.
A) Description:

The purpose of the MEN_SETTOGGLE routine is to select any
toggle element available in the predefined toggle list and
set this item into the specified menu field for the first
time. This is necessary, if the user want to start on a
toggle index other than the default index (1). A further
reason to use MEN_SETTOGGLE is to initialize the default
(CHARACTER) value of the toggle menu field desired, because
the menu field value is undefined until the first call to
MEN_TOGGLE have been made.
The maximum number of toggle elements contained in the
toggle list must be specified by TOGNR.

NOTE: The MEN_SETTOGGLE routine may only be executed on a
field declared as CHARACTER*8!

B) Parameters:

Parameter | Type | Meaning
-----------|------|---------------------------------------------
ID         | INTEGER | Identification number to reference
           | (input) | the desired field. Range: 1..99
TOGLST     | CHARACTER*8(TOGNR) | List of TOGNR toggle entries
           | (input) | with n characters each
TOGNR      | INTEGER | Maximum number of toggle entries
           | (input) | in toggle list TOGLST
TOGIND     | INTEGER | Number of toggle list element to
           | (output) | define the starting toggle value
C) Definition of subroutine:

SUBROUTINE MEN_SETTOGGLE (ID, TOGLST, TOGNR, TOGIND)

D) Example: For details see "men_example6.f"

C Example for definition of a toggle list:

CHARACTER*10 TOGLST(5)
CHARACTER TOGDIR
INTEGER ID,TOGNR,TOGIND

TOGLST(1) = 'Print'
TOGLST(2) = 'Submit'
TOGLST(3) = 'Delete'
TOGLST(4) = 'Initialize'
TOGLST(5) = 'Copy'
TOGNR = 5

C Start toggle operation on the 3. element ('Delete'):
CALL MEN_SETTOGGLE (ID,TOGLST,TOGNR,3)

C
CALL MEN_TOGGLE (ID,'U',TOGLST,TOGNR,TOGIND)

E) Errors:

If any error occurs, MEN_SETTOGGLE will display the appropriate error message on the terminal and abort the program.

F) Remarks:

A menu database must have been created by MEN_OPEN and the field must be defined (as CHARACTER) before calling MEN_SETTOGGLE.
A) Description:

The MEN_TOGGLE routine gets the next token (character string) depending on the selected direction (UP or DOWN) from a pre-defined toggle list, displays the token on screen field number ID and returns the current index of toggle list (TOGIND).

The maximum number of entries contained in the toggle list must be specified by TOGNR.

On beginning of a menu program the starting toggle index of a character field is set to 1 by default. The toggle index returned by MEN_TOGGLE may be used to decide which operation should be performed as next.

NOTE: The MEN_TOGGLE routine may only be executed on a field, which was declared as CHARACTER*n field!

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER (input)</td>
<td>Identification number to reference the desired field. Range: 1..99</td>
</tr>
<tr>
<td>TOGDIR</td>
<td>CHARACTER (input)</td>
<td>Desired toggle direction: 'U' means: toggle UP 'D' means: toggle DOWN</td>
</tr>
<tr>
<td>TOGLST</td>
<td>CHARACTER*n(TOGNR) (input)</td>
<td>List of TOGNR toggle entries with n characters each</td>
</tr>
<tr>
<td>TOGNR</td>
<td>INTEGER (input)</td>
<td>Maximum number of toggle entries in toggle list TOGLST</td>
</tr>
<tr>
<td>TOGIND</td>
<td>INTEGER (output)</td>
<td>Index of current toggle element</td>
</tr>
</tbody>
</table>
C) Definition of subroutine:

```
SUBROUTINE MEN_TOGGLE (ID, TOGDIR, TOGLST, TOGNR, TOGIND)
```

D) Example: For details see "men_example6.f"

```
C Example for definition of a toggle list:
C
CHARACTER*10 TOGLST(5)
CHARACTER TOGDIR
INTEGER ID,TOGNR,TOGIND

TOGLST(1) = 'Print'
TOGLST(2) = 'Submit'
TOGLST(3) = 'Delete'
TOGLST(4) = 'Initialize'
TOGLST(5) = 'Copy'
TOGNR = 5

CALL MEN_TOGGLE (ID,'U',TOGLST,TOGNR,TOGIND)
```

E) Errors:

If any error occurs, MEN_TOGGLE will display the appropriate error message on the terminal and abort the program.

F) Remarks:

A menu database must have been created by MEN_OPEN and the field must be defined (as CHARACTER) before calling MEN_TOGGLE.
A) Description:

The MEN_SAVRES routine is used to save the current menu definitions and field values into a temporary buffer or to restore a previous saved menu database. The temporary buffer must be great enough to store the whole menu database (32KB).

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPT</td>
<td>CHARACTER*3</td>
<td>Option for desired operation:</td>
</tr>
<tr>
<td></td>
<td>(input)</td>
<td>'SAV' Save current menu database</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'RES' Restore saved menu database</td>
</tr>
<tr>
<td>SAVBUF</td>
<td>INTEGER-array</td>
<td>Data buffer to store menu database</td>
</tr>
<tr>
<td></td>
<td>(input/output)</td>
<td>input: for RESTORE operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>output: for SAVE operation</td>
</tr>
</tbody>
</table>

C) Definition of subroutine:

SUBROUTINE MEN_SAVRES (OPT, SAVBUF)

D) Example:

```
CALL MEN_SAVRES ('SAV', SAVBUF)
```

C 1. Save the current menu database on SAVBUF:

C 2. Do anything...

C 3. Restore old menu database from SAVBUF:

```
CALL MEN_SAVRES ('RES', SAVBUF)
```
E) Errors:

If any error occurs, MEN_SAVRES will display the appropriate error message on the terminal and abort the program.

F) Remarks:

A menu database must have been created by MEN_OPEN before the first call to MEN_SAVRES performed.
### Name: MEN_GETI  
### Library: libmen77.a  
### Class: MEN

**Purpose:** Get an INTEGER value from menu database

**Created:** 27.2.93  
**Revised:**  
**Version:** 1.2

**Author(s):** H. Kroiss  
**Phone:** 1308

**Type of program:**  
[ ] Main  
[*] Subroutine  
[ ] Function

**System:**  
[*] UNIX  
[*] VAX/VMS  
[ ]

**Documentation:**  
[*] Help  
[*] Manual  
[ ]

---

### A) Description:

The **MEN_GETI** routine is used to get an INTEGER value from a selected menu database field. The function of **MEN_GETI** is very similar to **MEN_GET**, but only INTEGER values may be returned to the user program. If the specified field does not have INTEGER format, then **MEN_GETI** tries to convert the given field value to INTEGER format. If this is not possible, **MEN_GETI** returns a fatal error in **STAT** parameter.

In the most cases the procedure **MEN_GETI** is used for non-interactive dialog purposes.

**NOTE:** **MEN_GETI** will only return the desired value from the menu database, but will not perform a read function from the terminal screen field!

### B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER</td>
<td>Identification number to reference the desired field. Range: 1..99</td>
</tr>
<tr>
<td></td>
<td>(input)</td>
<td></td>
</tr>
<tr>
<td>IVAL</td>
<td>INTEGER</td>
<td>Integer variable to get value from menu database.</td>
</tr>
<tr>
<td></td>
<td>(output)</td>
<td></td>
</tr>
<tr>
<td>STAT</td>
<td>INTEGER</td>
<td>Returned status of operation, see E. Errors</td>
</tr>
<tr>
<td></td>
<td>(output)</td>
<td></td>
</tr>
</tbody>
</table>

### C) Definition of subroutine:

```
SUBROUTINE MEN_GETI (ID, IVAL, STAT)
```
D) Example:

```c
C Get an INTEGER value from menu file "men_example.men"
C field number 1:
C--------------------------------------------------------
INTEGER IVAL, STAT
LOGICAL ERR
C
C 1. Read saved menu file:
C----------------------------
CALL MEN_OPEN ('men_example.men', ERR)
C
C 2. Get INTEGER value from menu file:
C-------------------------------------
CALL MEN_GETI (1, IVAL, STAT)
IF (STAT .GT. 3) PRINT *, 'Error:', STAT
PRINT *, 'Integer value of field 1:', IVAL
```

E) Errors:

After calling MEN_GETI, the parameter STAT contains the appropriate error code:

<table>
<thead>
<tr>
<th>STAT</th>
<th>Error</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
<td>success</td>
</tr>
<tr>
<td>1</td>
<td>None</td>
<td>value was modified</td>
</tr>
<tr>
<td>2</td>
<td>Warning</td>
<td>field format does not correspond</td>
</tr>
<tr>
<td>3</td>
<td>Warning</td>
<td>data type of field has been changed</td>
</tr>
<tr>
<td>4</td>
<td>Fatal</td>
<td>conversion of field format not possible</td>
</tr>
<tr>
<td>5</td>
<td>Fatal</td>
<td>only input allowed on field</td>
</tr>
<tr>
<td>6</td>
<td>Fatal</td>
<td>specified field is not defined</td>
</tr>
<tr>
<td>7</td>
<td>Fatal</td>
<td>illegal field number (Range 1..100)</td>
</tr>
</tbody>
</table>

F) Remarks:

A menu database must have been created by MEN_OPEN and the field must be defined before calling MEN_GETI.
A) Description:

The MEN_GETR routine is used to get a REAL value from a selected menu database field. The function of MEN_GETR is very similar to MEN_GET, but only REAL values may be returned to the user program. If the specified field does not have REAL format, then MEN_GETR tries to convert the given field value to REAL format. If this is not possible, MEN_GETR returns a fatal error in STAT parameter.

In the most cases the procedure MEN_GETR is used for non-interactive dialog purposes.

NOTE: MEN_GETR will only return the desired value from the menu database, but will not perform a read function from the terminal screen field!

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER</td>
<td>Identification number to reference the desired field. Range: 1..99</td>
</tr>
<tr>
<td>RVAL</td>
<td>REAL</td>
<td>Real variable to get value from menu database.</td>
</tr>
<tr>
<td>STAT</td>
<td>INTEGER</td>
<td>Returned status of operation, see E. Errors</td>
</tr>
</tbody>
</table>

C) Definition of subroutine:

SUBROUTINE MEN_GETR (ID, RVAL, STAT)
D) Example:

```c
C Get a REAL value from menu file "men_example.men"
C field number 2:
C
C
INTEGER STAT
REAL RVAL
LOGICAL ERR
C
C 1. Read saved menu file:
C----------------------------------------------
CALL MEN_OPEN ('men_example.men', ERR)
C
C 2. Get REAL value from menufile:
C----------------------------------------------
CALL MEN_GETR (2, RVAL, STAT)
IF (STAT .GT. 3) PRINT *, 'Error:', STAT
PRINT *, 'Real value of field 2:', RVAL
```

E) Errors:

After calling MEN_GETR, the parameter STAT contains the appropriate error code:

```
<table>
<thead>
<tr>
<th>STAT</th>
<th>Error</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
<td>success</td>
</tr>
<tr>
<td>1</td>
<td>None</td>
<td>value was modified</td>
</tr>
<tr>
<td>2</td>
<td>Warning</td>
<td>field format does not correspond</td>
</tr>
<tr>
<td>3</td>
<td>Warning</td>
<td>data type of field has been changed</td>
</tr>
<tr>
<td>4</td>
<td>Fatal</td>
<td>conversion of field format not possible</td>
</tr>
<tr>
<td>5</td>
<td>Fatal</td>
<td>only input allowed on field</td>
</tr>
<tr>
<td>6</td>
<td>Fatal</td>
<td>specified field is not defined</td>
</tr>
<tr>
<td>7</td>
<td>Fatal</td>
<td>illegal field number (Range 1..100)</td>
</tr>
</tbody>
</table>
```

F) Remarks:

A menu database must have been created by MEN_OPEN and the field must be defined before calling MEN_GETR.
A) Description:

The MEN_GETC routine is used to get a CHARACTER*n value from a selected menu database field. The function of MEN_GETC is very similar to MEN_GET, but only CHARACTER*n values may be returned to the user program. If the specified field does not have CHARACTER*n format, then MEN_GETC returns a fatal error in STAT parameter.

In the most cases the procedure MEN_GETC is used for non-interactive dialog purposes.

NOTE: MEN_GETC will only return the desired value from the menu database, but will not perform a read function from the terminal screen field!

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER</td>
<td>Identification number to reference the desired field. Range: 1..99</td>
</tr>
<tr>
<td>CVAL</td>
<td>CHARACTER*n</td>
<td>Character variable to get value from menu database.</td>
</tr>
<tr>
<td>STAT</td>
<td>INTEGER</td>
<td>Returned status of operation, see E. Errors</td>
</tr>
</tbody>
</table>

C) Definition of subroutine:

```
SUBROUTINE MEN_GETC (ID, CVAL, STAT)
```
D) Example:

```c
C Get a CHARACTER*n value from menu file "men_example.men"
C field number 3:
C
C
INTEGER STAT
CHARACTER*80 CVAL
LOGICAL ERR
C
C 1. Read saved menu file:
C
CALL MEN_OPEN ('men_example.men', ERR)
C
C 2. Get CHARACTER value from menufile:
C
CALL MEN_GETC (3, CVAL, STAT)
IF (STAT .GT. 3) PRINT *, 'Error:', STAT
PRINT *, 'Character value of field 3:', CVAL
```

E) Errors:

After calling MEN_GETC, the parameter STAT contains the appropriate error code:

<table>
<thead>
<tr>
<th>STAT</th>
<th>Error</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
<td>success</td>
</tr>
<tr>
<td>1</td>
<td>None</td>
<td>value was modified</td>
</tr>
<tr>
<td>2</td>
<td>Warning</td>
<td>field format does not correspond</td>
</tr>
<tr>
<td>3</td>
<td>Warning</td>
<td>data type of field has been changed</td>
</tr>
<tr>
<td>4</td>
<td>Fatal</td>
<td>conversion of field format not possible</td>
</tr>
<tr>
<td>5</td>
<td>Fatal</td>
<td>only input allowed on field</td>
</tr>
<tr>
<td>6</td>
<td>Fatal</td>
<td>specified field is not defined</td>
</tr>
<tr>
<td>7</td>
<td>Fatal</td>
<td>illegal field number (Range 1..100)</td>
</tr>
</tbody>
</table>

F) Remarks:

A menu database must have been created by MEN_OPEN and the field must be defined before calling MEN_GETC.
A) Description:

The MEN_SETI routine is used to set an INTEGER value into a selected menu database field. The function of MEN_SETI is very similar to MEN_SET, but only INTEGER values may be set by the user program. If the specified field does not have INTEGER format, then MEN_SETI changes the field format to INTEGER and returns the status "3" on STAT parameter.

In the most cases the procedure MEN_SETI is used for non-interactive dialog purposes.

NOTE: MEN_SETI will only set the desired value into the menu database, but will not perform a write function to the terminal screen field!

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER</td>
<td>Identification number to reference the desired field. Range: 1..99</td>
</tr>
<tr>
<td>IVAL</td>
<td>INTEGER</td>
<td>Integer value to set into menu database.</td>
</tr>
<tr>
<td>STAT</td>
<td>INTEGER</td>
<td>Returned status of operation, see E. Errors</td>
</tr>
</tbody>
</table>

C) Definition of subroutine:

SUBROUTINE MEN_SETI (ID, IVAL, STAT)
D) Example:

```c
C Set an INTEGER value into menu file "men_example.men"
C field number 1:

---------------------------------------------------------------------
INTEGER        IVAL,STAT
LOGICAL        ERR
C
C 1. Read saved menu file:
C ---------------------------------------------
CALL MEN_OPEN ("men_example.men",ERR)

C 2. Set new INTEGER value into menu database:
C ---------------------------------------------
IVAL = 1234
CALL MEN_SETI (1, IVAL, STAT)
IF (STAT .GT. 3) WRITE *, 'Error:', STAT

C 3. Write back modified menu database:
C ---------------------------------------------
CALL MEN_SAVE ('men_example.men')
```

E) Errors:

After calling MEN_SETI, the parameter STAT contains the appropriate error code:

<table>
<thead>
<tr>
<th>STAT</th>
<th>Error</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
<td>success</td>
</tr>
<tr>
<td>1</td>
<td>None</td>
<td>value was modified</td>
</tr>
<tr>
<td>2</td>
<td>Warning</td>
<td>field format does not correspond</td>
</tr>
<tr>
<td>3</td>
<td>Warning</td>
<td>data type of field has been changed</td>
</tr>
<tr>
<td>4</td>
<td>Fatal</td>
<td>conversion of field format not possible</td>
</tr>
<tr>
<td>5</td>
<td>Fatal</td>
<td>only input allowed on field</td>
</tr>
<tr>
<td>6</td>
<td>Fatal</td>
<td>specified field is not defined</td>
</tr>
<tr>
<td>7</td>
<td>Fatal</td>
<td>illegal field number (Range 1..100)</td>
</tr>
</tbody>
</table>

F) Remarks:

A menu database must have been created by MEN_OPEN and the field must be defined before calling MEN_SETI.
### A) Description:

The MEN_SETR routine is used to set a REAL value into a selected menu database field. The function of MEN_SETR is very similar to MEN_SET, but only REAL values may be set by the user program. If the specified field does not have REAL format, then MEN_SETR changes the field format to REAL and returns the status "3" on STAT parameter.

In the most cases the procedure MEN_SETR is used for non-interactive dialog purposes.

**NOTE:** MEN_SETR will only set the desired value into the menu database, but will not perform a write function to the terminal screen field!

### B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER</td>
<td>Identification number to reference the desired field. Range: 1..99</td>
</tr>
<tr>
<td>RVAL</td>
<td>REAL</td>
<td>REAL value to set into menu database.</td>
</tr>
<tr>
<td>STAT</td>
<td>INTEGER</td>
<td>Returned status of operation, see E. Errors</td>
</tr>
</tbody>
</table>

### C) Definition of subroutine:

```fortran
SUBROUTINE MEN_SETR (ID, RVAL, STAT)
```
D) Example:

```c
C Set a REAL value into menu file "men_example.men"
C field number 2:
---------------------------------------------------------------
INTEGER STAT
REAL RVAL
LOGICAL ERR
C
C 1. Read saved menu file:
C --------------------------------------------
CALL MEN_OPEN ('men_example.men', ERR)
C
C 2. Set new REAL value into menu database:
C ---------------------------------------------
RVAL = 12.34E-2
CALL MEN_SETR (2, RVAL, STAT)
IF (STAT .GT. 3) PRINT *, 'Error:', STAT
C
C 3. Write back modified menu database:
C -------------------------------------------
CALL MEN_SAVE ('men_example.men')
```

E) Errors:

After calling MEN_SETR, the parameter STAT contains the appropriate error code:

<table>
<thead>
<tr>
<th>STAT</th>
<th>Error</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
<td>success</td>
</tr>
<tr>
<td>1</td>
<td>None</td>
<td>value was modified</td>
</tr>
<tr>
<td>2</td>
<td>Warning</td>
<td>field format does not correspond</td>
</tr>
<tr>
<td>3</td>
<td>Warning</td>
<td>data type of field has been changed</td>
</tr>
<tr>
<td>4</td>
<td>Fatal</td>
<td>conversion of field format not possible</td>
</tr>
<tr>
<td>5</td>
<td>Fatal</td>
<td>only input allowed on field</td>
</tr>
<tr>
<td>6</td>
<td>Fatal</td>
<td>specified field is not defined</td>
</tr>
<tr>
<td>7</td>
<td>Fatal</td>
<td>illegal field number (Range 1..100)</td>
</tr>
</tbody>
</table>

F) Remarks:

A menu database must have been created by MEN_OPEN and the field must be defined before calling MEN_SETR.
A) Description:

The MEN_SETC routine is used to set a CHARACTER*n value into a selected menu database field. The function of MEN_SETC is very similar to MEN_SET, but only CHARACTER*n values may be set by the user program. If the specified field does not have CHARACTER*n format, then MEN_SETC changes the field format to CHARACTER* and returns the status "3" on STAT parameter.

In most cases the procedure MEN_SETC is used for non-interactive dialog purposes.

NOTE: MEN_SETC will only set the desired value into the menu database, but will not perform a write function to the terminal screen field!

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER</td>
<td>Identification number to reference the desired field. Range: 1..99</td>
</tr>
<tr>
<td>CVAL</td>
<td>CHARACTER*n</td>
<td>Character value to set into menu database.</td>
</tr>
<tr>
<td>STAT</td>
<td>INTEGER</td>
<td>Returned status of operation, see E. Errors</td>
</tr>
</tbody>
</table>

C) Definition of subroutine:

```
SUBROUTINE MEN_SETC (ID, CVAL, STAT)
```
D) Example:

```c
C Set a CHARACTER*n value into menu file "men_example.men"
C field number 3:
C
C INTEGER STAT
C CHARACTER*80 CVAL
C LOGICAL ERR
C
C 1. Read saved menu file:
C ------------------------------------------
C CALL MEN_OPEN ("men_example.men",ERR)
C
C 2. Set new CHARACTER*n value into menu database:
C ------------------------------------------
C CVAL = 'This is a new text'
C CALL MEN_SETC (3, CVAL, STAT)
C IF (STAT .GT. 3) PRINT *, 'Error:', STAT
C
C 3. Write back modified menu database:
C ------------------------------------------
C CALL MEN_SAVE ('men_example.men')
```

E) Errors:

After calling MEN_SETC, the parameter STAT contains the appropriate error code:

<table>
<thead>
<tr>
<th>STAT</th>
<th>Error</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
<td>success</td>
</tr>
<tr>
<td>1</td>
<td>None</td>
<td>value was modified</td>
</tr>
<tr>
<td>2</td>
<td>Warning</td>
<td>field format does not correspond</td>
</tr>
<tr>
<td>3</td>
<td>Warning</td>
<td>data type of field has been changed</td>
</tr>
<tr>
<td>4</td>
<td>Fatal</td>
<td>conversion of field format not possible</td>
</tr>
<tr>
<td>5</td>
<td>Fatal</td>
<td>only input allowed on field</td>
</tr>
<tr>
<td>6</td>
<td>Fatal</td>
<td>specified field is not defined</td>
</tr>
<tr>
<td>7</td>
<td>Fatal</td>
<td>illegal field number (Range 1..100)</td>
</tr>
</tbody>
</table>

F) Remarks:

A menu database must have been created by MEN_OPEN and the field must be defined before calling MEN_SETC.
A) Description:

The VT_CLEARSCREEN routine clears the terminal screen. This subroutine may be used for general purposes on VT200-compatible I/O devices.

B) Parameters: none

C) Definition of subroutine:

```
SUBROUTINE VT_CLEARSCREEN
```

D) Example:

```
CALL VT_CLEARSCREEN
```

E) Errors: none

F) Remarks: none
A) Description:

The VT_RESETSCREEN routine resets the cursor position and the scroll region. Furthermore the terminal screen display will be cleared, if CLEAR is true. This subroutine may be used for general purposes on all VT200-compatible I/O devices.

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEAR</td>
<td>LOGICAL*1 (input)</td>
<td>Clear Flag, if set the terminal screen will be erased</td>
</tr>
</tbody>
</table>

C) Definition of subroutine:

```plaintext
SUBROUTINE VT_RESETSCREEN (CLEAR)
```

D) Example:

```plaintext
C Reset scroll region and cursor position and erase the VT200 compatible screen display:
C CALL VT_RESETSCREEN (.TRUE.)
```

E) Errors: none

F) Remarks: none
The VT_SETREGION routine is used to declare a scroll region on a VT200-compatible terminal display. The region on which line scrolling is enabled, will be defined by a start line number (SLIN) and an end line number (ELIN). After calling VT_SETREGION, all following terminal output will be scrolled on the specified region.

VT_SETREGION may be used for general purposes on all VT200-compatible I/O devices.

Parameter | Type | Meaning
---|---|---
SLIN | INTEGER (input) | Line number on which the scroll region should begin. Range: 1..24
ELIN | INTEGER (input) | Line number on which the scroll region should end. Range: 1..24

NOTE: SLIN may not be greater than ELIN!

Example:

Set a scroll region between line 10 and line 17:

```
call vt_setregion (10, 17)
```
A) Description:

The VT_CLEARREGION routine is used to clear a special part (region) of a VT200-compatible terminal display. The region on which the display should be cleared, will be defined by a start line number (SLIN) and an end line number (ELIN). After calling VT_CLEARREGION, the specified part of screen display will be erased.

VT_CLEARREGION may be used for general purposes on all VT200-compatible I/O devices.

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLIN</td>
<td>INTEGER (input)</td>
<td>Start line number for clear region. Range: 1..24</td>
</tr>
<tr>
<td>ELIN</td>
<td>INTEGER (input)</td>
<td>End line number for clear region. Range: 1..24</td>
</tr>
</tbody>
</table>

NOTE: SLIN may not be greater than ELIN!

C) Definition of subroutine:

SUBROUTINE VT_CLEARREGION (SLIN, ELIN)

D) Example:

C Clear a display region between line 10 and line 17:

CALL VT_CLEARREGION (10, 17)

E) Errors: none

F) Remarks: none
A) Description:

The VT_SET_CURSOR routine is used to set the cursor on a VT200-compatible terminal display to the specified display position. After calling VT_SET_CURSOR, all following terminal input or output will be done at this screen position.

VT_SET_CURSOR may be used for general purposes on all VT200-compatible I/O devices.

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIN</td>
<td>INTEGER</td>
<td>Line number to which the cursor should be moved. Range:1..24</td>
</tr>
<tr>
<td></td>
<td>(input)</td>
<td></td>
</tr>
<tr>
<td>ROW</td>
<td>INTEGER</td>
<td>Row number to which the cursor should be moved. Range:1..80</td>
</tr>
<tr>
<td></td>
<td>(input)</td>
<td></td>
</tr>
</tbody>
</table>

C) Definition of subroutine:

SUBROUTINE VT_SET_CURSOR (LIN, ROW)

D) Example:

C    Move the cursor to line 10, row 25:
C    -----------------------------------
CALL VT_SET_CURSOR (10, 25)

E) Errors: none

F) Remarks: none
A) Description:

The VT_READKEY allows the user to read the function keys F7 up to F14 and the cursor keys (UP, DOWN, LEFT, RIGHT) of a VT200-compatible terminal keyboard. After calling VT_READKEY, the user program is stopped and waiting for input of F7-F14 or cursorkeys. In this state the keyboard is locked for any other keys. After pressing a legal key, VT_READKEY returns the corresponding KEY number:

<table>
<thead>
<tr>
<th>KEY</th>
<th>corresponding keyboard function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Function key: F7</td>
</tr>
<tr>
<td>2</td>
<td>Function key: F8</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>8</td>
<td>Function key: F14</td>
</tr>
<tr>
<td>9</td>
<td>Cursor key: UP</td>
</tr>
<tr>
<td>10</td>
<td>Cursor key: DOWN</td>
</tr>
<tr>
<td>11</td>
<td>Cursor key: LEFT</td>
</tr>
<tr>
<td>12</td>
<td>Cursor key: RIGHT</td>
</tr>
</tbody>
</table>

VT_READKEY may be used for general purposes on all VT200-compatible I/O devices.

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY</td>
<td>INTEGER</td>
<td>Number of entered keyboard function (output)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(see table above). Range: 1..12</td>
</tr>
</tbody>
</table>
C) Definition of subroutine:

SUBROUTINE VT_READKEY (KEY)

D) Example:

INTEGER KEY

CALL VT_READKEY (KEY)

PRINT *, 'Keyboard function entered:', KEY

E) Errors: none

F) Remarks: none
A) Description:

A 'softkey' means a field of up to 8 characters which will be displayed on terminal screen on the last line over the corresponding function key of the keyboard. The user may assign any names to the softkeys and also change the names during program execution.

The VT_SOFTKEY routine displays 8 'softkeys' on line 24 of the terminal screen and waits for input of function keys F7 up to F14 or cursor keys (UP, DOWN, LEFT, RIGHT) of a VT200-compatible keyboard. While waiting for input, all other keys of the keyboard are locked. After pressing a legal key, VT_SOFTKEY returns the corresponding KEY number:

<table>
<thead>
<tr>
<th>KEY</th>
<th>corresponding keyboard function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Functionkey: F7</td>
</tr>
<tr>
<td>2</td>
<td>Functionkey: F8</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>8</td>
<td>Functionkey: F14</td>
</tr>
<tr>
<td>9</td>
<td>Cursor key: UP</td>
</tr>
<tr>
<td>10</td>
<td>Cursor key: DOWN</td>
</tr>
<tr>
<td>11</td>
<td>Cursor key: LEFT</td>
</tr>
<tr>
<td>12</td>
<td>Cursor key: RIGHT</td>
</tr>
</tbody>
</table>

WARNING: Line 24 of the terminal display will be always overwritten by VT_SOFTKEY! Therefore, menu programs should reserve line 24 for VT_SOFTKEY usage.

VT_SOFTKEY may be used for general purposes on all VT200-compatible I/O devices.
B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>S NAMES</td>
<td>CHARACTER*8</td>
<td>Names of softkeys to display on terminal display line 24.</td>
</tr>
<tr>
<td></td>
<td>(input)</td>
<td>(see table above). Range: 1..12</td>
</tr>
<tr>
<td>KEY</td>
<td>INTEGER</td>
<td>Number of entered keyboard function</td>
</tr>
<tr>
<td></td>
<td>(output)</td>
<td></td>
</tr>
</tbody>
</table>

C) Definition of subroutine:

SUBROUTINE VT_SOFTKEY (S NAMES, KEY)

D) Example:

INTEGER KEY
CHARACTER*8 S NAMES(8)

C 1. Define names for softkeys:

S NAMES(1) = 'Key-No.1'
S NAMES(2) = 'Key-No.2'
S NAMES(3) = 'Key-No.3'
S NAMES(4) = 'Key-No.4'
S NAMES(5) = 'Key-No.5'
S NAMES(6) = 'Key-No.6'
S NAMES(7) = 'Key-No.7'
S NAMES(8) = 'Key-No.8'

C 2. Display softkeys and wait for input of a function key:

CALL VT_SOFTKEY (S NAMES, KEY)
PRINT *,'Keyboard function entered:', KEY

E) Errors: none

F) Remarks: none
A) Description:

VT_READSTRING routine reads a string of characters beginning at the current cursor position up to SLEN characters. If a 'RETURN' was encountered the input will be finished and the actual length will be returned on RLEN.

The most important advantages in comparison with FORTRAN READ statements are:

- VT_READSTRING routine performs neither 'Carrage Return' nor 'Line Feed' operations,
- Illegal input (like control characters) will be eliminated
- The terminal input will be finished automatically if the desired number of characters (SLEN) have been entered.

VT_READSTRING may be used for general purposes on all VT200-compatible I/O devices.

B) Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRING</td>
<td>CHARACTER*</td>
<td>Character buffer to receive the input string.</td>
</tr>
<tr>
<td>SLEN</td>
<td>INTEGER</td>
<td>Number of desired characters to read</td>
</tr>
<tr>
<td>RLEN</td>
<td>INTEGER</td>
<td>Number of characters actually entered on terminal</td>
</tr>
</tbody>
</table>
C) Definition of subroutine:

```
SUBROUTINE VT_READSTRING (STRING, SLEN, RLEN)
```

D) Example:

```
INTEGER SLEN, RLEN
CHARACTER*80 STRING
C
C Read a string of characters (max. 10) on the
current cursor position:
C
C-----------------------------------------------------
CALL VT_READSTRING (STRING, 10, RLEN)
PRINT *, 'Entered string:', STRING(1:RLEN)
PRINT *, 'Length of string:', RLEN
```

E) Errors: none

F) Remarks: none
G. SUMMARY

This summary gives a short description of each routine and a summary of the call interface. More details are available in the full descriptions contained in section E of this manual.

G.1 Declarations:

```
INTEGER ID, FID, LIN, ROW, LNG, SLIN, ELIN, STAT, KEY, MEN_ID
INTEGER IVAL, ILOW, IHIGH, TOGB, TOGIN, SLEN, RLEN
REAL RVAL, RLOW, RHIGH
LOGICAL ERR, CLEAR
CHARACTER FMT, IO, ATT, TOGDIR
CHARACTER*3 OPT
CHARACTER*8 SNAME
CHARACTER*22 NAME
CHARACTER*80 CVAR, FNAME, BLINE, HLINE, STRING
CHARACTER*1n TOGLst(m)
```

G.2 Basic menu routines:

```
CALL MEN_OPEN (FNAME, ERR)
Reads a previous saved menufile named FNAME into menu database. ERR = .TRUE., if file FNAME not found.
If FNAME = ' ' then only the MEN data base will be initialized.

CALL MEN_SAVE (FNAME)
Save the current menu definitions and values in a menufile named FNAME.

CALL MEN_DEFINE (ID, NAME, LIN, ROW, LNG, FMT, IO, ATT)
Define menu Field number ID with screen Positions, Length, Format, I/O Direction and Attribute.

CALL MEN_INFO (ID, NAME, LIN, ROW, LNG, FMT, IO, ATT)
Get information about menu Field number ID: Screen Positions, Length, Format, I/O Direction and Attribute.

CALL MEN_MODIFY (ID, OPT, IVAL, CVAR)
Modify previous defined menu field parameters

CALL MEN_SET (ID, IVAL, RVAL, CVAR)
Set a value (INTEGER, REAL, CHARACTER) into menu field database number ID.

CALL MEN_GET (ID, IVAL, RVAL, CVAR)
Get a value (INTEGER, REAL, CHARACTER) from menu field database number ID.
```
CALL MEN_READ (ID)  Read a value from terminal menu field number ID into menu field database.

CALL MEN_WRITE (ID) Display a value from menu field database number ID on terminal.
ID = 0 means: all defined fields.

G.3 Advanced menu routines:

CALL MEN_REFRESH Refresh all defined 'Background' and menu field information on screen.

CALL MEN_SETID (ID)  Mark field number ID as the first field for possible input.

CALL MEN_GETID (FID) Get the current (or saved) field number
FID = MEN_ID (NAME) Returns the field identification number, which corresponds to the field name "NAME".

CALL MEN_SETBGRD (LIN,BLINE) Set a line of 'Background' characters into menu database

CALL MEN_GETBGRD (LIN,BLINE) Get a line of 'Background' characters from menu database

CALL MEN_WRITEBGRD Display all defined 'Background' lines on terminal screen.

CALL MEN_SETHELP (ID,HLINE) Set a string of characters as help for menu field number ID.

CALL MEN_GETHELP (ID,HLINE) Get the help message string defined for menu field number ID.

CALL MEN_SETRANGEI (ID,ILOW,IHIGH) Define a range of INTEGER values for menu field number ID.

CALL MEN_GETRANGEI (ID,ILOW,IHIGH) Get the currently defined range of INTEGER values of menu field number ID.

CALL MEN_SETRANGER (ID,RLOW,RHIGH) Define a range of REAL values for menu field number ID.

CALL MEN_GETRANGER (ID,RLOW,RHIGH) Get the currently defined range of REAL values of menu field number ID.
CALL MEN_CLEAR (ID)                  Clears the previous defined menu field
database number ID

CALL MEN_MOVE (ID,FID)   Moves all definitions of field number ID
to field number FID. The previous contents
of field ID will be cleared.

CALL MEN_NEXT (KEY,FID)   Find the next field depending on the
current field and the cursor direction
and return the next field FID.
KEY = 9 means Cursor UP
KEY = 10 means Cursor DOWN
KEY = 11 means Cursor LEFT
KEY = 12 means Cursor RIGHT

CALL MEN_SETToggled (ID, TOGLST, TOGNB, TOGIND)  Initialize a toggle field by setting the
TOGIND element of TOGLST into specified
field ID.

CALL MEN_Toggle (ID, TOGDIR, TOGLST, TOGNB, TOGIND)  Toggle character tokens specified at
TOGLST (number of possible tokens:TOGNB)
up (TOGDIR='U') or down (TOGDIR='D') and
return index of toggle list (TOGIND).

CALL MEN_SAVE (OPT, SAVBUF)  Saves and restores a menu database
temporary into or from a local buffer.
OPT = 'SAV' means: save current menu
OPT = 'RES' means: restore current menu
SAVBUF must be declared as follows:
INTEGER SAVBUF(8000)

G.4 Routines to perform simple dialog functions:
---------------------------------------------

CALL MEN_GETI (ID,IVAL,STAT) Get an INTEGER value from menu
database

CALL MEN_GETR (ID,RVAL,STAT) Get a REAL value from menu database

CALL MEN_GETC (ID,CVAL,STAT) Get a CHARACTER*n value from menu
database

CALL MEN_SETI (ID,IVAL,STAT) Set an INTEGER value into menu
database

CALL MEN_SETR (ID,RVAL,STAT) Set a REAL value into menu database

CALL MEN_SETC (ID,CVAL,STAT) Set a CHARACTER*n value into menu
database
G. SUMMARY of MEN Fortran interface routines

G.5 Routines to perform terminal I/O functions

CALL VT_CLEARSCREEN  Clears the terminal screen

CALL VT_RESETSCREEN (CLEAR)  
Reset cursor and scroll region. Clear screen, if CLEAR is true.

CALL VT_SETREGION (SLIN,ELIN)  
Set a scroll region on terminal screen, beginning with SLIN up to ELIN.

CALL VT_CLEARREGION (SLIN,ELIN)  
Clear a region on terminal screen, beginning with SLIN up to ELIN.

CALL VT_SETCURSOR (LIN,ROW)  Set the terminal cursor

CALL VT_READKEY (KEY)  Wait for input of a function key (F7..F14) or cursorkey.

CALL VT_READSTRING (STRING,SLEN,RLEN)  
Read a string of characters beginning at current cursor position up to SLEN characters. If a RETURN was encountered the input will be finished and the actual length will be returned on RLEN. 
Note: Illegal input (Control characters) will be eliminated!

CALL VT_SOFTKEY (SNAMES,KEY)  
Displays 8 Softkeys (SNAMES) on terminal line 24 and wait for input of a function key (F7..F14) or cursorkey.
1. Initialize menu database:

CALL MEN_OPEN (' ', ERR)

2. Define screen fields and set field pointer:

MEN_DEFINE parameters: (ID, NAM, LIN, ROW, LEN, FMT, TYP, ATT)

- ID: identification number of field
- LIN: line number
- ROW: row number
- LEN: maximum length of field in characters
- FMT: format of Field: I, C, R, F
- TYP: type of screen I/O: I, O, B
- ATT: attribute pf field: N, R, B, I, U

CALL MEN_DEFINE (1, 'Field-1', 4, 18, 10, 'F', 'B', 'N')
CALL MEN_DEFINE (2, 'Field-2', 4, 45, 10, 'I', 'B', 'I')
CALL MEN_DEFINE (3, 'Field-3', 5, 18, 20, 'C', 'B', 'U')
CALL MEN_DEFINE (4, 'Field-4', 6, 16, 20, 'C', 'O', 'B')
FID = 1
CALL MEN_SETID (FID)

3. Display screen background:

CALL VT_CLEARSCREEN

PRINT *,' Demonstration program for screen application'
PRINT *,' Real value: ' INTEGER value:'
PRINT *,'Character value:'
PRINT *, 'Only output:'
M E N - Menu and Dialog Software
H.1 Example programs for MEN subroutine package

4. Set default values, if desired:

CALL MEN_SET (1,IVAL,1.234567,CVAL)
CALL MEN_SET (2,123456789,RVAL,CVAL)
CALL MEN_SET (3,IVAL,RVAL,'Toggle Switch')
CALL MEN_SET (4,IVAL,RVAL,'State OK')

5. Display default screen field contents:

CALL MEN_WRITE (0)

6. Define names for softkeys:

KEYNAM(1) = 'CHANGE'
KEYNAM(2) =
KEYNAM(3) =
KEYNAM(4) =
KEYNAM(5) =
KEYNAM(6) =
KEYNAM(7) = 'EXECUTE'
KEYNAM(8) = 'EXIT'

7. Display softkeys and wait for input of a function key:

Note: KEY 1-8 corresponds to Functionkeys F7-F14
KEY 9 corresponds to Cursorkey UP
KEY 10 corresponds to Cursorkey DOWN
KEY 11 corresponds to Cursorkey LEFT
KEY 12 corresponds to Cursorkey RIGHT

CALL VT_SOFTKEY (KEYNAM,KEY)
GOTO (51,50,50,50,50,50,53,99,52,52,52,52) KEY
CONTINUE
CALL MEN_READ (FID)
GOTO 50
CONTINUE
CALL MEN_NEXT (KEY, FID)
GOTO 50

On this section, values may be assigned and user actions may be performed:

CONTINUE
for example:
CALL MEN_GETI (1,K,STAT)

8. Program exit: clear screen

CONTINUE
CALL VT_RESETSCREEN (.TRUE.)
STOP
END
M E N - Menu and Dialog Software

H.2 Example programs for MEN subroutine package

Procedure: MEN_EXAMPLE2 Version: 1.0 Date: 29-Mar-93

Author(s): H. Kroiss Language: FORTRAN Systems: VMS, UNIX

Function: Example program to demonstrate the generation of a simple menu at runtime by usage of the "MEN" FORTRAN software interface routines.

This program works like MEN_EXAMPLE1 except that the background will be generated with "MEN" routines and not directly by PRINT statements. Furthermore the generated menu will be saved on file "men_example2.men" at program exit.

Used routines: MEN_OPEN,MEN_DEFINE,MEN_SET,MEN_WRITE,MEN_READ,
MEN_SETID,MEN_NEXT,MEN_SAVE,MEN_SETBGRD,VT_SOFTKEY,
MEN_WRITEBGRD,VT_CLEARSCREEN,VT_RESETSCREEN

Compiling: $ f77 -c -u men_example2.f

Taskbuild: $ f77 men_example2.o -lmen77

PROGRAM MEN_EXAMPLE2
INTEGER IVAL,FID,KEY,J
REAL RVAL
CHARACTER*80 CVAL,BLINE(24)
CHARACTER*8 KEYNAN(8)
LOGICAL ERR

1. Initialize menu database:

CALL MEN_OPEN (' ',ERR)

2. Define screen fields and field pointer:

CALL MEN_DEFINE (1,'Field-1',4,18,10,'F','B','N')
CALL MEN_DEFINE (2,'Field-2',4,45,10,'I','B','I')
CALL MEN_DEFINE (3,'Field-3',5,18,20,'C','B','U')
CALL MEN_DEFINE (4,'Field-4',6,16,20,'C','O','B')
CALL MEN_SETRANGER (1,0.0,250.5)
CALL MEN_SETRANGEI (2,-10,100)
CALL MEN_SETCOUNT (1,'Real value')
CALL MEN_SETCOUNT (2,'Integer value')
CALL MEN_SETCOUNT (3,'Character value')
FID = 1
CALL MEN_SETID (FID)

3. Define screen background:

BLINE(1) = ' Demonstration program for screen application'
BLINE(2) = ' ' 
BLINE(3) = ' ' 
BLINE(4) = ' ' Real value: Integer value:'
BLINE(5) = 'Character value:'
BLINE(6) = 'Only output:'
DO 10 J=1,6
   CALL MEN_SETBGRD (J,BLINE(J))
   CONTINUE

4. Clear screen and display screen background:
   CALL VT_CLEARSCREEN
   CALL MEN_WRITEBGRD

5. Set default values, if desired:
   CALL MEN_SET (1,IVAL,1.234565,CVAL)
   CALL MEN_SET (2,123456789,RVAL,CVAL)
   CALL MEN_SET (3,IVAL,RVAL,'Toggle Switch')
   CALL MEN_SET (4,IVAL,RVAL,'State OK')

6. Display default screen field contents:
   CALL MEN_WRITE (0)

7. Define names for softkeys:
   KEYNAME(1) = 'CHANGE'
   KEYNAME(2) = '
   KEYNAME(3) = '
   KEYNAME(4) = '
   KEYNAME(5) = '
   KEYNAME(6) = '
   KEYNAME(7) = 'EXECUTE'
   KEYNAME(8) = 'EXIT'

8. Display softkeys and wait for input of a function key:
   CALL VT_SOFTKEY (KEYNAME,KEY)
   GOTO (51,50,50,50,50,50,50,50,53,99,52,52,52) KEY
   CONTINUE
   CALL MEN_READ (FID)
   GOTO 50
   CONTINUE
   CALL MEN_NEXT (KEY, FID)
   GOTO 50

On this section, values may be assigned and user actions
may be performed:
   CONTINUE

for example:
   CALL MEN_GETI (1,K,STAT)
   ...

9. Program exit: save menu and clear screen
   CALL MEN_SAVE ('men_example.men')
   CALL VT_RESETSCREEN (.TRUE.)
   STOP
   END
**M E N - Menu and Dialog Software**

**H.3 Example programs for M E N subroutine package**

---

**Procedure:** M E N _E X A M P L E 3  
Version: 1.0  
Date: 29-Mar-93

**Author(s):** H. Kroiss  
Language: FORTRAN  
Systems: VMS, UNIX

**Function:** Example program to demonstrate the usage of an existing menunil created by an interactive menu session or by the "M E N _G E N E R A T O R" program.

This program works like M E N _E X A M P L E 2, but it will be assumed that a menunil already exits and the menu database must not be created at runtime.

**Used routines:** M E N _O P E N , M E N _R E A D , M E N _N E X T , M E N _G E T I D ,  
V T _R E S E T S C R E E N , V T _S O F T K E Y

**Compiling:** $ f 7 7 - c - u m e n _e x a m p l e 3 . f$

**Taskbuild:** $ f 7 7 m e n _e x a m p l e 3 . o - l m e n 7 7$

---

**Program:**

```
PROGRAM M E N _E X A M P L E 3
INTEGER KEY,FID
CHARACTER*8 KEYNAM(8)
LOGICAL ERR

1. Read saved menunil file and get last field pointer:
   CALL M E N _O P E N ('men_example.men',ERR)
   IF (ERR) G O T O 99
   CALL M E N _G E T I D (FID)

2. Display Information on terminal screen:
   CALL M E N _R E F R E S H

3. Define names for softkeys:
   KEYNAM(1) = ' CHANGE '
   KEYNAM(2) = ' ' 
   KEYNAM(3) = ' ' 
   KEYNAM(4) = ' ' 
   KEYNAM(5) = ' ' 
   KEYNAM(6) = ' ' 
   KEYNAM(7) = ' EXECUTE' 
   KEYNAM(8) = ' EXIT '

3. Define screen background:
   CALL M E N _S E T I D (FID)
```

---
4. Display softkeys and wait for input of a function key:

```
50 CALL VT_SOFTKEY (KEYNAME, KEY)
GOTO (51, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50)
51 CONTINUE
CALL MEN_READ (FID)
GOTO 50
52 CONTINUE
CALL MEN_NEXT (KEY, FID)
GOTO 50
```

On this section, values may be assigned and user actions may be performed:

```
53 CONTINUE
```

5. Program exit: clear screen

```
99 CONTINUE
CALL VT_RESETSCREEN (.TRUE.)
STOP
END
```
M E N - Menu and Dialog Software

H.4 Example programs for MEN subroutine package

--------------------------------------------------------------------------

Procedure: MEN_EXAMPLE4  Version: 1.0  Date: 29-Mar-93

Author(s): H. Kroiss  Language : FORTRAN  Systems: VMS,UNIX

Function: Example program to demonstrate simple noninteractive dialog by usage of menufiles.

Used routines: MEN_OPEN, MEN_GETI, MEN_GETR, MEN_GE TC

   MEN_SETI, MEN_SETR, MEN_SETC, MEN_SAVE

--------------------------------------------------------------------------

Compiling: $ f77 -c -u men_example4.f

Taskbuild: $ f77 men_example4.o -lmen77

PROGRAM MEN_EXAMPLE4
INTEGER IVAL,STAT
REAL RVAL
CHARACTER*80 CVAL
LOGICAL ERR

1. Read saved menu file:

   CALL MEN_OPEN ('men_example.men',ERR)
   IF (ERR) GOTO 99

2. Get values from menufile:

   STAT |   Error | Meaning
       |         |          
0    | None    | success
1    | None    | value was modified
2    | Warning | field format does not correspond
3    | Warning | data type of field has been changed
4    | Fatal   | conversion of field format not possible
5    | Fatal   | only input allowed on field
6    | Fatal   | specified field is not defined
7    | Fatal   | illegal field number (Range 1..100)

   CALL MEN_GETR (1, RVAL, STAT)
   CALL MEN_GETI (2, IVAL, STAT)
   CALL MEN_GETC (3, CVAL, STAT)
   PRINT *, 'Value-1: ',RVAL,' Value-2: ',IVAL
   PRINT *, CVAL

3. If you want to modify and save values:

   ... Perform any action ...

   RVAL = 25.3E3
   IVAL = 1
   CVAL = 'This is an example'
   CALL MEN_SETR (1,RVAL, STAT)
   CALL MEN_SETI (2,IVAL, STAT)
   CALL MEN_SETC (3,CVAL, STAT)
   CALL MEN_SAVE ('men_example.men')

99 CONTINUE
STOP
END
M E N - Menu and Dialog Software

H.5 Example programs for MEN subroutine package

Procedure: MEN_EXAMPLE5  Version: 1.0  Date: 29-Mar-93

Author(s): H. Kroiss  Language : FORTRAN  Systems: VMS,UNIX

Function: Simple calculator program written with the "MEN"
FORTRAN software interface routines.

Used routines: MEN_OPEN,MEN_DEFINE,MEN_SET,MEN_WRITE,MEN_READ,
MEN_SETID, MEN_NEXT,VT_SOFTKEY,VT_CLEARSCREEN,
VT_RETSSCREEN

Compiling: $ f77 -c -u men_example5.f

Taskbuild: $ f77 men_example5.o -lmen77

---

PROGRAM MEN_EXAMPLE5
INTEGER IVAL,FID,KEY
REAL RVAL,A,B,C
CHARACTER*80 CVAL,
CHARACTER*80 KEYNAM(8)
LOGICAL ERR

1. Initialize menu database:
   CALL MEN_OPEN (' ',ERR)

2. Define screen fields and field pointer:
   CALL MEN_DEFINE (1,'Operand-1',9,5,12,'R','B','N')
   CALL MEN_DEFINE (2,'Operator', 9,26,1,'C','O','N')
   CALL MENDEFINE (3,'Operand-2',9,35,12,'R','B','N')
   CALL MEN_DEFINE (4,'Result', 9,60,12,'R','O','I')
   CALL MEN_DEFINE (5,'Error' 23,1,80,'C','O','I')
   FID = 1
   CALL MEN_SETID (FID)

3. Display screen background:
   CALL VT_CLEARSCREEN
   PRINT *, 'Simple Calculator Program'
   PRINT *,
   PRINT *,
   PRINT *,
   PRINT *,
   PRINT *,
   PRINT *, ' 1.Operand Operator 2Operand Result'
   PRINT *, '  ---------------------  ---------------------  ---------------------'
   PRINT *, ' // '
   PRINT *, ' // '
   PRINT *, ' // '
   PRINT *, ' 1.Operand '
   PRINT *, ' // '
4. Set default values:

CALL MEN_SET (1,IVAL,0.0,CVAL)
CALL MEN_SET (2,IVAL,RVAL,'+')
CALL MEN_SET (3,IVAL,0.0,CVAL)
CALL MEN_SET (4,IVAL,0.0,CVAL)

5. Display default screen field contents:

CALL MEN_WRITE (0)

6. Define names for softkeys:

KEYNAM(1) = ' ENTER '
KEYNAM(2) = ' + '
KEYNAM(3) = ' - '
KEYNAM(4) = ' * '
KEYNAM(5) = ' / '
KEYNAM(6) = ' = '
KEYNAM(7) = ' EXIT '

7. Display softkeys and wait for input of a function key:

CALL VT_SOFTKEY (KEYNAM,KEY)
CALL MEN_SET (5,IVAL,RVAL,' ')  
CALL MEN_WRITE (5)
GOTO (51,52,53,54,55,56,57,58,59,60)

51 CONTINUE
CALL MEN_READ (FID)
GOTO 50

52 CONTINUE
CALL MEN_SET (2,IVAL,RVAL,'+ ')
CALL MEN_WRITE (2)
GOTO 50

53 CONTINUE
CALL MEN_SET (2,IVAL,RVAL,'- ')
CALL MEN_WRITE (2)
GOTO 50

54 CONTINUE
CALL MEN_SET (2,IVAL,RVAL,'* ')
CALL MEN_WRITE (2)
GOTO 50

55 CONTINUE
CALL MEN_SET (2,IVAL,RVAL,'/ ')
CALL MEN_WRITE (2)
GOTO 50
CONTINUE
CALL MEN_GET (1, IVAL, A, CVAL)
CALL MEN_GET (3, IVAL, B, CVAL)
CALL MEN_GET (2, IVAL, RVAL, CVAL)
IF (CVAL(1:1) .EQ. '+' ) C = A + B
IF (CVAL(1:1) .EQ. '-' ) C = A - B
IF (CVAL(1:1) .EQ. '*' ) C = A * B
IF (CVAL(1:1) .EQ. '/' ) THEN
  IF (B .NE. 0.0) THEN
    C = A / B
  ELSE
    C = 0.0
  ENDIF
ENDIF
CALL MEN_SET (5, IVAL, RVAL, 'Error: Division per zero')
CALL MEN_WRITE (5)
ENDIF
ENDIF
CALL MEN_SET (4, IVAL, C, CVAL)
CALL MEN_WRITE (4)
GOTO 50
CONTINUE
CALL MEN_NEXT (KEY, FID)
GOTO 50
CONTINUE
CALL VT_RESETSCREEN (.TRUE.)
STOP
END
1. Initialize menu database:
   ____________________________
   CALL MEN_OPEN (' ', ERR)

2. Define screen fields:
   ____________________________
   CALL MEN DEFINE (1, 'Field-1', 4, 18, 10, 'C', 'B', 'N')
   CALL MEN DEFINE (2, 'Field-2', 4, 49, 2, 'I', 'O', 'I')

3. Display screen background:
   ____________________________
   CALL VT_CLEARSCREEN
   PRINT *, ' Demonstration program for a Toggle Field'
   PRINT *, '-------------------------------------------'
   PRINT *, ' Toggle command: '
   PRINT *, ' Toggle Index: '

4. Define toggle list (for example: 12 commands)
   ____________________________
   TOGLST(1) = 'Reset'
   TOGLST(2) = 'Finish'
   TOGLST(3) = 'Delete'
   TOGLST(4) = 'Copy'
   TOGLST(5) = 'Print'
   TOGLST(6) = 'Submit'
   TOGLST(7) = 'Initialize'
   TOGLST(8) = 'Help'
   TOGLST(9) = 'Remove'
   TOGLST(10) = 'Substitute'
   TOGLST(11) = 'Purge'
   TOGLST(12) = 'Edit'
M E N - Menu and Dialog Software

H.6 Example programs for MEN subroutine package

5. Set starting index of toggle list:

CALL MEN_SETTOGGLE (1, TOGLST, 12, 7)
CALL MEN_SET (2, 7, RVAL, CVAL)

6. Display default screen field contents:

CALL MEN_WRITE (0)

7. Define names for softkeys:

KEYNAM(1) = 'UP'
KEYNAM(2) = 'DOWN'
KEYNAM(3) = '
KEYNAM(4) = '
KEYNAM(5) = '
KEYNAM(6) = '
KEYNAM(7) = '
KEYNAM(8) = 'EXIT'

8. Display softkeys and wait for input of a function key:

CALL VT_SOFTKEY (KEYNAM, KEY)
GOTO (5I, 51, 50, 50, 50, 50, 50, 99, 50, 50, 50, 50) KEY

CONTINUE
IF (KEY .EQ. 1) THEN
  CALL MEN_TOGGLE (1, 'U', TOGLST, 12, TOGIND)
ELSE
  CALL MEN_TOGGLE (1, 'D', TOGLST, 12, TOGIND)
ENDIF
CALL MEN_SET (2, TOGIND, RVAL, CVAL)
CALL MEN_WRITE (2)
GOTO 50

9. Program exit: clear screen

CONTINUE
CALL VT_RESETSCREEN (.TRUE.)
STOP
END
PROGRAM MEN_EXAMPLE7
INTEGER IVAL,I1,I2,J
REAL RVAL,R1,R2
CHARACTER*80 CVAL,C1,C2
LOGICAL ERR

CALL MEN_OPEN (' ', ERR)
CALL MEN_DEFINE (1, 'Field-1', 1,34,10, 'I', 'I', 'N')
CALL MEN_DEFINE (2, 'Field-2', 1,34,10, 'I', 'I', 'N')
CALL MEN_DEFINE (3, 'Field-3', 3,34,15, 'F', 'I', 'N')
CALL MEN_DEFINE (4, 'Field-4', 4,34,15, 'F', 'I', 'N')
CALL MEN_DEFINE (5, 'Field-5', 5,34,20, 'C', 'I', 'N')
CALL MEN_DEFINE (6, 'Field-6', 6,34,20, 'C', 'I', 'N')

CALL MEN_SETBGDR (1, 'Please enter INTEGER value 1 :')
CALL MEN_SETBGDR (2, 'Please enter INTEGER value 2 :')
CALL MEN_SETBGDR (3, 'Please enter REAL value 1 :')
CALL MEN_SETBGDR (4, 'Please enter REAL value 2 :')
CALL MEN_SETBGDR (5, 'Please enter CHARACTER value 1:')
CALL MEN_SETBGDR (6, 'Please enter CHARACTER value 2:')
CALL MEN_REFRESH

Read all input values from terminal screen:
DO 10 J=1,6
  CALL MEN_READ (J)
  CONTINUE

Assign input values to local variables:
CALL MEN_GET (1,I1,RVAL,CVAL)
CALL MEN_GET (2,I2,RVAL,CVAL)
CALL MEN_GET (3,IVAL,R1,CVAL)
CALL MEN_GET (4,IVAL,R2,CVAL)
CALL MEN_GET (5,IVAL,RVAL,C1)
CALL MEN_GET (6,IVAL,RVAL,C2)

STOP
END
M E N - Menu and Dialog Software
H.8 Examples programs for MEN subroutine package

# /bin/csh
#--------------------------------------------------------------
# Procedure: MEN_EXAMPLE8  Version: 1.0  Date: 29-Mar-93
#--------------------------------------------------------------
# Language : csh  Systems: UNIX
#--------------------------------------------------------------
# Author(s): H. Kroiss  File: /usr/examples/men_example8
#--------------------------------------------------------------
# Function: Example program to demonstrate the usage of an
# existing menufile by a shell script.
#--------------------------------------------------------------
# It will be assumed that the field 1 in the menu file
# is declared as an INTEGER field an have the name
# 'FIELD-1'. Field 2 should be defined as a CHARACTER*n
# menu field. The name of the menu descriptor file is
# 'men_example.men'.
#--------------------------------------------------------------
# This shell script should read the current value of
# field 1, increment the value by 1 and write back
# the modified value to the menu file on field 1.
# Furthermore a new character string should be assigned
# to field 2.
#--------------------------------------------------------------
# Used programs: men_get_i, men_get_c, men_set_i, men_set_c
#--------------------------------------------------------------

1. Read INTEGER value from menu file on field number 1:
--------------------------------------------------------------
@ IVAL = `men_get_i men_example.men 1`
@ IVAL = `men_get_i men_example.men FIELD-1`
echo "Current integer value of field 1: "$IVAL"
@ $IVAL = ($IVAL + 1)

2. Write back modified INTEGER value to menu file:
--------------------------------------------------------------
men_set_i men_example.men 1 $IVAL
or:
men_set_i men_example.men FIELD-1 $IVAL

3. Read CHARACTER value from menu file on field number 2:
--------------------------------------------------------------
set CVAL = `men_get_c men_example.men 2`
or:
set CVAL = `men_get_c men_example.men FIELD-2`
echo "Current character value of field 2: "$CVAL"

4. Set a new character string on field number 2:
--------------------------------------------------------------
set CVAL = "This is a new character string"
men_set_c men_example.men 2 $CVAL
or:
men_set_c men_example.men FIELD-2 $CVAL

exit $status