Process Visualisation
JETTER VIADUKT

with

Message Editor

JETTER GmbH Steuerungstechnik
Gräterstraße 2
71642 Ludwigsburg
Germany
Tel. 07141/2550-0
Fax 07141/2550-25
Mailbox 07141/59834
Hotline 07141/2550-44

Documentation VIADUKT 2
Second edition
February 1994
Software release 1.40

Table of Contents

1. Introduction

The monitoring system **VIADUKT** is an intelligent Man-Machine-Interface for the PASE Process-PLC.

The VIADUKT provides following functions:

- operator guidance with clear text
- menu driven user interface
- error diagnosis
- comfortable input and display of process values
- graphical visualisation of process data
- errors and warnings in clear text or graphically

In many applications in the automation industry, the visualisation of process data is very helpful. It helps keeping an overview of complex processes. With the JETTER **VIADUKT**, it is possible to see internal data of the controller in realtime on the screen and to manipulate the data. Numerical values can be displayed in physical units, for instance the current speed of an axis in meters per second or the temperature in centigrate, etc.
The **JETTER VIADUKT** performs the necessary linear transformation from PLC internal representation into physical dimension and vice versa.

The value of binary signals, for instance inputs, outputs and flags can be displayed with the colour of drawn objects. If a certain valve is on or off can be shown by its colour on the screen. At the same time, the colour

of the corresponding pipe can also change its colour.

VIADUKT runs on any IBM AT class computer or true compatible. It requires a graphics adapter of EGA or VGA type. A mouse is necessary to edit frames.

JETTER offers VIADUKT in two versions: software only or a combination of software and a 19 inch rack mountable AT computer with monochrome or colour display.

The VIADUKT software is available as demo version. The demo version can be used to test the program and to create frames.

VIADUKT lite is a VIADUKT derivate with following restrictions:

o	no message editor	(switch /W /L /H)
o	no data transfer	(switch /D)
o	no DOS functions	(switch /E)
o	no transfer of date and time	(switch /C)
o	no TSR support	(switch /U /V)

1.1 Hardware requirements

The VIADUKT software requires following hardware:

- IBM AT, PS/2 or true compatible (for instance 286, 386SX, 386, 486).
- EGA or VGA graphics adapter.
- Microsoft compatible mouse.
- serial port COM1 for connection of PASE.
- parallel port for the copy protection device ("hardware lock")

The industrial PC offered by JETTER has following specifications:

- IBM compatible PC with 80386DX processor (40 MHz)
- 4 MB RAM.
- Harddisk 170 Megabytes.
- 3 1/2 inch disk drive 1.44 MB.
- VGA graphic adapter with LCD screen:
  - monochrome LCD
  - colour TFT-LCD
- 2 serial and one parallel port.
- keyboard with 14 function keys, numeric field and cursor keys. Connector for AT standard keyboard.

The cable necessary for connection of the PASE and a mouse are additionally available.

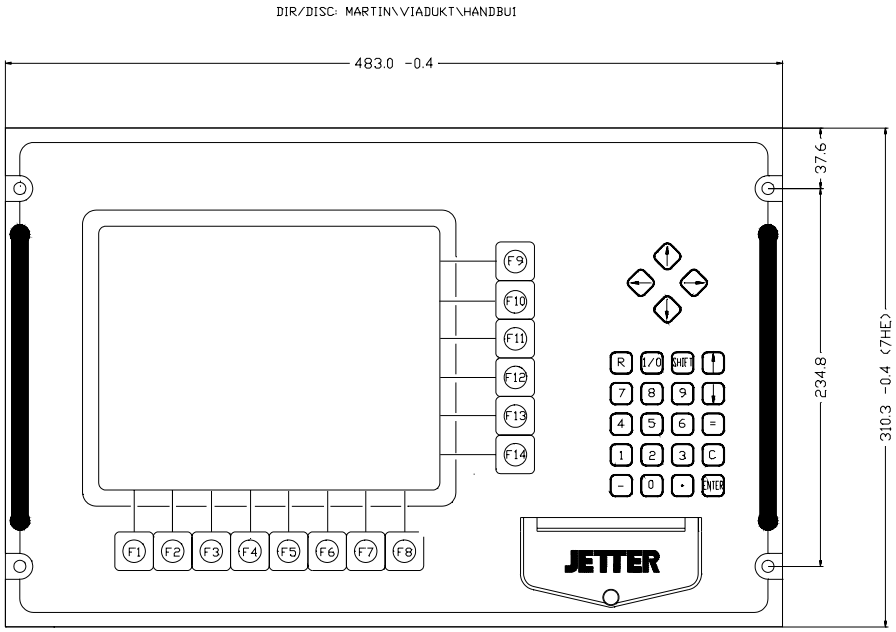


Figure 1 Front view

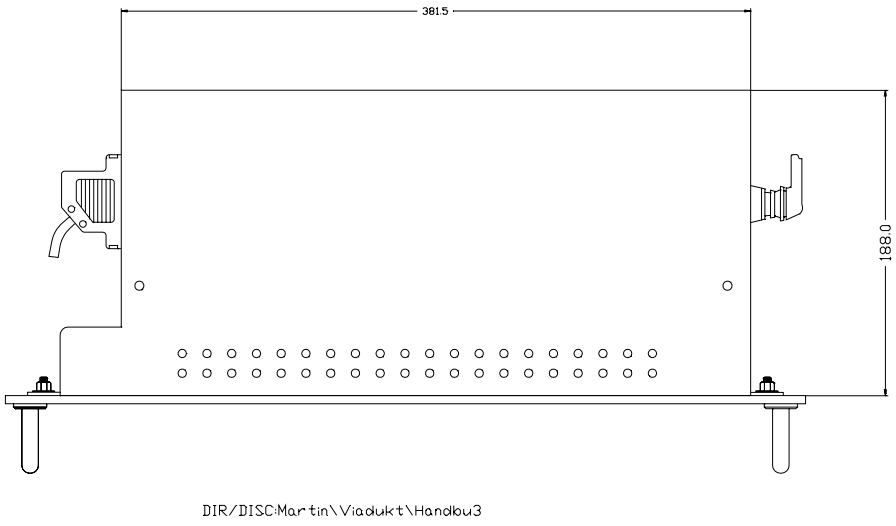


Figure 2 Plan view

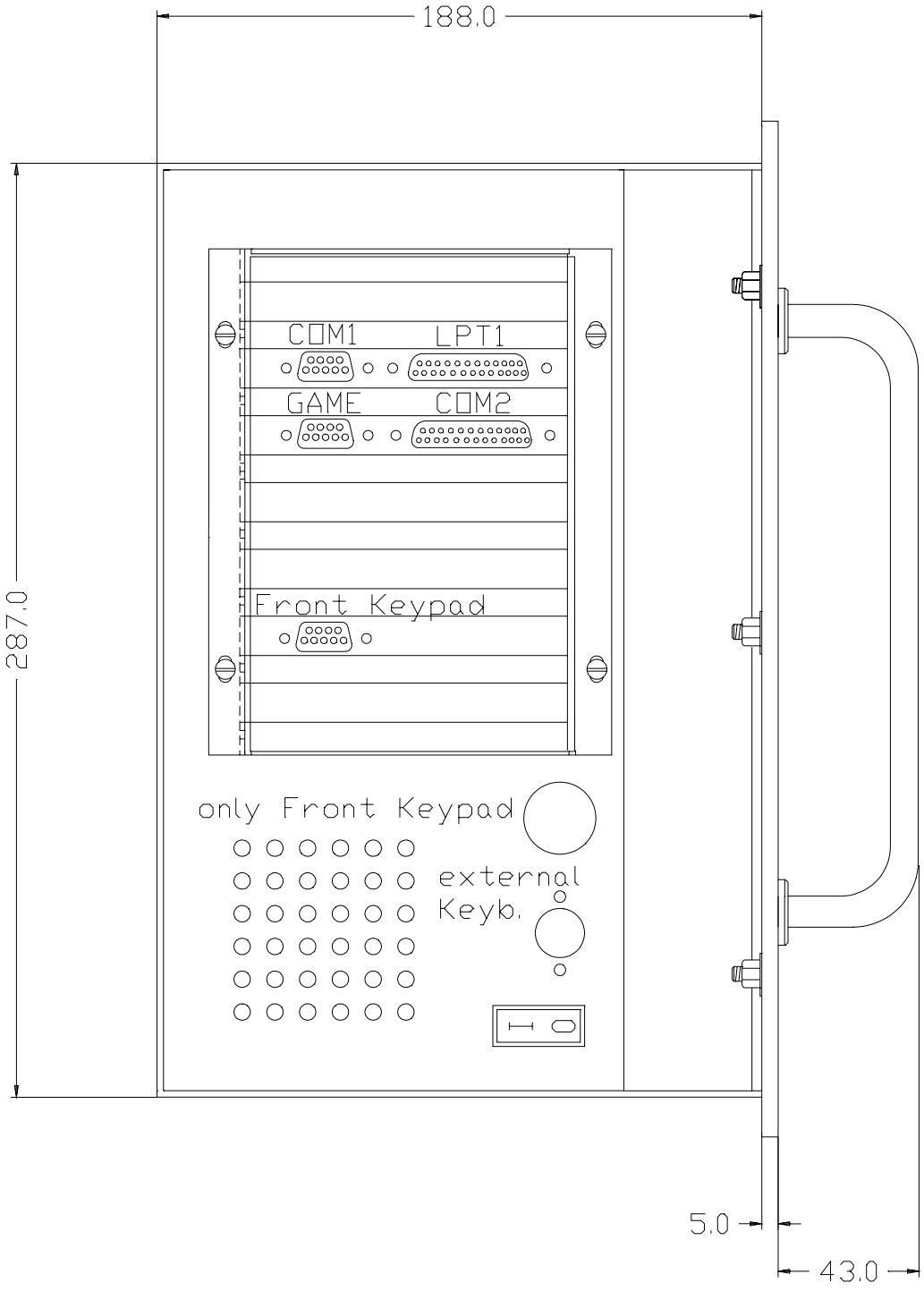


Figure 3 Side view (terminal assignment)

1.2 Connection of the PASE controller

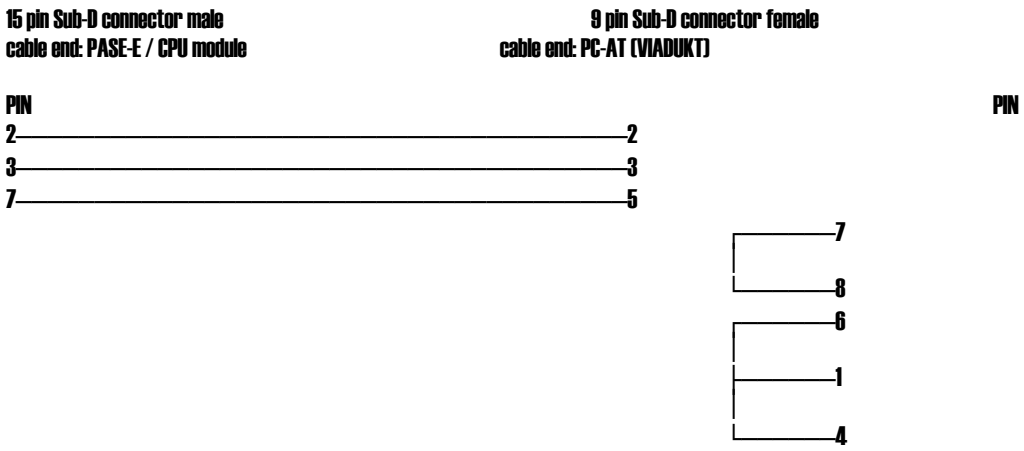
The process controller PASE must be connected to the serial port COM1. On the PC side, this is normally a male 9 pin Sub-D connector.

At the other end of the cable, the connection depends on the type of PASE to be connected.

For a PASE-E, there are two possibilities to connect a VIADUKT.

1.2.1 Connection to the PASE-E CPU, 15 pin Sub-D female

Because normally a display module is not used at the same PASE-E where a VIADUKT is connected, the 15 pin Sub-D connector is available for connection of the VIADUKT. To make this connection, the following cable is necessary:

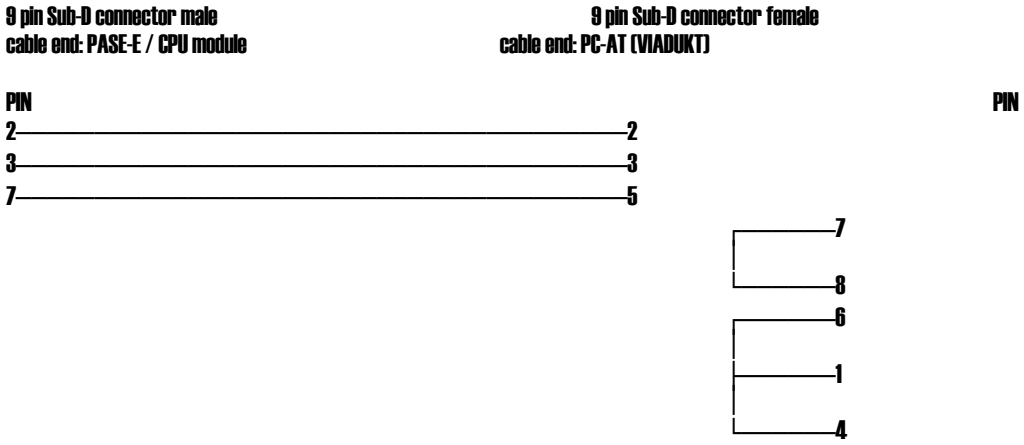


To the 9 pin Sub-D connector the PC for programming the PASE can be connected at the same time. In this configuration, two PC are connected to the PASE-E-CPU at the same time.

Remark: A "jumper" has to be plugged on the CPU module. The jumper is located directly "behind" the 9 pin connector.

1.2.2 Connection to the 9 pin Sub-D of the PASE-E respectively the PASE-Mikro

In some situations it might be desirable to have **one** PC and run the SYMPAS and VIADUKT program alternatively without changing connections to the PASE every time. This is possible with the standard programming cable for PASE-E (type "E-PK"):



In this case, a display module can be connected to the 15 pin Sub-D of the PASE-E at the same time.

1.2.3 Connection to the PASE-J CPU

Usage of the VIADUKT with PASE-J is possible starting with version 2.0 of the PASE-J operating system software. Connection to the PASE-J is established with the standard PASE-J programming cable (type "J-PK").

1.3 Connection of a mouse

To create frames with the VIADUKT program, a mouse is necessary. In "automatic mode", after creation of all frames, no mouse is necessary for operation.

Because VIADUKT uses COM1 for communication with the PASE, it is convenient to use COM2 for the mouse connection.

To put the mouse into operation, a program called "mouse driver" must be activated before the VIADUKT program is started. The mouse driver is normally delivered with the mouse and usually is named MOUSE.COM. It makes sense to write the call-up of this program into your AUTOEXEC.BAT file. In that case, the mouse is activated whenever the PC is started. For the corresponding call-up syntax refer to your mouse manual.

1.4 Software installation

1.4.1 JETTER VIADUKT PC rack

The software is already installed on the JETTER VIADUKT PC rack. The files are located in the directory C:\VIADUKT. The AUTOEXEC.BAT file contains among others the following lines:

```
...
CD C:\VIADUKT
VIAUK /A
```

1.4.2 Software VIADUKT

The software package **JETTER VIADUKT** consists of the following items:

1 disk (3.5 inch) with the program files
1 software copy protection device ("hardware lock")
1 user´s manual

For installation of the VIADUKT onto the harddisk simply use the install routine. The copy protection device must be connected to one of the parallel ports on your PC. You can use LPT1 to LPT3 to run the software. Installation is started by:

```
A:Install <ENTER>
```

ATTENTION: The parallel port is a **female** connector with 25 pins on the rear of your PC, normally labelled with "LPT". Connect to hardware lock **only** to this connector. If you are not sure which connector is the parallel port, then ask someone who seems to know it. Connecting the

hardware lock to a wrong connector may cause permanent damage to the device. On the hardware lock the direction of connection is labelled with arrows. You have to put the male side of the hardware lock into the female connector LPT of the PC.

After the files are copied to the harddisk, the program can be started with VIAUK or VLITEUK. The program detects the graphics adapter present in your PC (EGA or VGA) and uses the best possible resolution. Now appears the editor window on the screen where frames can be created.

At the center of the screen a crosshair cursor appears that can be moved with the mouse. You should check now if your mouse works. If it does not it might have one of the following reasons:

- mouse is connected to the wrong serial port.
- mouse driver is not installed.
- some errors in the configuration files of your PC (for instance CONFIG.SYS). Refer to your mouse manuals.

If the mouse works, the crosshair cursor can be moved over the drawing area. Outside of the drawing area, the crosshair disappears and an arrow sign makes is possible to choose one of the menu functions.

2. Creation of frames with the VIADUKT

2.1 The user interface

The term "user interface" in this case means how the VIADUKT program and the programmer of frames work together (the frame editor). The user interface consists of the following screen areas:

- o top side (pull down menus)
- o right side (command selection)
- o bottom side (command input and display)
- o left side (colour, font, and line selection)
- o center (drawing area)

Some fundamental editor functions may be mentioned before.

Snap the function moves the crosshair cursor in a certain grid which step width can be determined with the "Snap" function of the "Settings" menu.

Orthothe function only allows orthogonal, function relating movements of the cross wires. I.e. only lines can be drawn which form right angles to each other.

A window can be opened in the drawing editor to select object groups alternatively to the cursor (for MOVE, CHANGE, etc. operations). Two kinds of windows are distinguished:

<F> only objects are selected which extents are within the window.

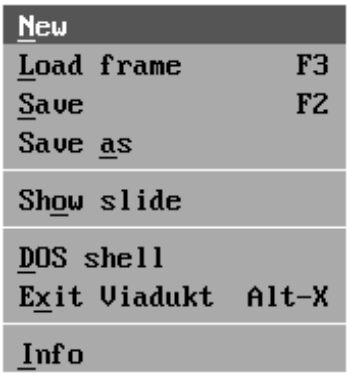
<K> all objects are selected which are partly or completely enclosed by the window.

Using the cursor in combination with the SHIFT key cancels selection of an object.

The **ALT-F5** key switches to the DOS screen, any key returns to the VIADUKT.

F7 display of the cursor coordinates On/Off.

2.1.1 "File" menu



New The screen is cleared, currently loaded data is erased and a new frame can be drawn or loaded. **VIADUKT** asks, if the current frame is to be saved before it is erased from the screen.

Load frame File is loaded and can be changed with the editor commands. If you use this command repeatedly, a frame can be composed of several small elements (library of frame elements).

Save An already loaded frame definition is saved in its current state using the current filename (the old version of that file is overwritten). Frames that are to be called by the PASE, must have **numeric** filenames ("1" to "8000000"). Other frames (for instance library elements or logos) that are combined to frames later on, can have any DOS compatible name (up to eight characters), for example "LOGO" or "VALVE".

Save as The file name can be defined before the file is saved.

Show slide File (frame) is loaded, but the data is not changeable. This function can be used to test if several frames fit together (overlapping areas and so on) or it can be displayed as a "background" behind a frame in order to have "construction lines". Any number of slides can be shown with this function. Only the data that were drawn or loaded with "Load frame" are saved, when the SAVE function is called. Data loaded with "Show slide" is ignored in that case. The slide data can also be wiped from the screen with the REDRAW command.

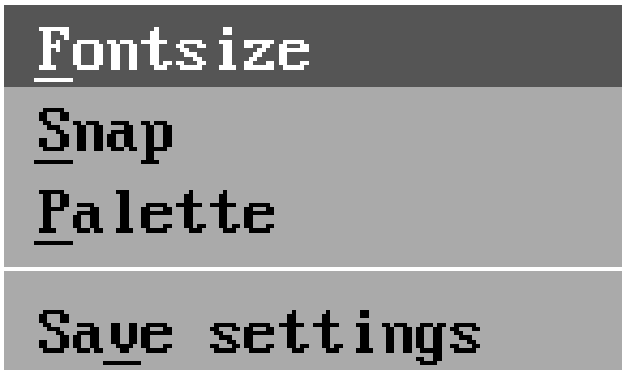
DOS shell Calls the DOS command interpreter COMMAND.COM. The **VIADUKT** program is still active and the screen content is not lost. You can now enter DOS commands such as DIR or COPY. Return to VIADUKT with the EXIT command.

Exit VIADUKT The program is terminated, the screen content has to be saved before.

Info The window makes following information available:

- o version number
- o current frame file
- o call-up parameters (DOS command line parameters / switches of VIAUK.EXE)

2.1.2 "Settings" menu



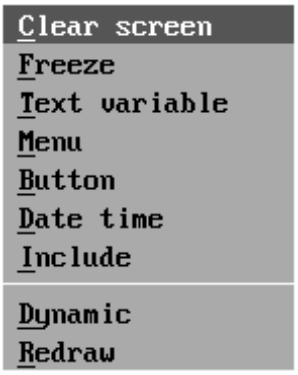
Fontsize Selection of the size for texts and numbers. When "Normal" is the current font type, then only integer numbers are allowed. Other fonts also accept fractional parts.

Snap Set the width of the snap grid. If the SNAP function is active, then the crosshair cursor can only be moved on a grid, whereby the width is defined with this command. This function for instance eases the construction of equally sized boxes.

Palette The user can create an application specific colour palette by this function and save it using the name of the VIADUKT standard palette VIADUKT.COL.

Save settings The current settings are saved in a file with the name VIADUKT.CFG. If VIADUKT starts and the file VIADUKT.CFG exists, the file is read and the settings are configurated accordingly. The following data is saved in VIADUKT.CFG :
Current colour, line type, line width, shape, font type, font width, font size, snap on/off, snap grid setting, ortho on/off, file name of the current frame, colour of the cross wires respectively the input frame.

2.1.3 "Draw" menu



Clear screen Clear edit area. The new empty window has the actual colour as background colour. The previous content of the screen is lost. Save the screen content before using the clear screen command, if necessary.
CLEAR SCREEN is an active drawing command that is kept in the definition of the frame, for instance when a frame containing a clear screen command is called in automatic mode, the screen is erased.

Freeze FREEZE is a drawing command. FREEZE stops the cyclical refresh of dynamical elements, without erasing them from the screen. To thaw the freezed elements the corresponding frame has to be called up again (nominal frame register).

Textvariable Texts can be input into the frame and thus into registers of the PASE controller. For detailed information see section 2.2.2.3.

Menu Using this drawing command it is possible to create a label field for one of the 14 function keys that are present on the front keyboard of the JETTER VIADUKT PC. The label field is drawn at a position that corresponds to the function key location on the JETTER VIADUKT PC.

However, the field can be moved afterwards with the MOVE command.
The field contains a 2 line text of 8 characters each. Optionally it can be defined that pressing the function key initiates the call-up of a predefined frame number.

Button

This function creates any buttons (switches, pushbuttons), which can be operated with the mouse in automatic mode. After the button function was started with the selection line the size of the button is defined by the start and the end point. After definition of the end point a window opens, in which the colour of the inscription text can be selected. The colour of the button itself corresponds to the current drawing colour. Now a window appears for definition of the following parameters:

o **Flag number**: this flag is set by the button (pushed button corresponds to a set flag).
o **Offset**: the offset value is added to the flag number if the flag is indirectly addressed by a register. The sum is the actual flag number that is set by the button.
o **Type**: two different button types can be used. The button (pushbutton) remains active as long as the mouse button is pushed. The switch toggles at each mouse click.
o **Text**: the texts for the button inscription are defined here.
o **Frame number**: beside flag control a button can also call up a further frame. This parameter specifies which frame is called up.

Date time

This object integrates the current date and time into a frame. The display format can be specified by the user. First of all the start point is defined from which on the date respectively time text is displayed. After that a window appears in which any texts together with following reserved commands can be placed:

Command:	**is replaced by:**
ap	am for times from 0.00.00 to 11.59.59 pm for times from 0.00.00 to 23.59.59
da	current day from 1 to 31
Da	current day from 01 to 31
date	automatical generation of the date in country format
day	name of day in shortform (e.g. Mon)
dow	name of the day (e.g. Monday)
ha	12 hour display
ho	24 hour display
mi	current minute from 0 to 59
Mi	current minute from 00 to 59
mo	current month from 1 to 12
Mo	current month from 01 to 12
mon	name of the month, short (e.g. Jan)
Mon	name of the month (e.g. Januar)
se	current second from 0 to 59
Se	current second from 00 to 59
time	automatical generation of the time in country format
ye	number of the year, 2 places (e.g. 93)
year	number of the year, 4 places (e.g. 1993)

oa	adds the abbreviations for the ordinal numbers

Example:

the format definition

```
Protocol of Mon daoa, year at time
```

results at January 5th, 1994 at 15.38 and 36 seconds in following display

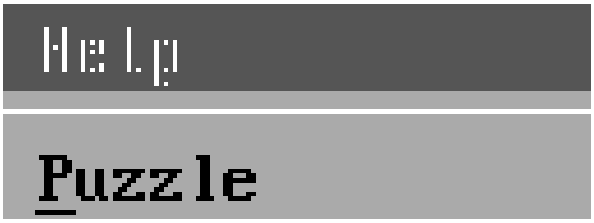
```
Protocol of January 5th, 1994 at 15:38:36
```

Include Any VIADUKT frames (extension .VIA) respectively PCX files (extension .PCX) can be integrated into a frame.

Dynamic Drawn elements can dynamically change their colour dependent on some information that is retrieved from the PASE (for instance state of an output). The second colour is asked.

Redraw This command refreshes the screen.

2.1.4 "Help" menu



Help Cancel with the upper left corner.

Puzzle Game

2.1.5 "Auto" menu

Preview The frame that is currently in the editor is zoomed to the whole screen size (like it is displayed in automatic mode). Return to the VIADUKT editor with Ctrl-End on the keyboard.

Automatic Start the automatic mode of **VIADUKT**. To use this mode, a PASE controller is necessary. This controller has to be connected to COM1 port as described in chapter 1.2. Values of numbers and bargraphs, etc. are displayed in realtime. The cursor can be moved on input variables and registers can be changed.

2.1.6 Right side: command selection, ortho, snap

At the right side of the screen, the commands necessary for creation of new elements are listed. The commands are selected by moving the mouse cursor onto the command name and pressing the left mouse button. An already selected command can be cancelled by pressing the right mouse button.

At the top right corner of the screen there are two buttons labelled Snap and Ortho. If you click these fields they flip from off to on and vice versa.

Whenever **ORTHO** is active, lines can only be drawn horizontally or vertically.

If the **SNAP** function is active, the crosshair cursor can only be moved on certain positions. How far these positions are located from each other can be defined using the menu "Settings, Snap".

Below the Ortho and Snap buttons, there is a list of drawing and edit commands.

Drawing commands that are directly selectable at the right side of the screen are: line, rectangle, circle, arc, ellipse, bar, text, value, fill and bargraph. There are two more drawing commands located inside the menu "Draw" that can be reached through the top line of the screen: clear screen and menu.

Below the list of drawing commands, there are the fields with the edit commands: move, copy, change, erase, last and end selection.

A thorough description of the drawing commands and the edit commands can be found in other parts of this manual.

2.1.7 Bottom side: command display, text input field

This is the dialogue window between the frame programmer and the VIADUKT. Normally the prompt is:

Command:

All drawing and edit commands that are selectable with the mouse can also be typed in using the keyboard. The only exception is "LAST".

If the command requests more parameters, they are displayed in this part of the screen.

2.1.8 Left side: colour, font, line style selection, fill pattern

Colours The current colour can be selected at the left margin. 16 different colours (gray scales) can be chosen.

Font The desired font can be determined in a window that appears.

LStyle The line style can be selected in a window.

Fill The pattern for the fill function can be defined.

Additionally the cursor position is displayed at the upper left margin (x and y position). During input of an angle the display shows the radius and the degree value.

The cross wires position values can be freezed with the F7 key.

2.1.9 Drawing area

In the center of the editor screen, the drawing area is located. Drawings are defined in a "world coordinate system", with a size of 1001 x 751 units. The valid range of coordinates is :

x-coordinate: 0 to 1000
y-coordinate: 0 to 750

The position (0,0) is the bottom left corner of the drawing area.

While the edit mode is active, the drawing area is zoomed to fit into the 4 menu areas. In automatic mode, the drawing area fills the whole screen.
The translation from world coordinates to real hardware pixels (640 x 350 EGA or 640 x 480 VGA) is performed by the VIADUKT program.
The result is, that is is not necessary to know the resolution of the display adapter that is used later on the display the frames at the point of time when the frames are defined. Therefore one can define frame libraries that are hardware independent. The elements of such libraries can be used on EGA and VGA systems without change.

A crosshair cursor is normally displayed on the drawing area. If the snap resolution is set for instance to 5, then the cursor can be moved on 201 x-positions and 151 y-positions (if SNAP is activated). The positions are 0, 5, 10, 15, and so on.

2.1.10 Input with the keyboard

As already described above, some of the commands and functions can also be activated with the keyboard. This is possible with all commands on the right side of the screen with the exception of SNAP, ORTHO and LAST.

The END-SELECTION function can be activated with the space bar.

The commands of the "Draw" menu can be input by the keyboard.

2.2 Definition of objects

On the following pages all available drawing objects are described:

An object that is created using one of the drawing commands is always defined using the current settings of colour, linetype and linewidth. If the object contains text information, the current settings of textfont, fontwidth and fontheight are used. Objects that contain filled areas use the current shape.
The current settings are displayed on the left side of the screen. If the next object has to be drawn with other settings, then these settings have to be changed **before** the activation of the drawing command.

2.2.1 Basic objects

2.2.1.1 Line

Activate with the mouse "Line" or use the keyboard: "LINE".
You have to point to the start and end points of the line. After the start point is set a rubber band line moves around with the crosshair cursor to show where the line would be drawn. If ORTHO is ON, only vertical or horizontal lines can be drawn. ORTHO can be switched on and off while the line command is active.
The x and y coordinates are displayed in the cursor position field until the start point is defined. Then the display changes to distance and angle for the defintion of the second point.

2.2.1.2 Rectangle

Activate with the mouse "Rect." or use the keyboard: "RECTANGLE"
The first point is the start point of the rectangle, the second point is the end point (diagonal corner). Therefore the length of both sides is defined with the second point.

2.2.1.3 Bar

Activate with the mouse "Bar" or use the keyboard: "BAR"
A bar is the same element as a rectangle except that it is filled with the current colour. The sequence of definition is the same as for the rectangle.

2.2.1.4 Circle

Activate with the mouse "Circle" or use the keyboard: "CIRCLE"
The first point to enter is the center of the circle. The size of the circle varies now by moving the cursor around. The position display shows the actual radius. Click the left mouse button to fix the radius.

2.2.1.5 Arc

Activate with the mouse "Arc" or use the keyboard: "ARC"
Center, first point and end point define an arc. If ORTHO is ON, the angle between center and end point can only be 90, 180, 270 or 360 degrees. The arc is always drawn in the mathematical positive direction (from 0 to 360 degrees anticlockwise).

2.2.1.6 Ellipse

Activate with the mouse "Ellipse" or use the keyboard: "ELLIPSE"
The first point defines the center of the ellipse. The second point defines the first radius and the third point defines the second radius of the ellipse.

2.2.1.7 Text

Activate with the mouse "Text" or use the keyboard: "TEXT"
Before using the text command, set the font type and size using the Settings / Font type menu. The first point is the start point of the text. Then the text string must be entered using the keyboard. The string input is confirmed with the ENTER key. The input can be cancelled with the ESC key.
If several lines of text have to be written, the space bar function repeats the last command (here: text). The right mouse button also recalls the last command.
The text string and the Font size / Type can be changed afterwards with the change command.

2.2.1.8 Fill

Activate with the mouse "Fill" or use the keyboard: "FILL"
This object is used to fill an area with a shape of a certain colour. After FILL is activated, a point inside the area that is to be filled must be clicked with the mouse. The next selection is to show the border colour with the mouse. The selected area is now filled with the current shape and colour.

ATTENTION: o the element(s) that surround the area to be filled have to form a **closed** border.
o the **boundary colour** must not be equal to the **fill colour** if **dynamic** is used.

2.2.2 Dynamic objects

Objects are said to be dynamic, if one or more of their attributes can be influenced from a PASE controller connected to the VIADUKT.

The objects VALUE, BARGRAPH and TEXT VARIABLE are dynamic by definition. These objects can show the content of registers of the PASE controller either as decimal value (VALUE) or graphically (BARGRAPH) or as string (TEXT VARIABLE). Any object can be made dynamic by applying the command DYNAMIC on it. These objects are then able to change their colour dependent on some bit variable of the PASE controller.

2.2.2.1 Value

This object displays the content of registers as a decimal numeric string. The numerical value is refreshed continuously. Optionally, a value object can be used to allow input of new values into the specified register by the operator.

To define a VALUE object, the start point for the numeric string has to be defined like the start point of a text string.

Now a window opens with the "Value parameters" label. Here the data for dynamic definition has to be entered:

Register number: Number of a register. The content of that register is displayed continuously in automatic mode. Example: The current position of axis 51 is to be displayed. In that case enter 15109 in the register number field.
The register number can also be defined **indirectly** in the same sense as in a PASE controller. To use this feature,

enter the letter **R** before entering the register number. An application of this feature is for instance a set editor (see example below).

Register offset:

If the register number is indirectly specified, then this parameter can be used to simplify the program effort in the PASE controller for such applications as set editors. The number of the register in automatic mode is determined with the following formula:

(Register number) + Register offset

Number of digits xxx. :

Number of places on the left side of the decimal point. The numerical string is displayed in a fix sized area on the screen. It is definable how many digits are to be displayed left and right of the decimal point. Only values between 1 and 7 make sense for that parameter. The space for a minus sign is reserved automatically.

Number of digits .xxx :

Number of places on the right side of the decimal point. Value range is 0 to 9.

Note: numbers in the PASE system are 24 bit integers. Therefore the range of values is -8 million to + 8 million, i.e. about 7 digits. The definition of 9 digits after the decimal point is possible but gives no additional information.

Display multiplier:

The multiplier can be used to convert a numeric value from the controller internal representation into a numeric system that the user is familiar with (e.g. physical units). Example: The actual position of an axis (register xxx09) is counted in increments. The display on the VIADUKT is to show the value in mm. For instance 50 increments

per mm. To solve this, a multiplier of 0.02 has to be specified in the multiplier parameter field.

The default value for this parameter is 1.0 so that numerical values are equal in PASE and VIADUKT.

Display offset:

For the same reason as in case of the multiplier parameter, an offset parameter can be specified. The value specified as Offset is added **after** multiplication of the register contents with the multiplier. An application for this parameter would be the conversion of the reading of an analog pressure sensor connected to a PID module. For instance the value 0 means 1000 mbar and the value 2047 means 5000 mbar. To display that register in mbar on the VIADUKT screen, the multiplier has to be 4000/2048 = 1.953 and the offset has to be 1000.

The default value for the offset is 0.0 so that values inside the PASE and VIADUKT are equal.

These are the parameters of a value object that is used to display the content of a register.

If the value object allows input from the operator, then enter a **Y** in the "input allowed" parameter field. Now two more parameters have to be specified.

Minimum value, maximum value:

If the value object is selected in automatic mode with the cursor, the operator is allowed to input new information (new numerical value). This input is terminated with the ENTER key. The value that was entered will now be checked for validity. The value must be greater than or equal to the minimum value **and** less than or equal to the maximum value. If this is not the case, an error message is displayed for 2 seconds and the cursor can not leave this input field until a value is entered that is valid.

If the value is valid, the display offset is subtracted and the result is divided by the display multiplier. The result is then transferred to the PASE controller into the register that is specified with the Register number and Register offset parameters.

2.2.2.2 Bargraph

With this object, the content of registers can be displayed in form of a bar with variable size. The size is refreshed continuously.

The graphical definition of the bargraph is a three step operation: the first point is the corner point on the base line of the bargraph. The base line defines the width of the bargraph and the orientation of the dynamic sizeing: the bargraph will grow perpendicular to the base line.

The third point is the end point of the bargraph (maximum bar). This point defines the maximum point and the direction of movement (below or above of the base line, or to the right or left if base line is vertical).

Now a window opens where the parameters of the bargraph object has to be entered.

Register number and Register offset have the same meaning as in the VALUE object definition (see above).

Maximum value: If the specified register contains that value, the bargraph is drawn from the base line up to the bar maximum.

Minimum value: If the specified register contains that value, the bargraph is drawn with a heigth of zero and consequently is invisible.

All values between minimum value and maximum value are converted to a corresponding heigth of the bargraph. If the actual values are outside of this range, they are cut at the range limits.

2.2.2.3 Text variable

Using this object register contents can be displayed as ASCII strings on the screen.
Texts can also be input into registers of the PASE controller with this function. Three characters can be stored in a 24 bit register since a ASCII character occupies 8 bit. The first register of a string has a special purpose:

- the length byte (bits 0 to 7) contains the number of characters that are displayed by the VIADUKT.
- the status byte (bits 8 to 15) defines the refresh mode that is used by the VIADUKT:
  - 0 no display refresh
  - 1 display refresh with confirmation (the number 0 is written into the status byte for confirmation)
  - 2 diplay refresh without confirmation

First the start point has to be set with the mouse for definition of a text variable. Now a window appears in which the text variable parameters have to be specified:

Register number and register offset have the same meaning as in case of the VALUE object.

Text length specifies the maximum number of characters that can be displayed on the screen dependent on the current length byte. If the length byte exceeds the value of the text length, the ∞ character is output instead of the last possible character. Three different display modi can be used:

```
1. Text length > 0: flush left display, whereby
                    the difference between
                    defined text length and the
                    number of read characters
                    is filled with spaces
2. Text length = 0: as 1. but without filling
                    with spaces
3. Text length < 0: flush right
```

Input allowed: Text can be input in the corresponding PASE registers if this field is inscribed by **Y**.

For the rest is valid: only characters from the ordinal number 32 (space) on are displayed respectively accepted as input. Control characters are displayed as reversed question marks ¿.

2.2.2.4 Dynamic

DYNAMIC is not an object that can be drawn. It is a command that can be applied on all objects described so far. If applied, these objects change their colour in dependence upon a bit variable in the PASE controller.

To activate DYNAMIC, click the item DYNAMIC in the DRAW menu. Now click the object that is to have a dynamic attribute.

The first parameter is the desired second colour for the selected object. If a FILL object is selected, it is **not** allowed to use the border colour as the second colour.

The dynamic display is dependent on a bit variable in the PASE controller. The following bit variables can be used: inputs, outputs, flags and individual bits of a register. Specify the type and number of the bit variable; in case of a register bit the definition is register number and bit number (0 .. 23). You can also use indirect addressing (see VALUE object).

The second colour is used, if the bit variable has a value of 1 or TRUE, the normal colour is used otherwise.

Example: The state of the limit switch of a servo axis is to be displayed by a change of the colour of a filled circle. If the limit switch is active the filled area has to be red, green otherwise. To solve that task, the fill object that normally fills the circle with green colour must be made dynamic. The number parameter is the status register of the servo controller (because the limit switch is not connected to an input module). The type of switch is "register" and the bit number is 4 (negative limit switch). The second colour is red.

2.3 Change objects (edit commands):

Available edit commands are MOVE, COPY, CHANGE, CHANGE DYNAMIK, ERASE and END SELECTION. In order to edit objects they first have to be selected.

2.3.1 Object selection

The selection is done with the mouse (left mouse button)

- Lines, rectangles, circles, arcs:
  These objects are selected by pressing the left mouse button with the selection rectangle on any visible part of the object.

- An ellipse has to be selected on any of its four vertices.

- Bargraph, bar, button, clear screen:
  Click on the border of the bar or bargraph. Because clear screen is some sort of bar, it is selected by clicking the border of the drawing area.

- Text, value, text variable, date time:
  Texts and values are selected by clicking their start point. For that purpose, a little dot is drawn at the startpoint of text and value objects. The dot is not visible in automatic mode.

- Fill pattern (fill colour):
  A fill object is selected by clicking the start point. The start point is marked with a little dot that has always another colour than the fill colour. In automatic mode the start point dot is not visible.

- Menu: within the menu box.

- Freeze, include: the "u" key opens a input window in which invisible objects can be selected.

In addition to the selection with the mouse on the drawing area the possibility exists to select the last drawn object. Use the field labelled "Last" to select the last object.

When all desired objects are selected, the process of object selection has to be finished either by clicking the "Endsel." field on the screen or by pressing the space bar on the keyboard.

2.3.2 Move

The command moves one or more objects. The first step is to select the objects that are to be moved. Selection is performed according to 2.3.1. After the selection, a move vector has to be shown by specifying the start and end point ("from where, to where").
MOVE erases the selected objects at their original place and draws them at the new place.

2.3.3 Copy

The steps of this command are equal to MOVE. The original object(s) are not erased however.

Note: The objects CLEARSCREEN and MENU can not be copied.

2.3.4 Change

This command allows to change the parameters of the objects TEXT, VALUE, BARGRAPH, TEXT VARIABLE, BUTTON, MENU, DATE TIME, INCLUDE. After selection of an object, the changeable parameters appear on the command line and/or a parameter window. Confirm the new settings with <Enter> for command line parameters or 'OK' in the parameter window.

2.3.5 Erase

With this command, any object can be erased from the screen. After the Endsel. function (or space bar) the selected objects are erased. Before Endsel. the erase command can be cancelled with the right mouse button.

2.3.6 **Chg.dyn** (change dynamic)

Dynamic objects can be changed with help of this function. The corresponding objects are selected with a cursor. A window appears after Endsel. in which the parameters of the dynamic can be changed. The ERASE object or the "E" key deletes the dynamic of an object.

2.3.7 Last

The previous object is selected by this function. Repeated use of the function selects the previous objects. The LAST function in connection with the SHIFT key selects in reverse direction. A function has to be chosen before (for instance MOVE, CHANGE, etc.).

2.3.8 Endsel.

Every selection of objects for functions like MOVE, ERASE, CHANGE has to be closed with END SELECTION. Then the selected object(s) can be modified, moved, etc.

2.4 Definition of frames

2.4.1 The default values

Font:	Small	
Fontsize:	Width	2.0
	Height	2.0
Line type:	continuous, thin	
Colour:	White	
Fill Shape:	Solid	
Background colour:	Black	

2.4.2 My first frame

Here is a short description of the steps necessary to make a little dynamic frame:

Inside a rectangle is to be a circle that changes its colour according to the state of an output of the PASE.

Output 401 not active	--> green
Output 401 active	--> red

Below the rectangle, a numeric value appears that displays the content of register 8704.

Starting at the DOS prompt you enter the command name of VIADUKT (VIAUK). Now the editor window appears and you can immediately begin to draw. The name of the frame is asked as soon as you save your work for the first time.

2.4.2.1 Drawing of the basic objects

- Select the command 'Rect.' at the right menu and activate it by pressing the left mouse button. Move the mouse to the drawing area and point to the two cornerpoints of the rectangle.

- Select the white colour: move the mouse cursor to the left side of the screen and click on the white button.
  Select the command 'circle' and draw the circle totally inside of the rectangle.

- Fill the circle area with a green colour:
  - Select the green colour.
  - Select the command 'Fill'
  - Point to a point inside the circle area (for instance around the center of the circle).
  - Point to the border of the circle (i.e. select border colour).
  - Select colour white again.

  At this point it is worth to save the work you have done so far. To do this, select the "File" menu at the top line of the screen and choose the "Save" command.

  Now you are asked to enter the file name for the frame. Enter "100". The frame is now saved in a file with the name 100.VIA in the current directory.

2.4.2.2 Dynamic

- In order to establish the dependence between the colour of the circle area and output 401, the DYNAMIC command is now used.

- Click 'Dynamic' in the 'Draw' pull down menu.
- Answer the 'Select objects' prompt with a click on the pick point of the fill object. You will hear a short beep. This is the confirmation that the fill object was selected. End selection process by pressing the space bar.
- Select colour red as the second colour.
- Enter 401 in the 'Number' parameter field and Output in the 'Switch' parameter field.

2.4.2.3 Definition of a VALUE object to display register 8704

- The input data for a value object in this example are:
    - Click the 'Value' command.
    - Point to the start point.
    - Enter the parameter data: enter the number of the register that is to be polled periodically (in this example: 8704).
    - Enter number of digits to the left of the decimal point: 5.
    - Enter number of digits to the right of the decimal point: 0.
    - Display multiplier and offset need no change. Leave the default values 1.00 and 0.00 unchanged.
    - Answer 'Allow input' with N (No).
    - Close the parameter window with the 'OK' button.

The frame of this example is ready now. Save it again using 'Save' in the 'File' pull down menu. You can review the actual file name with the 'Info' command in the 'File' pull down menu.

To have a look at the frame in the full size of the screen, choose the PREVIEW command in the pull down menu 'Exit'. The frame that is currently loaded in the editor is displayed in the way as it will be displayed later on in automatic mode, i.e. zoomed to fill the whole screen.

In order to activate the example frame in automatic mode, it is necessary to have a PASE connected to the VIADUKT. Start the automatic mode with 'Automatic' in the 'Exit' pull down menu. The frame is displayed on the screen, if the register 1 contains the value 100. Enter that value into register 1 for instance with the SYMPAS set up mode.

2.4.3 Example SET EDITOR (with indirect addressing)

A typical application of the VIADUKT system is the input/output of data sets. The following example demonstrates an input/output frame for 100 data sets of 4 parameters each.

Task:

A 2 axes robot is to move to 100 positions, one after another. These positions have to be displayed at the screen. The operator is able to change any value at the screen.
--> 100 data sets has to be displayed, each of them with the following 4 parameters:

X-position, speed X, Y-position, speed Y

These variables have to be stored in the PASE in registers. The order of storage is to be:

Set 1: Reg. 1000:Position. X
Reg. 1001:Speed X
Reg. 1002:Position Y

Reg. 1003:Speed Y

Set 2: Reg. 1004:Position X
Reg. 1005:Speed X
Reg. 1006:Position Y
Reg. 1007:Speed Y

.....

Set 100: Reg. 1396:Position X
Reg. 1397:Speed X
Reg. 1398:Position Y
Reg. 1399:Speed Y

Now a frame has to be designed that can display 5 sets of data at the same time. The number of the first set can be chosen by the operator.

The solution uses indirect addressing in the VIADUKT: a pointer register is reserved (register 10 in this example). Using the display multiplier (= 0.25) and the display offset (= -249), the register number can be calculated out of the set number of the first set that is to be displayed.

register 10 = (set number - offset) / multiplier

Example: if set number 2 has to be displayed the pointer is calculated with

register 10 = (2 - -249) / 0.25 = 1004

1004 is the register number for the x position of set 2.

The parameters that are to be displayed (in the example, these are the sets 2, 3, 4, 5 and 6) are stored in the registers starting with register 1004. It is therefore sufficient to refer to register 10 and address the individual parameters with ascending register offsets. The frame can be defined in the following manner:

```
         set number:                register    = 10
         ===========                multiplier  = 0,25
                                    displayoffs.= -249

 set           X-position   X-speed      Y-position   Y-speed
 ============================================================

 Reg=10        Reg=R10      Reg=R10      Reg=R10      Reg=R10
 RegOffset=0   RegOffset=0  RegOffset=1  RegOffset=2  RegOffset=3
 multipl=0.25
 dispoff=-249

 Reg=10        Reg=R10      Reg=R10      Reg=R10      Reg=R10
 RegOffset=0   RegOffset=4  RegOffset=5  RegOffset=6  RegOffset=7
 multipl=0.25
 dispoff=-248

 Reg=10        Reg=R10      Reg=R10      Reg=R10      Reg=R10
 RegOffset=0   RegOffset=8  RegOffset=9  RegOffset=10 RegOffset=11
 multipl=0.25
 dispoff=-247

 Reg=10        Reg=R10      Reg=R10      Reg=R10      Reg=R10
 RegOffset=0   RegOffset=12 RegOffset=13 RegOffset=14 RegOffset=15
 multipl=0.25
 dispoff=-246

 Reg=10        Reg=R10      Reg=R10      Reg=R10      Reg=R10
 RegOffset=0   RegOffset=16 RegOffset=17 RegOffset=18 RegOffset=19
 multipl=0.25
 dispoff=-245
```

In the last set displayed, the following registers are used (example, set 2 is chosen, therefore register 10 = 1004):

setnumber	: (1004 * 0,25) + -245 = 6
position X	: content of reg(1004 + 16), i.e. content of R1020
speed X	: content of reg(1004 + 17), i.e. content of R1021
position Y	: content of reg(1004 + 18), i.e. content of R1022
speed Y	: content of reg(1004 + 19), i.e. content of R1023

2.5 DOS Command line parameters

If VIADUKT is started without parameter /A the editor is activated and the user interface described in chapter 2.1 is displayed:

VIAUK

If the hardware lock is not plugged the demo version of the VIADUKT is started.

By using one or more command line parameters at VIADUKT start, you can influence the behaviour of VIADUKT.

VIAUK /**A**

The program directly starts the automatic mode. In this case no mouse driver is necessary to run the program.
This parameter is normally specified after all frames are defined and the VIADUKT is installed at the end user's machine.
The JETTER VIADUKT (Hardware) PC has an AUTOEXEC.BAT file that includes this command at the end. If the PC is switched on the VIADUKT program is started automatically in automatic mode.

VIAUK /**Rxxxx**

Number of the PASE register that contains the number of the frame that has to be displayed. If you omit this parameter, register 1 is used (default setting).
VIADUKT periodically scans one register of the PASE to determine the number of the frame that has to be displayed. If parameter /R is not specified VIADUKT uses register 1 for this purpose. If for any reason another register is to have this function, then its number has to be specified at VIADUKT start. If for instance register 1000 has to be used -> VIAUK /R1000 (see also 3.1).

VIAUK /**Gaaa**

VIADUKT automatically detects the type of graphic adapter present on the computer system (EGA or VGA). Using the /GEGA parameter, you can prescribe that EGA is used even on systems with VGA.
"aaa" stands for adapter name. "EGA" or "VGA" are valid names.

VIAUK /**Fxxxx**

Number of the first flag that is influenced by the function keys of the VIADUKT. Without specifying this parameter, function key F1 is copied to flag 201, F2 to flag 202 and so on.

VIAUK /**Dnnnn**

Activation of the data transfer function. The number to be specified here is the first register number of the transfer control block (see chapter 6).

VIAUK /**Px**

Selection of the controller type the VIADUKT is connected to in automatic mode.
"x" stands for type of PLC. Possible values are "E" for PASE-E and "J" for PASE-Junior.

If this parameter is not specified the PASE-E settings are selected.

VIAUK /**X <cc> or <file name>**

This command line parameter specifies the VIADUKT graphics adapter. Two possibilities have to be distinguished. First a number defines the colour of the crosshair respectively the input frame. Second a definition file determines a wide range of specifications of the graphics adapter.

VIAUK /**Cxxxx** (not VIADUKT lite)

This function transfers the time and the date from the VIADUKT hardware clock into the PASE controller. The **xxxx** parameter defines the first register of a register block in the PASE controller that is used to store the corresponding information:

```
xxxx    = hand shake register (0: no information transfer,
          1: information transfer, confirmation with 0)
xxxx+1  = minutes (0..59)
xxxx+2  = hours   (0..23)
xxxx+3  = day of the week (0 = sunday, ...)
xxxx+4  = day     (1..31)
xxxx+5  = month   (1..12)
xxxx+6  = year    (1980..2099)
```

Following command line parameters are described in the corresponding sections respectively manuals:

- /W, /H, /L refer to message editor description.
- /U, /V refer to chapter extended functions by TSR programs respectively the documentation file USERPROG.PAS.
- /E refer to chapter DOS functions.

It is possible to use more than one of the parameters at the same time. Example: start VIADUKT in automatic mode, use register 99 as nominal frame register and use flags 1 to 14 for the function keys F1 to F14:

VIAUK /A /R99 /F1

For the daily work with VIADUKT exists a possibility to store the command line parameters you often need in a file. If you specify the filename of this file as the first parameter of the command line, then VIADUKT will go through the file and use the parameters in the file, then use the parameters that are on the command line:

Example: The file M.CMD contains the following line:

```
/R99 /F1 /D100
```

Call VIADUKT from DOS with the following line

```
VIAUK M.CMD /A
```

First the file M.CMD is read and interpreted (use register 99, function key flags are 1 to 14 and activate data transfer function). Then the rest of the command line is read: /A consequently the automatic mode is started.

2.5.1 Updates (new program versions of VIADUKT)

Like every other program, VIADUKT may contain bugs that are not yet detected when VIADUKT is delivered. To simplify the task of installing a new program version, VIADUKT provides the possibility to automate the update. This is especially important for the JETTER VIADUKT hardware PC, because this PC does not have an alphanumeric keyboard, so it is not possible for instance to enter a command like "COPY A:*.* C:".

With the build-in Update function, a program version update is easy and can be done by the end user.
The command line parameter /**A** has an additional option:
If /A is specified with an additional number between 1 and 9 (for instance VIAUK /A5), then the operator has a time of 5 seconds after the start of the VIADUKT to request an update. To request the update, the operator must press F6 within the specified time. VIADUKT then requests the operator to insert the disk with the new program version into the disk drive and then press the ENTER key.

From a programmer's point of view: The VIADUKT program is stopped and a DOS ERRORLEVEL of 97 is returned to DOS. The program that started VIADUKT (normally the AUTOEXEC.BAT) has to take care now to copy the new program version.

The batch file UPDATEXP.BAT that demonstrates the use of this function is contained on the release disk of VIADUKT.

2.6 Function keys

2.6.1 Basic functions of the function keys

The function keys of the VIADUKT can be used directly to control the program flow in the PASE program. For this purpose, the function keys are copied to flags in the PASE. Without using the command line parameter /F the following table is valid:

F1 flag 201
F2 flag 202
F3 flag 203
....
F14 flag 214

Using the command line parameter /F another starting flag number can be defined (see 2.5).

If a function key is pressed, then the corresponding flag in the PASE is set. When the key is released again, then the flag is reset.

If the function keys F11 .. F14 are not physically present, their function can be activated using SHIFT F1 to SHIFT F4.

2.6.2 Menu function

With the menu function (activation in the 'Draw' pull down menu) the task of labelling function keys is simplified. Additionally menu structures can be defined that are independent of the program flow in the PASE controller.

2.6.2.1 Pure labels

Label fields appear at the right side and at the bottom of the screen as rectangle with two short text strings. If necessary, they can be moved with the move command.

Label fields for function keys F1 to F8 appear at the bottom and F9 to F14 at the right side of the screen (corresponding to the layout of the frontpanel of the JETTER VIADUKT PC).

To define a label field:

- o Click 'Menu' in the 'Draw' pulldown menu. A parameter window appears.
- o Specifiy the function key number (1 to 14).
- o Enter the text string for the first line of the label (max. 8 characters).
- o Enter the text string for the second line.

Note: If no text is specified, VIADUKT does **not** draw a border line. This is useful, if a label field has a frame call function (see below) and if you want to write menu texts in a height as you like.

2.6.2.2 Label field with frame call-up

The sequence of inputs is the same as in 2.6.2.1. Additionally the number of a frame is specified that has to be displayed, if the function key is pressed. VIADUKT will write this number into the actual frame register (default: register 3).

The frame called with the function key can again contain menu objects, which assign new functions to the function keys. In this way it is possible to create a tree of menues with different frames that can be called independently of the PASE program.

The menu object of a function key remains active until a new menu object with the same function key number redefines it or until a frame is loaded that contains a CLEARSCREEN command. After a CLEARSCREEN command was executed, all menu objects are erased.

3. Automatic mode

Automatic mode can be started with

Start VIAUK with command line parameter /A

or

Select 'Automatic' in the 'Auto' pull down menu.

To stop the automatic mode, press the two keys 'Ctrl' and 'End' on the keyboard simultaneously. If automatic mode has been started from the editor, then the program will return to the editor. If automatic mode has been started from DOS with /A, then VIADUKT will stop and return to DOS.

VIADUKT has the following functions in automatic mode.

- Periodically read the register that contains the number of the frame to be displayed. If this number changes, the new number is treated as a file name and the frame stored in that file is displayed (if the file exists). (3.1).

- Scan of the function keys. Pressing or releasing a function key will set or reset the corresponding flag.
  If a menu object is defined for the function key and a frame number is assigned to that key, then the frame is loaded and displayed and the number is stored into the actual frame register in the PASE (2.6).

- If input is allowed for value parameters currently on the screen, then numeric input is accepted from the operator. (3.3).

- Periodically read the registers of the data transfer control block if /D command line parameter was specified. If there is a command in the control block, this command is executed (6).

- o the display of message windows (command line parameters /W, /H, /L).

- o time transfer to the PASE (call-up parameter /C).

If the PASE controller is not connected to the VIADUKT or if there are problems with the communication between PC and PASE, then an error message appears in the middle of the screen: 'PASE offline'.

In that case, the automatic mode can be stopped in the same way with 'Ctrl + End'.

3.1 Display of frames

In automatic mode, VIADUKT periodically reads the **nominal frame register** which contains the frame number. Normally this is register No. 1. With command line parameter /R however, any register can have this function (2.5).
In any case, the two register following the nominal frame register are also occupied by the VIADUKT software. These are the **confirmation register** and the **actual frame register**.

VIADUKT searches for a file which name is consists of the number stored in the nominal frame register and the extension .VIA that is appended to the number. If the file was found by the VIADUKT and was displayed successfully the frame number is stored into the confirmation register. Also the actual frame register contains the frame number.

If the requested frame is not available (file not found) or there was an error during loading of the file the confirmation register is set to -1.

3.2 Function keys

The function keys of the JETTER VIADUKT front panel (F1 .. F14) or the keys of a PC keyboard (F1 .. F12) are used to control certain flags in the PASE controller. If a function key is pressed, then the corresponding flag is set. When the function key is released, then the flag is reset. The relationship between function key number and flags number is :

F1 flag 201
F2 flag 202
F3 flag 203
....
F14 flag 214

Using the command line parameter /F another starting flag number can be specified (2.5).

If there are less than 14 function keys available physically, the flags can be controlled using the SHIFT key:

F11 = SHIFT F1 flag 211
F12 = SHIFT F2 flag 212
F13 = SHIFT F3 flag 213
F14 = SHIFT F4 flag 214

3.3 Numeric input

If the frame that is currently displayed contains VALUE objects that allow input, then the content of the register specified in the value object definition can be changed by the operator.

For this purpose, a CURSOR can be moved around the screen onto every value object that has the 'input allowed' parameter set to YES. The cursor is a rectangle that surrounds the input field.
After a new frame was displayed, the cursor is located on the first value object which allows input, but is invisible. The first object is the one that is displayed nearest to the top left corner of the screen.

The cursor is visible, as soon as one of the following keys is pressed:

ENTER	"opens" the input field of the actual value object. The default value in the input field is the current value.
arrow keys	Moves the cursor to the next value object in the direction of the arrow.
0 to 9	Opens the input field of the actual value object. The digit that opened the field is treated as first character of the new value, therefore the actual value which normally would have been displayed as default value is lost.

The input field remains active until it is closed with one of the keys described below. As long as the input field is active, this variable is excluded from the dynamic refresh of value objects.

Following keys are valid during input:

0 to 9	input of a number

BACKSPACE erases the last digit entered

DELETE or C erases the entire input field. The number that was displayed at the begin of the input sequence is displayed again.

The input field is closed with one of the following keys:

ENTER Ends input, cursor remains on that value object but becomes invisible.

arrow keys Ends input, cursor moves to the next value object input field in the direction of the arrow (if one exists). See also 3.3.1 cursor logic.

ESC (= F14 on JETTER VIADUKT Hardware)
Ends input. The value in the input field is not transmitted to the PASE. The cursor remains on that value object but becomes invisible.

If the ENTER key or the arrow keys are pressed, the numeric input is checked against the boundary in the definition of the value object. If the value is within the boundary it is accepted and transferred to the PASE (first the offset is subtracted, the result is divided by the multiplier and that result is transferred to the PASE).

If the value is not within the range specified in the value object definition, it is impossible to leave the input field. A valid number must be entered first. With the DELETE key, the original value can be restored in the input field.

3.3.1 Cursor logic

The rules by which the cursor is moved if an arrow key is pressed depend on the definition point of the value objects. If the next field is searched, the one(s) that are in the same column or row (according to the arrow direction) have the first priority. If this is desired, the easiest way to force it is to use the snap function during the definition of the value objects.

If there is no value object in the arrow direction on the same row/column, then a value object is searched that has a definition point within 90 degrees in the direction of the arrow. If more than one are found, the one that is closest to the current input field is used.

If there is still no value object within 90 degress, an area of 180 degrees is used. If no object is found on that side of the current input field, then the cursor is not moved at all, but becomes invisible.

3.4 Monitor functions

The monitor functions of the VIADUKT are comparable to the functionality of the keyboard/display modules of the PASE system (refer to PASE-E user's manual chapter III / 7). Its possible to read and write registers, flags, outputs and read inputs.

The monitor functions are available in automatic mode. They are activated with the "R" or "I" key (key labelled "I/O" on the JETTER VIADUKT PC). If one of these keys is pressed, a window will open in the middle of the screen. While a monitor function is active, the "background" of the screen is "freezed", i.e. no periodic update of objects will occur.

Register functions are always started with the "R" key. Double-pressing the "R" starts **flag functions**. Pressing the "R" key three times activates the **text variable function**.
Similar functions are started with "I" for **outputs** and **inputs**.

The next step is to enter the number of the register, flag, input or output. The following value ranges are valid:

0 - 65535 register, whereby	
50000 - 59999 are network registers	(PASE-E, PASE-Mikro)
0 - 4095 flag	(PASE-E)
1 - 2655 flag	(PASE-Mikro)
101 - 3216 output, input	(PASE-E)
1 - 128 input	(PASE-Mikro)
1 - 80 output	(PASE-Mikro)

The ENTER key ends the input and the content of the desired item is displayed on the screen and refreshed continuously.
This is the display mode of the monitor functions. In this mode several key strokes are possible:

"R"	register, flag or text variable function (mentioned above).
"I"	input, output functions (mentioned above).
"SHIFT-R"	current program line, task state display, see remark below.
"ESC", "ENTER"	Monitor function window is closed and process monitoring is resumed.

"=", SPACEBAR

If the selected item is **not** INPUT and if the input of values for the other types is not forbidden via register 8224 (PASE-E), register 65 (PASE-Mikro) then these keys start the input sequence for a new value for the selected item. The following keys are available for the input sequence:

0..9, -	:	Number input
"."	:	Decimal point for floating point registers (8960..9015; PASE-E only)
"Del","C"	:	If input string = "0" -> quit input, return to display mode or monitor functions. If input string <> "0" -> input string is set to "0".
"Esc":		Quit input and return to display mode of monitor functions.
"ENTER"	:	Accept input string and transfer value to PASE. Flags, output: input = 0 --> transferred value = 0 input <> 0 --> transferred value = 1 Register: transferred value = input Then return to display mode of monitor functions.
"R","I"	:	Input a new number

Remark to **SHIFT-R**: the RT file of the current PASE program is necessary that is created at compilation of the PASE program by SYMPAS. VIADUKT has to be informed about the file path if the RT file is not placed in the actual directory. This is realized by the DOS command:

```
SET RT_DIRECTORY=path
```

4. Reserved PASE register ranges

The PASE register ranges that are used by the VIADUKT are described here. The beginning of the register block in the PASE that controls the cooperation between PASE and VIADUKT can be defined by the user (command line parameter /Rxxxx). The default setting for the beginning of the register block is register 1 (**PASE register 1**).

The register block that controls the cooperation of PASE and VIADUKT:

```
register x     =   nominal frame register
         x+1   =   confirmation register
         x+2   =   actual frame register
         x+3   =   status register (VIADUKT)
         x+4   =   frame offset register
         x+5   =   status register (PASE)
         x+6   =   input register
         x+7   =   reserved
         x+8   =   reserved
         x+9   =   reserved
```

Nominal frame register

The number of the frame that is displayed in automatic mode is stored in this register. The frame number contained in the register corresponds to a VIA file. If the nominal frame register contains 110 the frame with the file name 110.VIA is displayed on the VIADUKT screen in automatic mode.

Confirmation register

VIADUKT writes into the confirmation register that frame number which is currently displayed on the VIADUKT screen (automatic mode). -1 is stored into the confirmation register if the requested frame is not displayed on the VIADUKT screen.

Actual frame register

If VIADUKT branches from a frame that was requested by the nominal frame register to a further frame requested by a menu function, the number of the base frame remains in the confirmation register. Thus the values of the nominal frame register and the confirmation register are identical. The actual frame register, however, stores the number of that frame which was started from the base frame by the menu function (automatic mode).

Status register (VIADUKT)

The bits of the register have following meaning:

```
bit 0 :  0 = no input active
         1 = a VALUE or TEXT VARIABLE object is input
             currently
```

Frame offset register

This value is added to the content of the nominal frame register. The sum defines the number of the frame that is represented on the VIADUKT screen (automatic mode).

Status register (PASE)

The bits of the register have following meaning:

```
bit 0 :  0 = the input into a VALUE or TEXT VARIABLE object is
             allowed
         1 = the input is not allowed - ongoing inputs are
             terminated

bit 1 :  0 = the content of the read and write register is
             directly used as file name (see data transfer)
         1 = the content of the read and write register is
             used as pointer to a file name (a file selection
             window is offered if the file name contains
             wild cards).

bit 2 :  0 = see bit 2 = 1
         1 = if a DA file was activated in a selection window
             the file name is stored into the text register

bit 3 :  0 = see bit 3 = 1
         1 = an additional flag is set if a frame is called up
             by the menu object (this bit has to be defined
             before VIADUKT is started)

bit 4 :  0 = the file name is renewed created. The old file
             with the same name is overwritten
         1 = the data block is appended to an already
             existing file with the specified file name. The
             file is created if no such file name exists
```

Input register

The number of the register that is input with a VALUE or TEXT VARIABLE objects appears in the input register. The value -1 in the input register means that no input takes place.

5. Reserved PASE flag ranges

The PASE flag ranges that are used by the VIADUKT are described here. The beginning of the flag block in the PASE that controls the cooperation between PASE and VIADUKT can be defined by the user (command line parameter /Fxxxx). The default setting for the beginning of the flag block is flag 201 (**PASE flag 201**).

The flag block that controls the cooperation of PASE and VIADUKT:

VIADUKT function keys	**PASE flags**
F1	201
F2	202
F3	203
F4	204
F5	205
F6	206
F7	207
F8	208
F9	209
F10	210
F11	211
F12	212
F13	213
F14	214
↑	221
↓	222
←	223
→	224
Page ↑	225
Page ↓	226

The PASE flags 201 to 250 are reserved for the use of the VIADUKT.

6. Data transfer function (not VIADUKT lite)

For modern machines that have to handle a variety of different tools or workpieces, the PASE-E system provides registers to store about 5000 (PASE-E Plus CPU: 28000) values. If this is not enough, the data transfer function allows a PASE program to use the harddisk of the VIADUKT as an additional register storage device. The harddisk is used as an external memory device. The PASE can issue commands to read and write files on the harddisk. Before a file is written, the PASE can determine the register range that is stored on disk and specify a the filename for the register range file.

The data transfer function is only available, if the command line parameter /**D** was specified when the VIADUKT software was started. After the letter D the number of the first register of the data transfer function control block has to be specified.

The data transfer function control block is a block of 4 adjacent registers that have the following meaning. If the command line is:

VIAUK /D200

then the control block is located from registers 200 to 203 and the registers have the following names:

200 Read register
201 Write register
202 Start register number
203 End register number

The numbers correspond to the command line parameter /**D200**.

While automatic mode the VIADUKT periodically reads the registers "Read register" and "Write register". These registers have to be used by the PASE program to request a data transfer by the VIADUKT.

In detail the registers have following functions:

Read register

If the PASE program writes a value greater than 0 into this register, this is a command for the VIADUKT to read a data file on the harddisk and transfer its content to the PASE. The name of the file is equal to the content of the write register and has the extension ".DA".
A file of this type and structure can be created with the data transfer function "Write" or with the programming software for PASE systems ("SYMPAS").
When the transfer has finished, VIADUKT sets the read register to zero. If an error occured during the transfer the error code is written into the read register:

```
   0 = no data transfer error
  -1 = file selection window was terminated with ESC
  -2 = file not found
  -3 = directory not found
 -18 = the directory contains no DA files
-101 = insufficient disk space
-150 = floppy disk write protected
-152 = floppy drive not ready
```

Write register

If the PASE sets this register to a value greater than 0, this is the command for the VIADUKT to transfer data from the PASE to a file on the harddisk. The name for the file is numeric and is equal to the content of the write register plus the extension ".DA". Data is written to that file in a format that can be read with the data transfer function "Read" and with the SYMPAS program function "File.DA -> register".

When all data is written to the selected file, the VIADUKT sets the write register to zero. If an error occured during the transfer, the write register is set to a negative number (error codes mentioned above). If the DA file is not placed in the current file VIADUKT can be informed about the corresponding directory:

```
SET DA_DIRECTORY=path
```

Start register number

This register contains the number of the first register of the PASE where data has to be read when the write function is activated.

End register number

This register contains the number of the last register of the PASE where data has to be read when the write function is activated.

Example:

The contents of registers 100 to 140 have to be saved onto harddisk. Using the example above (command line parameter /D200), the PASE programm must first load the start and end register numbers:

```
REGISTER_LOAD [ 202 with 100 ]
REGISTER_LOAD [ 203 with 140 ]
```

Then the data transfer can be started with the assignment of the filename to the write register:

```
REGISTER_LOAD [ 201 with 3333 ]
```

Now the file "3333.DA" is opened on the harddisk, the contents of registers 100 to 140 are written to that file. The file is then closed and the value 0 is written into register 201 (the write register). This is the handshake for the PASE. The PASE program can wait for the completion of the write process for instance with the lines:

```
WHEN
      REG 201
      <
       1
 THEN
      IF
      REG 201
      <
       0
 THEN
  error code....
```

Further functions can be controlled by the status register (PASE) with the bit 1, 2 and 4 (for detailed information refer to chapter 4. Reserved PASE register ranges).

Status register (PASE)

```
bit 1 :  0 = the content of the read and write register is
             directly used as file name (see data transfer)
         1 = the content of the read and write register is
             used as pointer to a file name (a file selection
             window is offered if the file name contains
             wild cards).

bit 2 :  0 = see bit 2 = 1
         1 = if a DA file was activated in a selection window
             the file name is stored into the text register

bit 4 :  0 = the file name is renewed created. The old file
             with the same name is overwritten
         1 = the data block is appended to an already
             existing file with the specified file name. The
             file is created if no such file name exists
```

7. DOS functions (not VIADUKT lite)

This application makes DOS functions available that can be selected in a window by the operator of the VIADUKT process monitoring system.

Following DOS functions can be specified in a instruction file by the user:

- **Dir**
- **Copy**
- **Del**
- the execution of **bat**ch files
- the call-up of a **DOS shell** (PC keyboard required)

The desired DOS functions are integrated into the VIADUKT by the user with help of instruction files. These instruction files generate at runtime selection windows if the operator pushes a certain VIADUKT function key. The different DOS functions can be selected in the window that appeared.
The user defines the DOS functions which are available to the operator at runtime in instruction files by key words.

The instruction file and thus the contained DOS functions has to be declared to the VIADUKT at start-up with a command line parameter. Before the instruction file was created by the user in any ASCII editor.

Following **key words** are defined:

1. **WinTitle** titel of the main selection window (max. 36 characters)

2. **UserKey** key (combination) for activation of the function. Valid keys are:
   - o F1 to F12
   - o SHIFT key

3. **Lock** **Y** = the key for activation of the DOS functions has to be defined in the current VIADUKT frame as menu key. In this case no SHIFT key combination has to be specified.

   N = (default setting) the key for the DOS functions opens the window defined in the instruction file without recognizing the content of the current VIADUKT frame. Here the combination between SHIFT key and function key should be used, to leave the not combined function keys to the menu function.

4. **Function** available DOS commands and functions are:
   - o Dir
   - o Copy
   - o Del
   - o Convert
   - o Batch
   - o Shell

4.a **MenuTitle** the title line of the selection window that displays the different DOS functions is specified here.

4.b **SourcePath** the source directory of the function is defined here:
source directory + name (wild cards possible)

4.c **DestPath** the destination directory of the function or in case of the BATCH function the parameters determined here:
dest. directory [*.EXT] or parameter (Batch)

4.d **AllText** the DOS function seizes all files with identical file names using the AllText line, if wild cards are used (see also example).

4.e **PostErase** only available for CONVERT function. The source files are deleted after the conversion.

5. **Position** 4 parameters succeed the key word that have following meaning:

- X: this parameter defines the x coordinate of the left, upper window corner (if the parameter is 0 the window is centered on the axis).
- Y: this parameter defines the y coordinate of the left, upper window corner (if the parameter is 0 the window is centered on the axis).
- B: defines the width of the message window (min. 15, max. 70).
- H: specifies the window height (2..12).

Following conventions have to be met in order to write instruction files with the ASCII editor:

- a space has to be placed between key word and key word parameter; in addition spaces can be used for optical arrangement.
- commentaries should be introduced with semicolon.
- lines with key words must not contain commentaries.
- incomplete or incorrect key word descriptions are ignored and are not included in the selection menu.
- error messages are output at runtime.

Following **command line parameter** has to be defined at VIADUKT call-up:

VIAGR /a /**eFileName.Ext**

An **example** explains the facts:

Task: all files with the extension EX1 that are placed in the C:\TESTDIR1 directory have to be copied into the C:\TESTDIR2 directory. Thereby the extension EX1 has to be renamed into EX2.

The user now defines the structure of the selection window in a instruction file. The window can be opened by the operator with help of a function key to select the desired DOS function.

This instruction file can be created in any ASCII editor. For instance the file contains following lines:

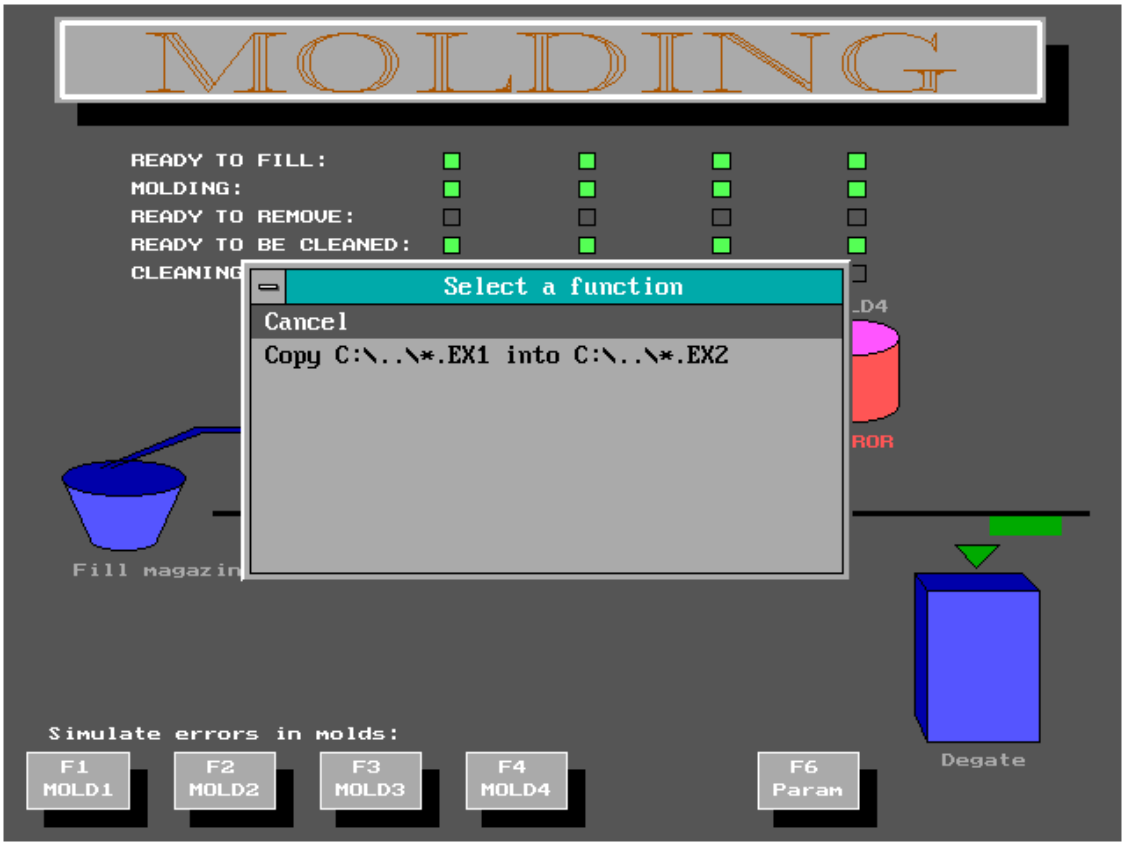
```
;File DOSFUNC1.DFC
;
WinTitle = Select a function
UserKey = F10 shift
;
;
Function = Copy
MenuTitle = Copy C:\..\*.EX1 into C:\..\*.EX2
SourcePath = c:\testdir1\*.EX1
DestPath = c:\testdir2\*.EX2
AllText = All files
;
```

Now this file can be stored using any file name. The file name DOSFUNC1.DFC is chosen for this example.
This file either has to be placed in the same directory as VIAUK.EXE or the complete path of the instruction file has to be specified in the command line parameter.

Then **VIADUKT** is started with following **command line parameter**:

VIAUK /a /e**DOSFUNC1.DFC**

Following selection window appears in the current frame if the operator pushes the SHIFT F10 key combination:



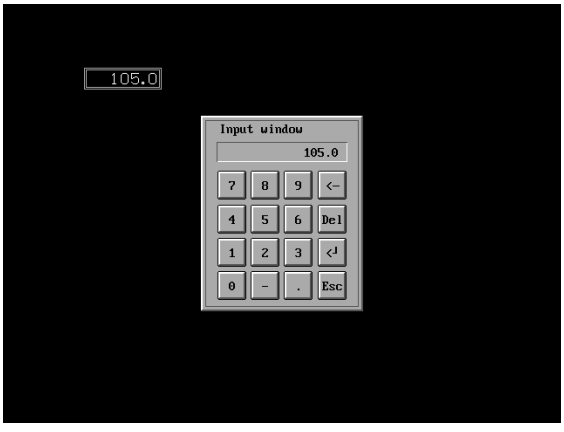
The operator can now select the desired function in the window. The example represented here provides the copy function mentioned above and the cancel function for the selection window itself. Any number of DOS functions can be made available to the operator in a selection window by creating corresponding instruction files.

8. Input in VALUE or TEXT VARIABLE objects via mouse (automatic mode)

The operator can enter data into VALUE or texts into TEXT VARIABLE objects in automatic mode. The different input fields are activated in a certain internally defined sequence by the cursor keys. The changed values can be confirmed (ENTER) or the value input can be terminated (ESC).

The input into VALUE or TEXT VARIABLE objects can be realized via mouse if the VIADUKT is also during automatic mode equipped with a mouse, track ball or a similar device. Thereto the operator has to point with the mouse to the number value or text that have to be modified. Corresponding to the type of input object either a numeric key pad or a keyboard opens. Using this input devices the operator can enter the necessary data.

VALUE or **TEXT** input with the respective windows:



Point with the mouse to the text or the number that is be modified and edit the text respectively the number in the input window.

Both input windows as well the key pad for number input as the keyboard for text input can be created by the user and thus supplied with characters that are not available in the VIADUKT standard input windows (available on disk).

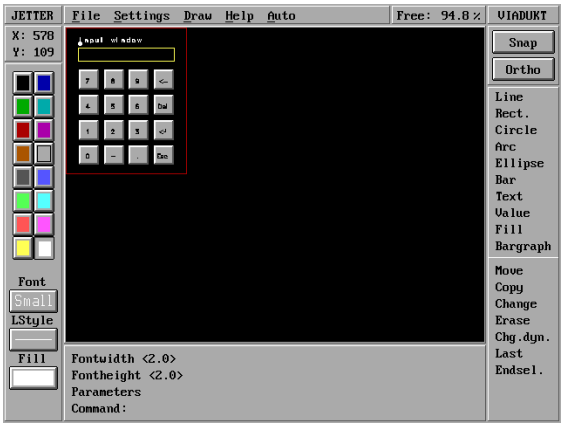
If the operator of the VIADUKT points to a VALUE object with the mouse, VIADUKT searches for a file with the name INNUMBER.VIA. If VIADUKT finds such a file, INNUMBER.VIA is opened in form of the key pad mentioned above.

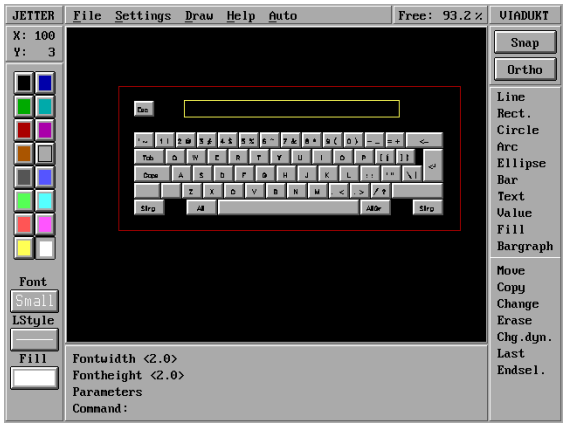
If the operator of the VIADUKT points to a TEXT VARIABLE object with the mouse, VIADUKT searches for a file with the name INSTRING.VIA. If VIADUKT finds such a file, INSTRING.VIA is opened in form of the keyboard mentioned above.

As well the file INNUMBER.VIA as the file INSTRING.VIA are common VIADUKT frames that, however, contain only certain objects.

Their special file names define them as base frames for the both input windows mentioned above.

The following figures show INNUMBER.VIA and INSTRING.VIA as frames in the editor:





Some conventions have to be kept to that the frames are accepted as input windows respectively that they correctly operate. Besides the frame for the number input window has to be stored with the name INNUMBER.VIA, the frame for the text input window with the name INSTRING.VIA.

Rules for input frame creation:

- only RECTANGLE, BUTTON and TEXT objects are noticed.

- the RECTANGLE object defines as well the size of the dialogue window as the size of the input window.
  - dialogue window size: for this a rectangle has to be defined in red colour.
  - input window size: for this a rectangle has to be defined in yellow colour.

  The position of the input window on the editor screen is not relevant since the window is centered in automatic mode automatically.

- any number of input keys can be defined with the **BUTTON** object. Beneath position and size of the button a inscription text and a code can be defined that is evaluated after mouse click to the corresponding button (key).
  The code has to be defined in the "flag number" input field. The code for the button pushed together with a defined CAPS LOCK button has to be entered into the "Offset" field.

  Following key **codes** are supported.

  o control characters are:

  8 = BACKSPACE
  13 = ENTER
  27 = ESCAPE

  o further the ASCII characters 32 to 255 are valid.

- code greater than 255 are evaluated as extended keyboard codes:
  313 = CAPS LOCK: if a letter between 'a' and 'z' (code 97 to 122) is clicked during activity of CAPS LOCK the corresponding capitel letter is generated. **Exception:** a value in the offset field has priority if such a parameter was specified.
  338 = DEL: deletes the complete input and shows the previous value.

- any texts can be created within the input window with the **TEXT** object.

- the use of the "Normal" font is recommended to guarantee the same size relations on the editor screen as in the input window of the automatic mode.

9. PCX files

PCX files can be integrated into VIADUKT frames (monochrome or 16 colours). These PCX files can be used as background frames in which further elements can be inserted.

VIADUKT also features the creation of hardcopies from the VIADUKT screen (preview and automatic) that can be stored as PCX files onto the hard disk and for instance used for documentation purposes.

The **CTRL-P** key combination starts the hardcopies from the preview respectively automatic screens. A window appears for detailed specification of the hardcopy. Then the file name has to be determined in a second window and the file is stored onto hard disk.

The "PCX file" window offers following selection possibilities:

Whole screen	the complete screen is copied.
Active window	the active window is copied.
Section (mouse)	a section defined via mouse is copied.
Active window (0/0)	the active window is copied. Additionally the coordinates are projected to the origin.
Section (0/0)	a section defined via mouse is copied. Additionally the coordinates are projected to the origin.

The **INCLUDE** function integrates the PCX files into a frame ("Draw" menu). The user can choose between the import formates .VIA and .PCX.
If the user integrates one of both file formats into the frame using the INCLUDE function, so the import files - the .VIA or .PCX files - become no direct part of the current frame. The current frame only contains an include remark of the corresponding file that has to be integrated. Thus the difference between "Load frame" and the including of a .VIA file becomes comprehensible (this is not valid for PCX files since they can only be integrated by the INCLUDE function).

A frame that is additionally loaded into an existing frame ("Load frame") becomes an integral part of the base frame, its data are added to the base frame and stored onto hard disk.

Only an include remark is inserted into the base frame if an additional frame is integrated with INCLUDE. The data of the second frame remain in the original file.

PCX files can be created with any commercial design software (Paintbrush, CorelDraw, etc.) and integrated into the VIADUKT.

If Paintbrush (Windows) is used the palette file VWINDOWS.COL should be loaded into Paintbrush to guarantee a maximum of compatibility between Paintbrush and VIADUKT colours.

Besides the Paintbrush files have to be converted with PCX16VIA.EXE before they are imported into the VIADUKT. For detailed information refer to the PCX16VIA.DOC file.

10. Runtime error messages in any foreign language

Error messages that VIADUKT delivers during runtime in automatic mode can be translated into any foreign language. Thereto the VIAMSG.MSG file has to be modified. The english error messages are overwritten with their foreign language counterparts whereby the line scheme must be preserved.
Also the total number of lines of the VIAMSG.MSG must not be changed. Else the file is ignored and no error messages are displayed.

11. Function extension by TSR programs (not VIADUKT lite)

VIADUKT can communicate with a TSR program written by the user optionally. The creation of such a TSR program by the user requires comprehensive programming knowledge. For detailed information refer to the **USERPROG.PAS** file available on disk.

12. Graphics adapter configuration

Following command line parameters adapt the VIADUKT to the graphics adapter.

VIAUK /**X <cc> or <file name>**

> This switch specifies the graphics adapter of the VIADUKT in detail. Either the colour of the crosshair can be defined by a number value <cc> or a definition file <file name> sets comprehensive configurations.

Colour definition of the input frame and the crosshair with VIAUK /X**cc**. Following values are possible for 'cc':

0 = black	8 = dark grey
1 = blue	9 = light blue
2 = green	10 = light green
3 = cyan	11 = light cyan
4 = red	12 = light red
5 = magenta	13 = light magenta
6 = brown	14 = yellow
7 = light grey	15 = white

Default setting: light grey

Key words for the definition file:

- **GrDriver**
- **GrMode**

Possible values for **GrDriver** and **GrMode** (indented values) are:

```
3 = EGA  (> 64 kB)
      0 = 640 * 200 with 16 colours
      1 = 640 * 350 with 16 colours
      3 = 640 * 350 with  2 colours
4 = EGA  (64 kB)
      0 = 640 * 200 with 16 colours
      1 = 640 * 350 with  4 colours
      3 = 640 * 350 with  2 colours

9 = VGA
      0 = 640 * 200 with 16 colours
      1 = 640 * 350 with 16 colours
      2 = 640 * 480 with 16 colours
```

Example: EGA board with memory >64kB with a resolution of 640*350 pixel and 16 colours requires following definition:

- GrDriver = 3
- GrMode = 1

Further key words that are supported in the definition file are:

- **Xpixel**
- **Ypixel**

Number of screen pixel for the x and y coordinates. Xpixel and Ypixel must not correspond to the resolution of the defined graphics mode (GrDriver, GrMode).

Example:

- Xpixel = 320
- Ypixel = 256

- **ScrWidth**
- **ScrHeight**

The physical height and width of the screen can be specified for calculation of the height/width relation.

Example:

- ScrWidth = 960
- ScrHeight = 768

- **monochrome**

The bitplane used by the graphics adapter can be determined with this key word. Following values can be specified:

1 = blue bitplane
2 = green bitplane
3 = red bitplane

Example:

- monochrome = 2

Commentary lines that have to be introduced by a semicolon ";" can be inserted between the definitions.

Index

VIADUKT

Message Editor

(Not applicable to Viadukt Lite

Table of Contents

1. Survey

11/93

Message texts of any kind and length can be output to windows, the printer or logfiles with help of the message editor for the VIADUKT.

The messages can be error or state messages of a machine or a controlled process or instructions for the operator of a machine. These instructions can context sensitively relate to malfunctions or special machine states (for instance change of the drill head, tool change generally, etc.).

The messages are displayed at the VIADUKT screen or printed or stored into a logfile by supervision of the VIADUKT.
Any combinations of output devices above can be realized. A message can appear in a window of the VIADUKT screen and the total message or parts of it can simultaneously be stored into a logfile.

The messages are edited and tested with help of the message editor (VIATXTUK.EXE) that is available on the VIADUKT disk. VIADUKT version V1.30 (upwards) is required for use of the message texts.

Control of the error messages, that means their initialisation, display in windows, output to the printer, storage into logfiles or combinations of these, is realized by the VIADUKT and the PASE controller commonly.

The PASE controller defines in a registers the error messages that are demanded by the process and the VIADUKT displays, stores or prints the corresponding message.

On the other hand the PASE controller can query if the message, that is displayed in a window on the VIADUKT at the moment, is confirmed by the operator.

A multitude of additional functions that can be added to the command line call-up of the VIADUKT as optional parameters, provide further possibilities of process support.

The message texts that are displayed in a window, printed or stored into a logfile, are edited and tested in the message editor.

The message texts have to be tested because variables can be integrated into the message - for instance the state of an output. This presupposes certain logical capabilities of the complete message management and therefore requires the checking of the message text by a compilation run. VIADUKT uses this compiled object file for the taking of the messages. Three different object message files that differs by three different extensions, are created dependend on the selected output mode. The message is displayed in a window, printed or stored into a logfile in dependence upon the compiled output files which differ by their extensions. Also combinations of the output devices can be realized. Such a message file can contain any number of messages that are called-up by the PASE controller corresponding to the requirements of the process.
The message file has to be placed in that directory which is defined at the command line call-up of the VIADUKT (file name with path).

An editor screen appears after start-up of the message editor that provides windows, pull down menus and mouse support. The message texts are written using this editor with its variety of functions in the pull down menus.

After this, additional functions that exceeds the features of an usual editor, allow the check of the message by a test run. The message output which later is realized on the screen, send to a printer or stored onto the hard disk by the VIADUKT, is simulated after compilation. The error messages appear as if they would be displayed in a window later on.

The message editor may be leaved after compilation of the corresponding message file. The VIADUKT accesses to a defined PASE register that contains the number of the message for display which can be modified by the PASE controller. The message is displayed corresponding to this number until it is confirmed by the operator or the timeout time elapsed.

2. Starting the message editor

The message editor is started with the following DOS command line:

```
C:\..\VIATXTUK
```

The message editor screen appears after the line was confirmed with RETURN.

3. The editor screen

The graphical user interface of the message editor consists of following - elements:

- o the selection line for the single pull down menus (first line)
- o one or several windows in which texts can be edited, compiled, checked and corrected
- o the selection line of the most important menu functions (last line)

A window, several of it can be opened at once, provides following functions:

[■]	the window is closed if this symbol is activated with the left mouse button (position: top, left side)
Number	number of opened windows (position: top, right side; this number in connection with the ALT key activates the corresponding window)
C:\...	current file name with path (position: top, in the middle)
[⇳]	the previous window size can be restored with help of this symbol (position: top, right side)
*	this symbol indicates that the file was changed. The file was not modified if the symbol is absent, that means that the file in the window and the file on the hard disk are identical (position: bottom, left side).
Number:Number	the first number defines the line, the second number the column of the cursor position (position: bottom, left side).

Function ranges:

4 function ranges can be used in a window.

- the first line (double line) can be seized with the left mouse button - the mouse is moved while the left button remains pushed - and thereby the complete window can be moved.

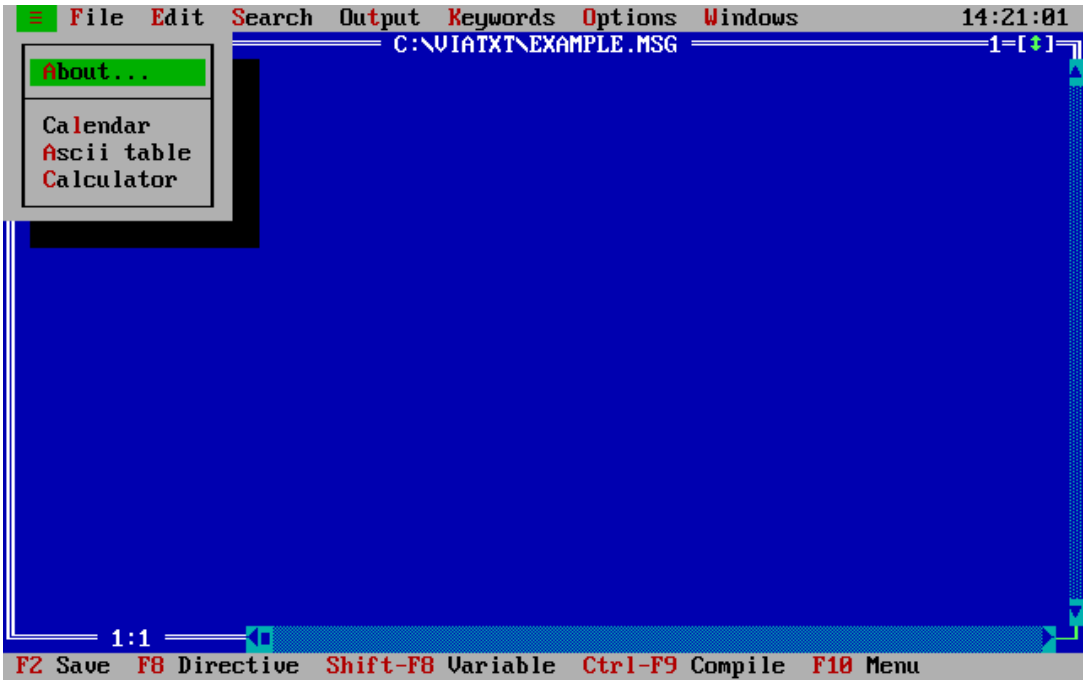
- the scroll bar at the right side scrolls the window up and down, the scroll bar at the bottom scrolls the window to the right respectively to the left.

- the window corner at the bottom on the right side (not separately marked) serves for the adjustment of the window size. For that purpose the mouse button has to remain pushed at the corner and the mouse has to be moved until the desired window size is reached. After size adjustment the window can be seized in the first line (double line) and moved completely.

4. The pull down menus

This section represents the pull down menus sequently. Numerous menu functions can be activated by hotkeys, the most important functions can be accessed in the bottom line directly. For detailed information about the functions see the examples at the end of this manual.

4.1 "≡" menu

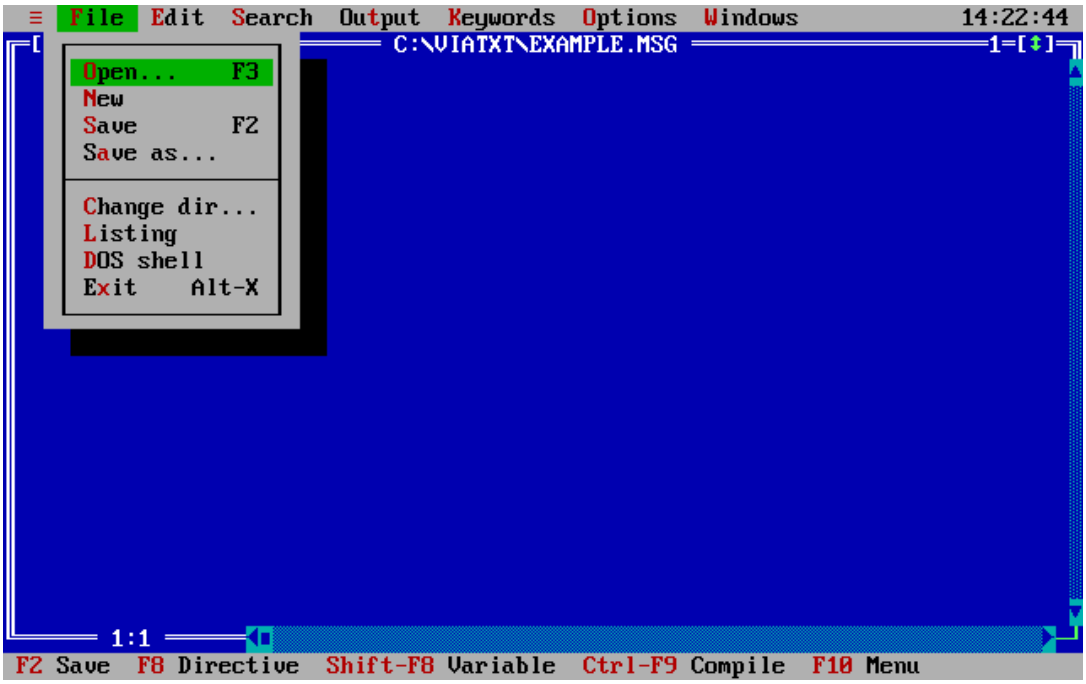


About	information window - e.g. version number
Calendar	century calendar

ASCII — the table lists the ASCII code of the character set (hexadecimal and decimal). A possible application may be the creation of text boxes for messages - for this purpose the necessary ASCII codes can be refered to in the table and inserted into the message text with the corresponding ALT key combination.

Calculator — pocket calculator

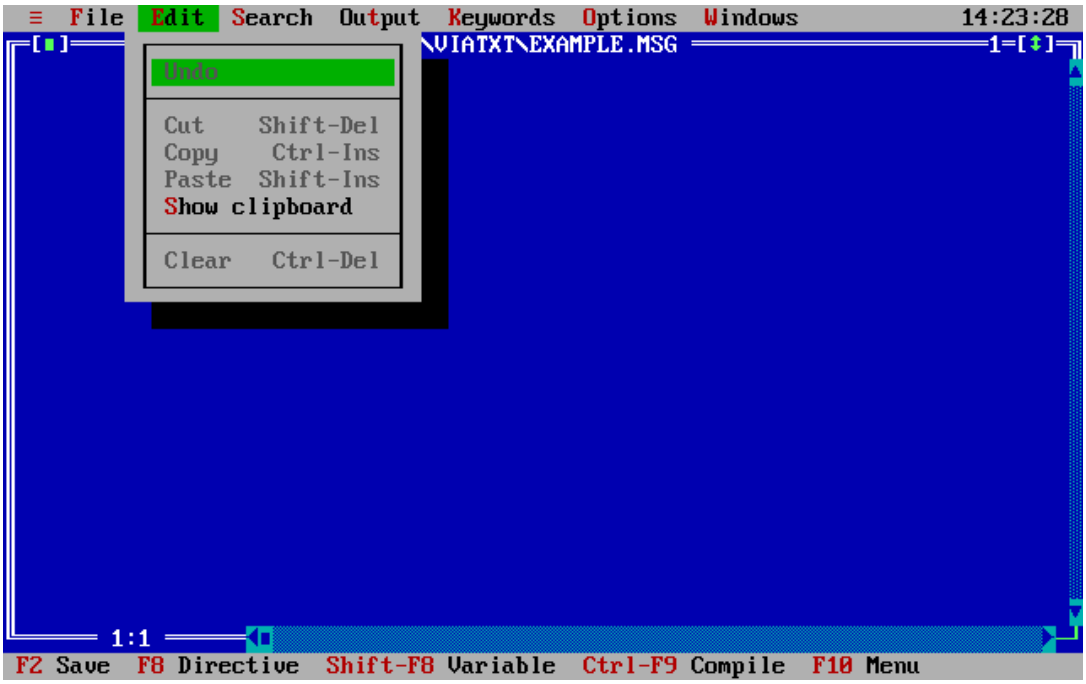
4.2 "File" menu



Open — the function loads a file into the editor. A window appears in which the file name can be specified after the selection line was activated via mouse click. Either the name can be defined or the file can be selected in the file list with the mouse. All source code files have the default extension ".MSG" (but any extension is possible). The compiled source code files that are simulated in a window, have the extension ".LST".

New	opens a new window that is named "Untitled". The file can be named with the "Save as" selection line and stored onto the hard disk.
Save	stores the file onto the hard disk using its current name. The name and the path of the file is displayed in a window line.
Save as	in contrast to "Save" this function assigns a new name (and path) and stores it onto the hard disk using the new name.
Change dir	changes the current directory. The integrated tree function supports the directory change.
Listing	prints the active window.
DOS shell	exit to DOS. Return with EXIT to the message editor after all commands are executed.
Exit	the message editor is terminated, the program returns to DOS.

4.3 "Edit" menu



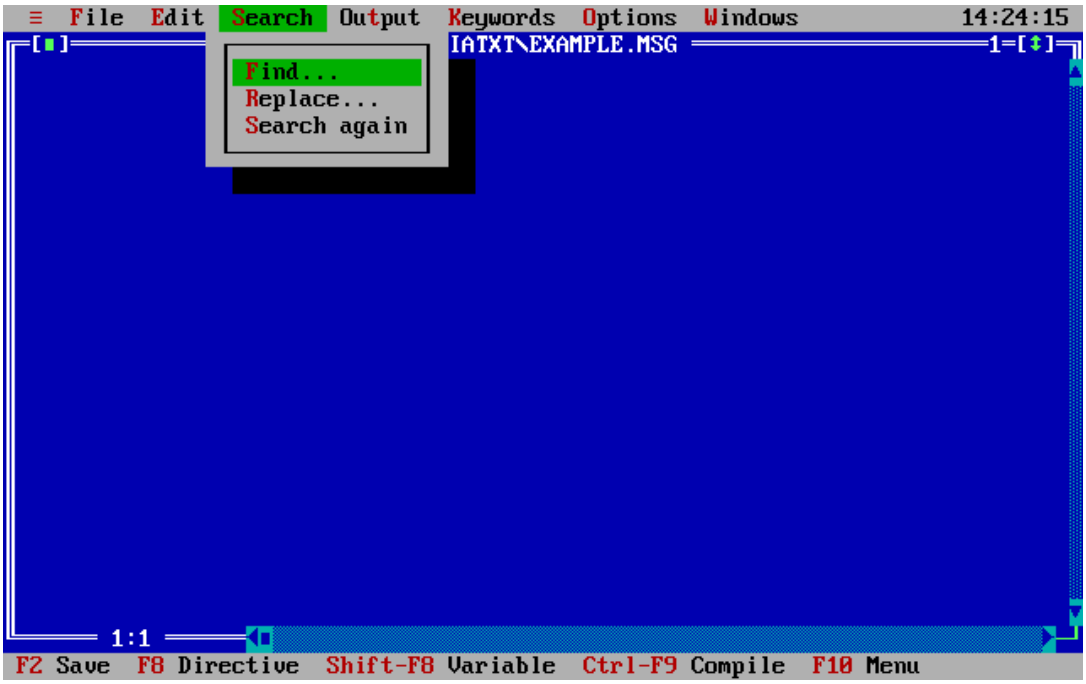
Undo — restores the last deleted line. The intermediate buffer stores only one line that can be restored after execution of some functions.

Cut — cuts out blocks from the text. First the text was marked with the pushed left mouse key or the combination of the SHIFT and the cursor keys. Then the marked block can be cut out from the text and transfered into the clipboard.

Copy — copies the marked text into the clipboard - in contrast to the cut function the text remains.

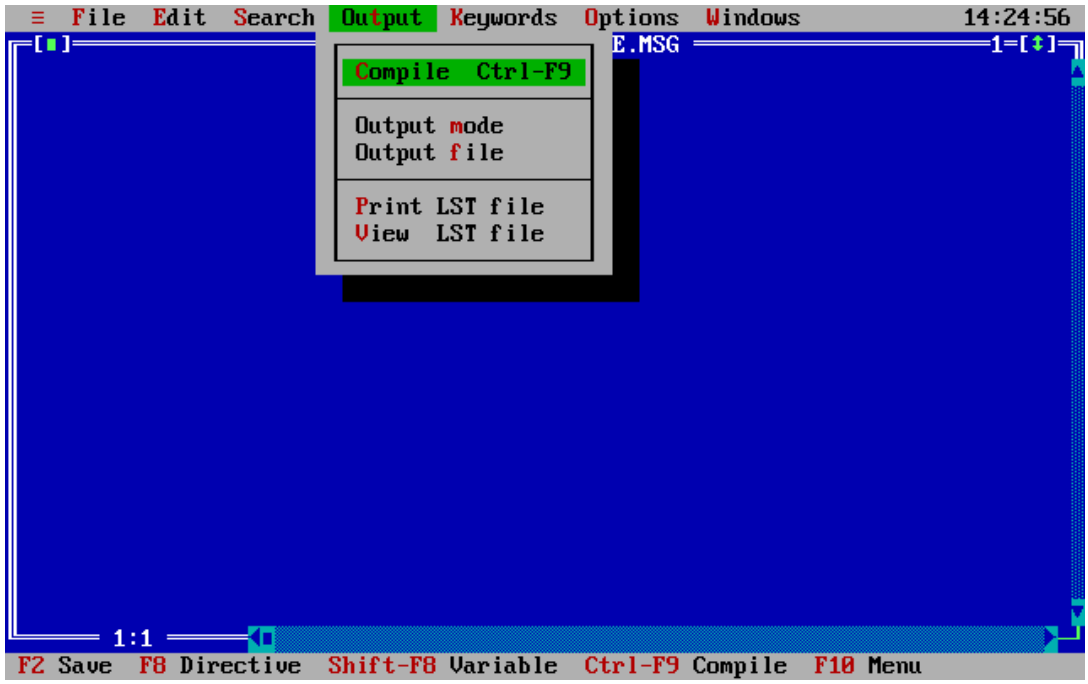
Paste	inserts the content of the clipboard at the current cursor position.
Show clipboard	shows the content of the clipboard.
Clear	erases the marked text.

4.4 "Find" menu



Find	the search term and some search criteria can be defined in a window. Besides a window can be opened which lists the previous terms.
Replace	a window appears in which the search term and the term for replacement can be defined.
Search again	the previous function ("Find" or "Replace") is repeated using the last specified definitions.

4.5 "Output" menu



Compile — starts the compilation of the message. Errors in the message text are indicated with red error bars. The compiler starts and represents the messages in simulated windows when the message text is correct. These windows with the extension ".LST" should be closed (e.g. with ALT-F3) after they are examined.

Output mode — the function defines at which output device the message appears - in a window, at the printer or in a logfile. Corresponding to this definition the compiler generates object code files with the extensions ".W", ".H" or ".L".

Furthermore following options can be set in the output mode window before compilation.

- o the representation of the index definitions in the ".LST" window can be determined.
- o the representation of the COLOR and DISPLAYTIME directives with their corresponding parameters during simulation run, additionally to the resulting message windows can be enabled.
- o the display of the LST files can be enabled or disabled.

The variables DATE and TIME consider the specific representation criteria if the country code is defined.
Following line has to appear in the CONFIG.SYS to be able to use this function:

```
INSTALL = C:\DOS\NLSFUNC.EXE C:\DOS\COUNTRY.SYS
```

Output file

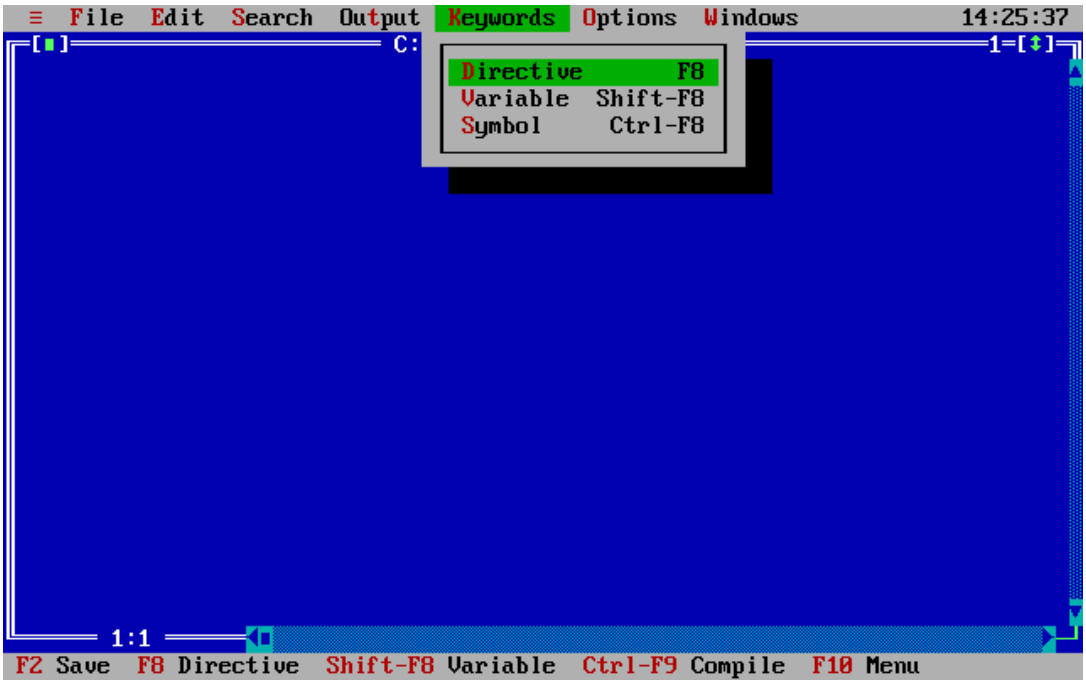
in this window name and path of the output file can be specified.
Essentially three files are used respectively are created that differ by their extensions:
- name.MSG: source code message file
- name.LST: test file for message window simulation
- name.W or .H or .L: object code **output file**; this file is used for representation of the error messages by the VIADUKT. The other files are working files.

Print LST file

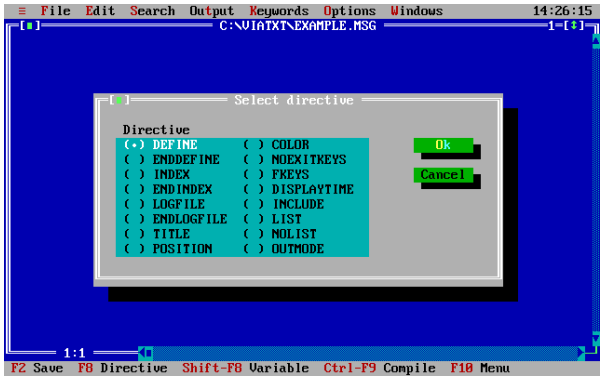
prints the files of the window simulation that are created during compilation.

View LST file displays the files for window simulation that are created during compilation.

4.6 "Keywords" menu



4.6.1 Directive following pop-up window appears after activation of the selection line:



DEFINE — the directive marks the begin of a definition block. Variables, file names, etc. can be specified in a definition block (for further information refer to chapter 8).

ENDDEFINE — this directive marks the end of a definition block.

INDEX — this directive stands at the beginning of each message text. A number succeeds the keyword that is the identification number for the later call-up of the corresponding message. This number is defined by the PASE controller in a register at runtime; the VIADUKT scans this register periodically and represents the corresponding messages at the different output devices (window, printer, logfile).

ENDINDEX — the text of a message ends at this directive.

LOGFILE — text that follows this directive is represented as well in a window as in a logfile. This directive makes possible both to limit the output into a logfile to the most important facts and display in a window comprehensive information.

ENDLOGFILE — the text that is written into the logfile ends at this directive.

TITLE — the string that follows the directive defines the title line of the message window.

POSITION — the directive is succeeded by 4 parameters with following meaning:

X: this parameter defines the x-coordinate of the upper left edge of the window (the window is placed in the middle of the screen if the parameter is 0).

Y: this parameter defines the y-coordinate of the upper left edge of the window (the window is placed in the middle of the screen if the parameter is 0).
W: defines the width of the message window (min. 15, max. 70).
H: defines the height of the window (2...12).

COLOR 4 parameters succeed the directive that have following meaning:
P1: foreground color
P2: background color
P3: headline foreground color
P4: headline background color

Following colors correspond to the parameter values:

0	black	1	blue	2	green
3	cyan	4	red	5	magenta
6	brown	7	lightgrey	8	darkgrey
9	lightblue	10	lightgreen	11	lightcyan
12	lightred	13	lightmag.	14	yellow
15	white				

NOEXITKEYS this function prevents that a message window can be cancelled with the keys RETURN or ESC. Thus the window has to be closed by the PASE controller or the timeout function.

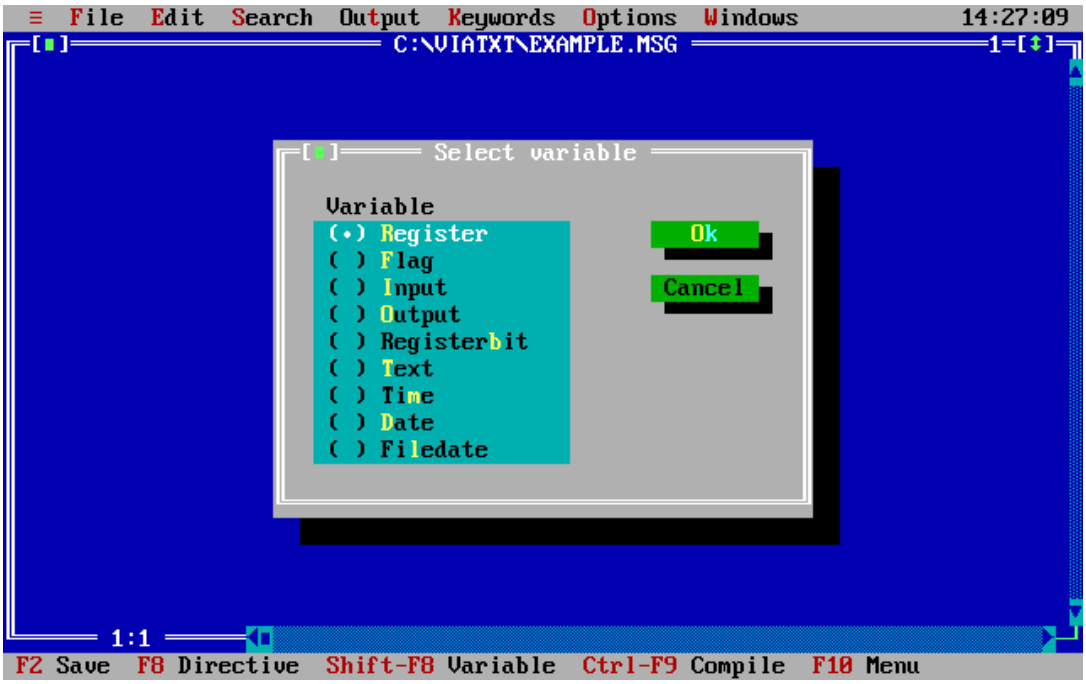
FKEYS the parameters with the syntax (Fx = flag no., [...]) assigne a flag number to the function keys of the VIADUKT, which can be activated during display of a message window.

	The assignment of the function keys to the flag numbers during the display of a message window is not equal to the assignment during operation without message windows. That means that during display of a message window function keys can have different functions defined by the flag assignment with the FKEYS directive.
DISPLAYTIME	the parameter of this directive specifies the timeout time of the message window. The message window is taken back from the VIADUKT screen after the time elapsed (parameter value in unit seconds). The message is lost in this case.
INCLUDE	the parameter of this directive defines the name of the file that is used as include file. The structure of an include file is equal to the structure of a message file that means that each message file can be used as include file.
LIST	a range can be marked with LIST and NOLIST if parts of the message file (.MSG) need not to be displayed at the window simulation run (.LST-file). The default setting is LIST. Therefore NOLIST has to mark the begin of the part for exclusion and LIST has to mark the begin of the part that is displayed again.
NOLIST	see directive LIST.
OUTMODE	this directive provides 3 parameters: o window o printer o logfile

The directive defines with the corresponding parameter, at the beginning of a message file where the messages appear, which are defined in this file - in a window, at the printer or in a logfile. This definition (placed at the beginning of the message file) substitutes the definition of the output mode in the "Output" pull down menu, "Output mode" selection line. The definition in the message file has priority.

4.6.2 Variable

following pop-up window appears after the activation of the selection line:



REGISTER

this system variable integrates contents of any registers into the message. A further pop-up window appears after activation of the selection line in which following parameters can be defined:

- o device: always number 0.
- o number: defines the register number.
- o offset: this parameter is added to the register number defined above, if it is addressed indirectly.
- o places before point: number of places before the decimal point.

o places behind point: number of places behind the decimal point.
o factor: this factor is multiplied with the register content before the register value appears in the message text.
o offset: this value is added to the register content, before the content appears as variable in the message text.

The both last parameters are used for unit conversion of the register contents.

FLAG

this system variable integrates states of any flags into the message. A further pop-up window appears after activation of the selection line in which following parameters can be defined:

o device: always number 0.
o number: defines the flag number.
o offset: this parameter is added to the flag number defined above, if it is addressed indirectly.
o fieldlength: this parameter defines the display width of the status texts mentioned below. Between three kinds of representation can be distinguished:

1. fieldlength > 0: flush left display, the difference between field length and number of read characters is filled with spaces.
2. fieldlength = 0: equal to point 1 but without filling with spaces.
3. fieldlength < 0: flush right

o text 0: this text appears in the message if the status of the flag is 0.
o text 1: this text appears in the message if the status of the flag is 1.

INPUT

this system variable integrates states of any inputs into the message. A further pop-up window appears after activation of the selection line in which following parameters can be defined:

- o device: always number 0.
- o number: defines the input number.
- o offset: this parameter is added to the input number defined above, if it is addressed indirectly.
- o fieldlength: this parameter defines the display width of the status texts mentioned below. Between three kinds of representation can be distinguished:
  1. **fieldlength > 0:** flush left display, the difference between field length and number of read characters is filled with spaces.
  2. **fieldlength = 0:** equal to point 1 but without filling with spaces.
  3. **fieldlength < 0:** flush right
- o text 0: this text appears in the message if the status of the input is 0.
- o text 1: this text appears in the message if the status of the input is 1.

OUTPUT

this system variable integrates states of any outputs into the message. A further pop-up window appears after activation of the selection line in which following parameters can be defined:

- o device: always number 0.
- o number: defines the output number.
- o offset: this parameter is added to the output number defined above, if it is addressed indirectly.

- o fieldlength: this parameter defines the display width of the status texts mentioned below. Between three kinds of representation can be distinguished:
  1. fieldlength > 0: flush left display, the difference between field length and number of read characters is filled with spaces.
  2. fieldlength = 0: equal to point 1 but without filling with spaces.
  3. fieldlength < 0: flush right
- o text 0: this text appears in the message if the status of the output is 0.
- o text 1: this text appears in the message if the status of the output is 1.

REGISTERBIT this system variable integrates states of any registerbits into the message. A further pop-up window appears after activation of the selection line in which following parameters can be defined:

- o device: always number 0.
- o number: defines the registerbit number.
- o offset: this parameter is added to the registerbit number defined above, if it is addressed indirectly.
- o fieldlength: this parameter defines the display width of the status texts mentioned below. Between three kinds of representation can be distinguished:
  1. fieldlength > 0: flush left display, the difference between field length and number of read characters is filled with spaces.
  2. fieldlength = 0: equal to point 1 but without filling with spaces.
  3. fieldlength < 0: flush right
- o text 0: this text appears in the message if the status of the registerbit is 0.

o text 1: this text appears in the message if the status of the registerbit is 1.

TEXT

this system variable integrates any text sequences into the message. The texts are stored as ASCII characters in registers, which succeed one another, and can be distinguished by definition of the first register:

o device: always number 0.

o number: here the register number is defined from which on the text is stored as a ASCII string.

o offset: this parameter is added to the register number defined above, if it is addressed indirectly.

o fieldlength: this parameter defines the display width of the status texts mentioned below. Between three kinds of representation can be distinguished:

1. fieldlength > 0: flush left display, the difference between field length and number of read characters is filled with spaces.
2. fieldlength = 0: equal to point 1 but without filling with spaces.
3. fieldlength < 0: flush right

TIME

this system variable displays the current time in the message.

DATE

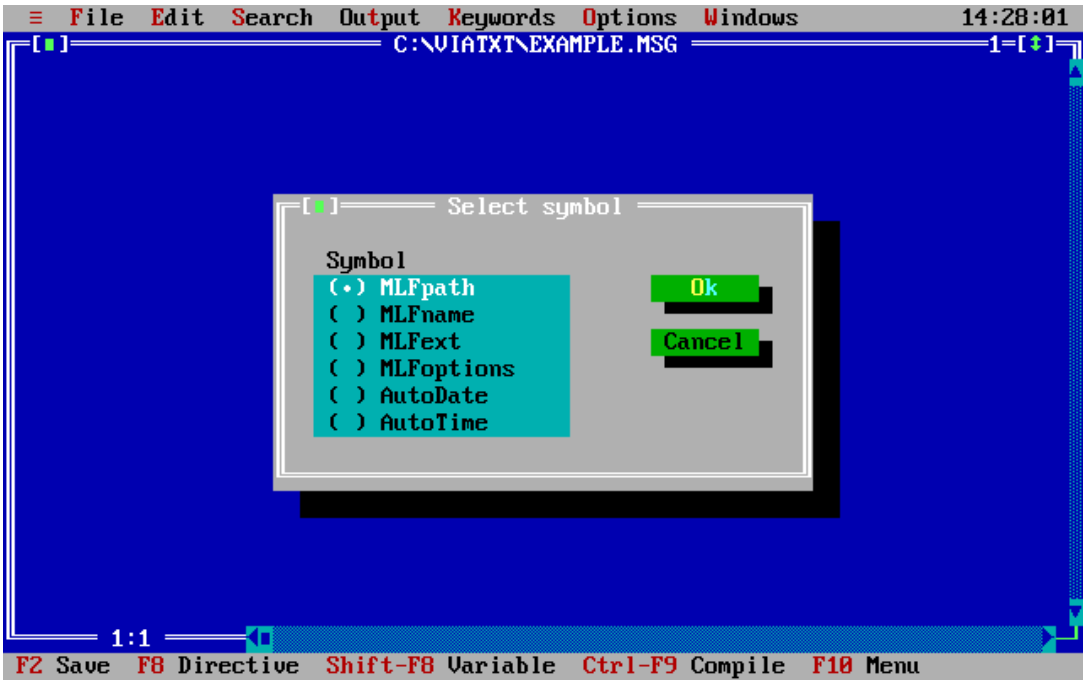
this system variable displays the current date in the message.

FILEDATE

this system variable creates file names which are composed by the current date (e.g. 921001 for October 1st, 1992). The variable can be used to automize the creation of names for logfiles.

4.6.3 SYMBOL

following pop-up window appears after the activation of the selection line:



MLFpath — this symbol defines the path of a logfile. The default setting is the current path. The definition of the path is possible in the define section of the message text.

MLFname — this symbol defines the name of a logfile. The default setting is the name which is composed by the current date (921001). A definition of the file name is possible in the define section of the message text.

MLFext — this symbol defines the extension of a logfile. The default setting is ".LOG". A definition of the file extension is possible in the define section of the message text.

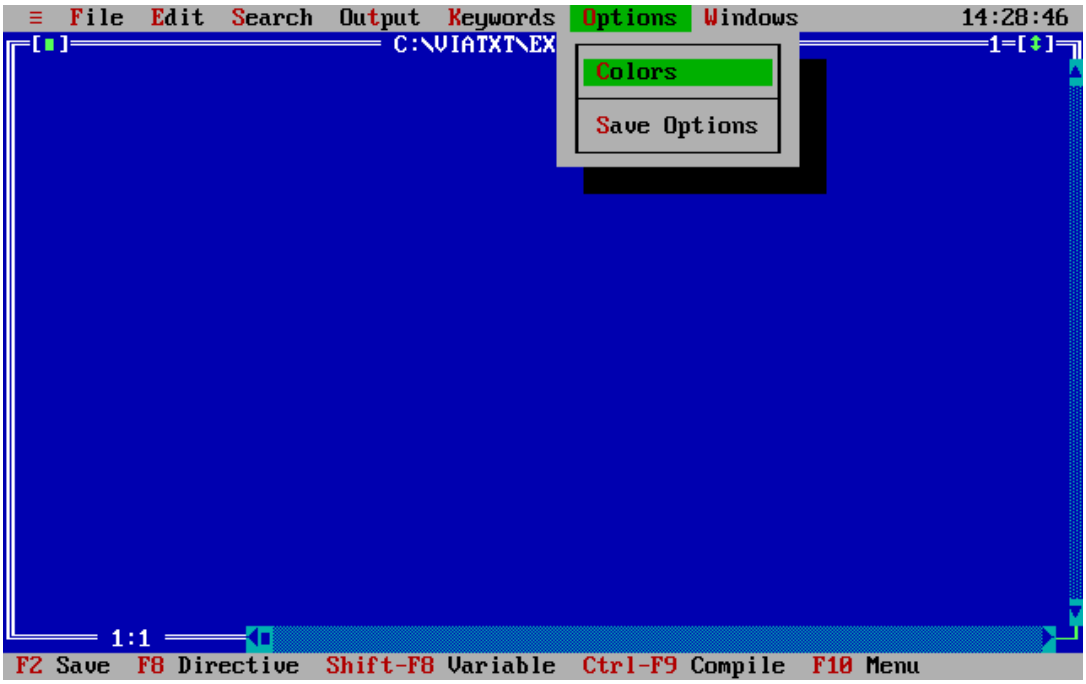
MLFoptions — options can be defined by this symbol in the define section of a message text, which are lead through additionally:

- /Rxxxx: number of the register, which content is interpreted as string and inserted between logfile name and extension (register content = 1234):
  `C:\directory\filename1234.ext`
- /O: the logfile is opened again, the old content is overwritten.
- /A: the message is annexed to the existing logfile (default setting).
- /E: the logfile is deleted.
- /P: the logfile is printed.
- /V [xxxx] the logfile is displayed in a file viewer. An additional register can be defined optionally, in which the start line number of the logfile listing is specified.

AUTODATE — this symbol can be switched on or off in the define section of the message text. The current date is set at the beginning of each message automatically, if the function is activated.

AUTOTIME — this symbol can be switched on or off in the define section of the message text. The current time is set at the beginning of each message automatically, if the function is activated.

4.7 "Options" menu



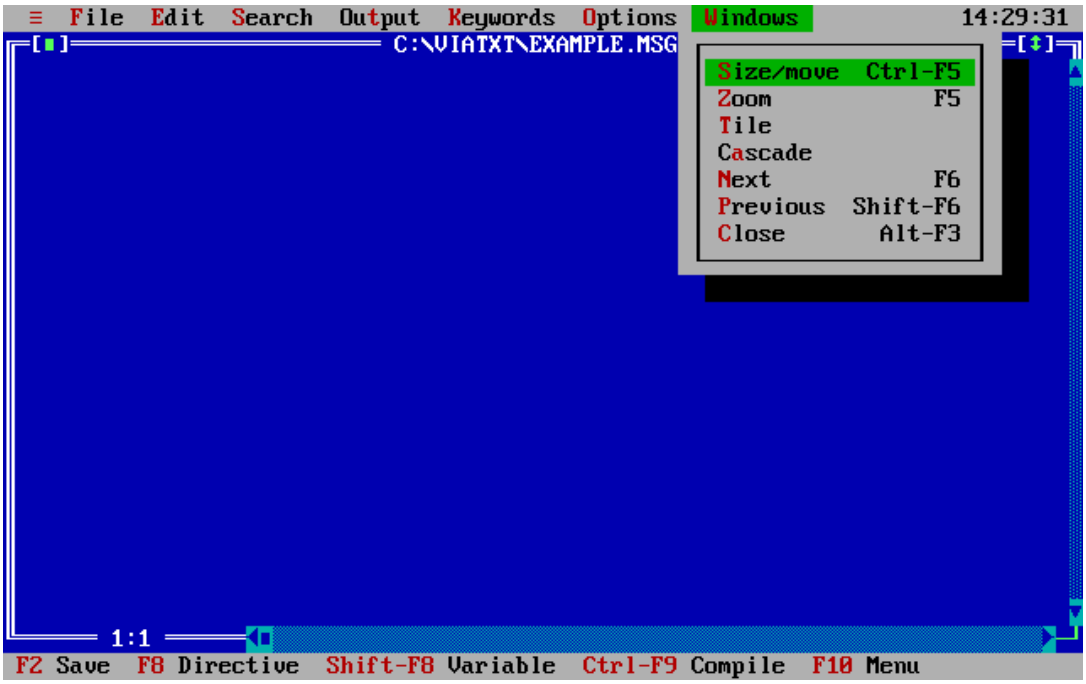
COLORS

the color settings for the message editor can be specified by this selection line. A window appears after selection in which 4 subwindows can be activated. The TAB key (or the mouse) activates the windows "Group", "Item", "Foreground", "Background" sequentially. The SHIFT-TAB key combination activates the windows in revers sequence. The mouse or the cursor keys can be used for movement within the windows.
Each point of the "Group" window relates to one or several subpoints in the "Item" window. The desired color can be assigned to each item by the "Foreground" and "Background" windows.The color setting is cancelled with the ESC key, confirmed with the RETURN key.

A test text is placed below the "Background" window, in which the effect of the color settings can be checked.

SAVE OPTIONS this function saves the color settings mentioned above.

4.8 "Window" menu



SIZE/MOVE — after confirmation of the selection line with the RETURN key, the position of the window can be moved with the cursor keys. The size of the window can be modified with the SHIFT cursor keys (see also chapter 3. The editor screen). The RETURN or the ESC key cancels the function.

ZOOM — the function toggles between the full size and the last setting.

TILE — the windows are displayed beneath each other in columns and rows if more windows are used.

CASCADE — the windows are cascaded.

NEXT	this function activates the next window. Also the mouse can be used for activation of the next window.
PREVIOUS	the function activates the previous window.
CLOSE	the function closes the active window.

5. Files, Extensions, etc.

The message editor uses respectively creates following files:

- **NAME.MSG**

  The message texts are typed into this file. It is the source file that is compiled into the simulation file NAME.LST and the corresponding object file for windows (NAME.W) for the printer (NAME.H) and for logfiles (NAME.L).

- **NAME.W**

  Object (output) file for a message window that is created with the "Compile" selection line in the "Output" menu.

- **NAME.H**

  Object (output) file to the printer that is created with the "Compile" selection line in the "Output" menu.

- **NAME.L**

  Object (output) file into a logfile that is created with the "Compile" selection line in the "Output" menu.

- **NAME.LST**

  The file simulates the output of the message in a window, after the compilation was started with the "Compile" selection line in the "Output" menu.

- **NAME.CFG**

  The message editor saves all color settings in this file. The file is created after the "Save Options" selection line was activated.

6. Switches (Start-up parameters)

6.1 Switches behind the LOGFILE directive (optionally)

/A: append: the messages are appended to the logfile and stored completely. The LOGFILE directive automatically creates path and file name (default setting).

/O: overwrite: erases the content of the existing logfile and stores the current message. Thus only the last message is stored in the logfile. The LOGFILE directive automatically creates path and file name.

/E: erase: the logfile is deleted.

/P: print: this option prints the logfile; no further functions can be executed by VIADUKT during the logfile is printed.

/Rxxxx: the content of the specified register (xxxx) is inserted between the logfile name and the extension.

/V [xxxx]: the logfile is displayed in a file viewer. An additional register can be specified optionally, in which the start line number for the file listing can be defined.

6.2 Switches at the VIAUK (VIADUKT) start-up for message control

VIAGR /**Wxxxx"Filename.W"**

The switch activates the message function. The message number is defined in the register with the number xxxx. The messages defined by this number are taken from the message file with the name "Filename.W" and displayed in a window. The complete path has to be specified if the message file and VIAUK are not placed in the same directory.

VIAGR /**Hxxxx"Filename.H"**

The switch activates the message function. The message number is defined in the register with the number xxxx. The messages defined by this number are taken from the message file with the name "Filename.H" and printed. The complete path has to be specified if the message file and VIAUK are not placed in the same directory.

VIAGR /**Lxxxx"Filename.L"**

The switch activates the message function. The message number is defined in the register with the number xxxx. The messages defined by this number are taken from the message file with the name "Filename.L" and stored into a logfile. The complete path has to be specified if the message file and VIAUK are not placed in the same directory.

7. Error messages

The compiler considers following errors during compilation of a message file, displays the error text in a red error bar and marks the error place with the cursor.

Error 1: "(" expected.

The compiler expects a opening bracket at this place.

Error 2: "," not found.

The compiler expects a comma at this place.

Error 3: ")" not found.

The compiler expects a closing bracket at this place.

Error 4: Integer constant expected.

The compiler expects a integer constant at this place.

Error 5: Real constant expected.

The compiler expects a real number at this place.

Error 6: String constant expected.

The compiler expects a text string at this place.

Error 7: Integer out of range.

The value range of the number marked by the cursor was exceeded.

Error 8: String constant out of range.

The text string marked by the cursor is too long.

Error 9: This directive is not allowed here.

Either #LIST and/or #NOLIST were defined between #INDEX and #ENDINDEX or #OUTMODE is not placed at the top of the message text.

Error 21: Unknown symbol.

The symbol marked by the cursor was not defined in the define section of the message text.

Error 22: Symbol already exists.

The symbol was already defined in the define section of the message text.

Error 30: #ENDDEFINE expected

The #DEFINE directive was not closed by a #ENDDEFINE directive.

Error 31: #ENDDEFINE without #DEFINE

The #ENDDEFINE directive was used without previous #DEFINE.

Error 32: "=" expected.

The compiler expects a "="-character at the place marked by the cursor.

Error 33: "=" not found.

Function key definition with #FKEYS without assignment by the "="-character.

Error 35: #ENDINDEX expected.

The #INDEX directive was not closed by a #ENDINDEX directive.

Error 36: #ENDINDEX without #INDEX.

The #ENDINDEX directive was used without previous #INDEX directive.

Error 40: #ENDLOGFILE expected.

The #LOGFILE directive was not closed by a #ENDLOGFILE directive.

Error 41: Filename expected.

#INCLUDE or #LOGFILE (only if used within an output file for logfiles) without specification of the file name.

Error 42: #ENDLOGFILE without #LOGFILE.

The #ENDLOGFILE directive was used without previous #LOGFILE directive.

Error 43: Invalid parameter.

An invalid parameter was used as logfile option.

Error 50: Too many #INCLUDE.

More than one include file level was used. Nesting of include files is not valid.

Error 51: Filename.Ext not found.

The compiler has not found the defined include file.

Error 52: F1...F26 expected.

General error function key definition with #FKEYS.

Error 53: Illegal output device.

An invalid output device was defined behind the #OUTMODE directive. Valid output devices are 'window', 'printer', and 'logfile'.

8. Examples

The following message examples explain the function of the several elements. The specific message functions which exceed the function range of a common editor are placed in the "Output" and "Keywords" menus.

8.1 First example

```
; File: vt_ex1.msg
;
; This file shows a simple first example of the VIADUKT Text window
; feature.
; As you may already know, lines starting with a semicolon are COMMENT
; lines and their contents are ignored.
; Every text message has to start with the #INDEX keyword. After the
; keyword, a number must be present. This number has to be used by the PASE
; controller to force the text message on the screen of the VIADUKT.
; Numbers may be in any order, but we suggest to sort them in an
; incremental order.
;
#INDEX 3
; This line starts the definition of the text. The text may be called
; by the PASE using the value 3.
Hello, I am the message number 3, I am 53 characters
in width and 8 lines high.

Messages like this may be used i.e. to indicate the
occurence of errors or warnings in the machine.

              PRESS <ENTER> TO CONTINUE.

#ENDINDEX
; The #ENDINDEX keyword is necessary to close the section of text for
; Index 3.


#INDEX 4
; This is another text:
- - - - - - - - - - WARNING - - - - - - - - - -

          Temperature alarm in oven 1.

           PRESS <ENTER> TO CONTINUE.
#ENDINDEX
```

```
; In order to play with this file, compile it using the CTRL-F9 key. This
; will create the file VT_EX1.W . A window will open with the
; title "...\VT_EX1.LST". The contents show you, how this text will
; appear in a popup window in the VIADUKT's graphic display. After you
; have examined this display, close the window with Alt-F3 or with the
; mouse (upper left button).
; Then start VIAUK with the command line
;  VIAUK /a /w111"VT_EX1.W"

; Using the PASE program, the monitor function in the VIADUKT or the
; SYMPAS setup mode, change register 111 to a value of 3. Then the first
; text will appear in a pop-up window. Press ENTER and watch the contents
; of register 5 at the SYMPAS setup screen. It changes to 0.
; Now change register 111 to a value of 4, the second text will appear.

; That is all a PASE program has to do to display predefined texts in
; a VIADUKT popup text window.

; Continue with VT_EX2.MSG
```

8.2 Second example: output to windows, the printer, into a logfile

```
; File: vt_ex2.msg
;
; This is the second example of the VIADUKT Text window feature.

#INDEX 4
; The message of index 4 is made more informative in the way, that
; the actual date and time are displayed. To insert the keywords
; @DATE and @TIME, type them or select them from the menu SHIFT-F8.
@DATE @TIME
- - - - - - - - - - WARNING - - - - - - - - - -

          Temperature alarm in oven 1.

           PRESS <ENTER> TO CONTINUE.

#ENDINDEX
;
; To use this file, compile it using the CTRL-F9 key. This
; will create the file VT_EX2.W . Then start VIAUK with the command line
;  VIAUK /a /w111"VT_EX2.W"

; Using the PASE program, the monitor function in the VIADUKT or the
; SYMPAS setup mode, change register 111 to a value of 4.
; The WARNING text will appear with the date and time at the top of
; the message.

; Continue with VT_EX3.MSG
```

This message file can now be compiled with the key combination CTRL-F9. The compiler creates a test file with the extension ".LST" and displays it on the screen. This file simulates the later message output in the VIADUKT. For that purpose the file is represented in a window. The simulation files should be closed after they were cheked, to avoid unnecessary accumulation of windows on the editor screen.
The output device can be defined with the "Output mode" selection line in the "Output" menu. The compiler creates a message file with the extensions ".W", ".H", ".L" corresponding to the output mode definition.

Following representation is created by compilation that is started with the "Compile" selection line in the "Output" menu:

```
/////////////// Index = 4 \\\\\\\\\\\\\\\
```

```
┌──────────────────────────────────────────────┐
│                 PASE message                 │
├──────────────────────────────────────────────┤
│ 03.11.1993 15:32:22                          │
│ - - - - - - - - - - WARNING - - - - - - - - - -│
│                                              │
│          Temperature alarm in oven 1.        │
│                                              │
│           PRESS <ENTER> TO CONTINUE.         │
│                                              │
├──────────────────────────────────────────────┤
│                  [   OK   ]                  │
└──────────────────────────────────────────────┘
```

The error number appears in the first line, the index line, that is the index of the error message. This index can be specified in a register which number is defined at the start-up of VIAUK. The number in the register defines which message is displayed in the window on the VIADUKT screen (see section 6.2).

The output device for the error message has to be determined at the start-up of VIADUKT on DOS level:

```
VIAGR /A /Wxxxx"Filename.W"
```

shows the message in a window. The message number defined by the number that follows the #INDEX directive in the message text is written into register xxxx by the controller. The VIADUKT scans periodically the content of register xxxx, and displays the corresponding messages in a window.

```
VIAGR /A /Hxxxx"Filename.H"
```

prints the messages.

The message number defined by the number that follows the #INDEX directive in the message text is written into register xxxx by the controller. The VIADUKT scans periodically the content of register xxxx, and prints the corresponding messages.

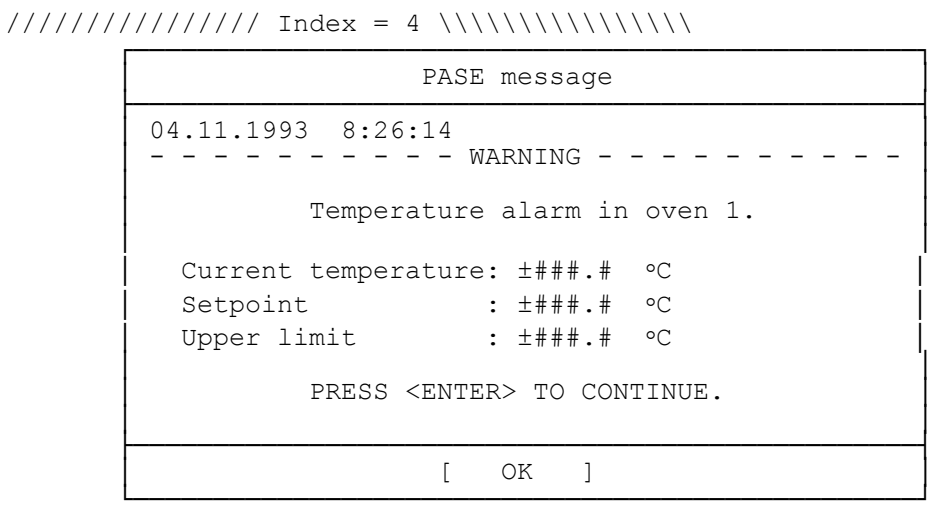
```
VIAGR /A /Lxxxx"Filename.L"
```

stores the messages into a logfile. The message number defined by the number that follows the #INDEX directive in the message text is written into register xxxx by the controller. The VIADUKT scans periodically the content of register xxxx, and stores the corresponding messages into a logfile.
The name of the logfile either results from the default settings (corresponding to the current date, for instance 921020 for October 20th, 1992) or can be defined by the user (see #LOGFILE directive).

8.3 Third example

```
; File: vt_ex3.msg
;
; This is the third example of the VIADUKT text window feature.

#INDEX 4
; The message of index 4 is made even more informative:
; Instead of simply telling a temperature problem, the actual
; temperature values shall be displayed. Assume the temperature
; setpoint is stored in register 230, the upper alarm limit in register
; 430 and the current temperature was stored by the PASE program in
; register 162. The register value 0 equals -40 °C and 1023 equals +160 °C.
; To show the temperature in degrees, a unit conversion is necessary.
;
; To call the keyword REGISTER you use the SHIFT-F8 key and select
; Register. Now enter all required values into the input fields. Step
; through the input fields with TAB and SHIFT-TAB (or with the mouse).
; After all entries are correct, press the ENTER key (or select OK with
; the mouse).
;
; To change an already written variable, position the cursor on any
; character of the variable definition and press SHIFT-F8. The
; definition window will open again, taking the current values of the
; definition as defaults.
;
; Now try to do this. Place the cursor on the @REGISTER text in the
; line where the current temperature is displayed and press SHIFT-F8.
@DATE @TIME
- - - - - - - - - - WARNING - - - - - - - - - -

          Temperature alarm in oven 1.

  Current temperature: @REGISTER(0,162,0,3,1,0.1955,-40)  °C
  Setpoint           : @REGISTER(0,230,0,3,1,0.1955,-40)  °C
  Upper limit        : @REGISTER(0,430,0,3,1,0.1955,-40)  °C

          PRESS <ENTER> TO CONTINUE.

#ENDINDEX
;
; To use this file, compile it using the CTRL-F9 key. This
; will create the file VT_EX3.W . Then start VIAUK with the command line
;  VIAUK /a /w111"VT_EX3.W"

; Continue with VT_EX4.MSG
```

Following message simulation appears on the screen after compilation (CTRL-F9):



The representation of the message can be checked and corrected in the message text if necessary. Before the simulation window (*.LST) should be closed (ALT-F3). A further compilation run shows the success of the changes.

8.4 Fourth example

```
; File: vt_ex4.msg
;
; This is the fourth example of the VIADUKT text window feature.

#INDEX 4
; Additionally to the current values, a help text is displayed, telling
; the operator what to do to solve the problem.
; Also the current state of the machine is displayed using the FLAG
; keyword. Any boolean variable (FLAG,INPUT,OUTPUT,REGISTERBIT) can
; display two different texts depending on the state of the boolean.
;
; In this example let's assume the AUTOMATIC/MANUAL flag is 10.
@DATE @TIME
- - - - - - - - - - WARNING - - - - - - - - - -

          Temperature alarm in oven 1.

  Current temperature: @REGISTER(0,162,0,3,1,0.1955,-40) °C
  Setpoint           : @REGISTER(0,230,0,3,1,0.1955,-40) °C
  Upper limit        : @REGISTER(0,430,0,3,1,0.1955,-40) °C

  Machine is in @FLAG(0,10,0,9,"MANUAL","AUTOMATIC")  mode.

  Solution: Open valve "cold water" using pushbutton "c.w." or
            a screw driver. Then open all windows and call
            the fire department.

          PRESS <ENTER> TO CONTINUE.

#ENDINDEX
;
; To use this file, compile it using the CTRL-F9 key. This
; will create the file VT_EX4.W . Then start VIAUK with the command line
;  VIAUK /a /w111"VT_EX4.W"

; You will find that the window now has a "scroll bar" at the right hand
; side. This scroll bar appears if more text is available as displayed on
; the current screen. Use the cursor keys to step through the text.

; Continue with VT_EX5.MSG
```

8.5 Fifth example: definition of symbols

```
; File: vt_ex5.msg
;
; This is example no. 5 of the VIADUKT text window feature.

; New in this example is the usage of symbol definitions. The user
; may define symbols for later usage. This is especially helpful for
; variable definitions if the same variable is used in more than one
; text window. See example below: The three register variables are now
; written between a #DEFINE and #ENDDEFINE keyword. To use the variables,
; now the assigned name (e.g. %temp1) has to be used.
; Important: a symbol has to start with the % character.
#DEFINE
 %temp1    = @REGISTER(0,162,0,3,1,0.1955,-40)
 %set1     = @REGISTER(0,230,0,3,1,0.1955,-40)
 %limit1   = @REGISTER(0,430,0,3,1,0.1955,-40)
#ENDDEFINE


#INDEX 4
@DATE @TIME
- - - - - - - - - - WARNING - - - - - - - - - -

          Temperature alarm in oven 1.

  Current temperature: %temp1  °C
  Setpoint           : %set1  °C
  Upper limit        : %limit1  °C

  Machine is in @FLAG(0,10,0,9,"MANUAL","AUTOMATIC")  mode.

  Solution: Open valve "cold water" using pushbutton "c.w." or
            a screw driver. Then open all windows and call
            the fire department.

          PRESS <ENTER> TO CONTINUE.

#ENDINDEX

#INDEX 5
@DATE @TIME
* * * * * * * * * * ALARM * * * * * * * * * * *

  The setpoint for oven 1 is %set1  °C with an
  upper limit of %limit1  °C. However the actual
  temperature is %temp1  °C. Therefore a general
  shut-down of the factory was initiated. Call
  Mr. boss on phone line 0815-4711 and then try
  to leave the room as fast as possible.

  DO NOT PRESS <ENTER>, INSTEAD LEAVE THE ROOM.

#ENDINDEX
```

```
; To use this file, compile it using the CTRL-F9 key. This
; will create the file VT_EX5.W . Then start VIAUK with the command line
;  VIAUK /a /w111"VT_EX5.W"

; Continue with VT_EX6.MSG
```

8.6 Sixth example: @AUTODATE, @AUTOTIME

```
; File: vt_ex6.msg
;
; This example introduces some of the predefined system symbols.
; They may be redefined using the normal DEFINE keyword.
;
; Instead of writing the @DATE and @TIME keywords down in every text index,
; a system symbol %AUTODATE and %AUTOTIME may be used. The default setting
; of these symbols ist OFF. If one or both of them are set ON, the
; corresponding value is written at the very beginning of each text window.
#DEFINE
 %temp1    = @REGISTER(0,162,0,3,1,0.1955,-40)
 %set1     = @REGISTER(0,230,0,3,1,0.1955,-40)
 %limit1   = @REGISTER(0,430,0,3,1,0.1955,-40)

 %AutoDate = ON
 %AutoTime = ON
#ENDDEFINE


#INDEX 4
; Please consider that this line is comment now : @DATE @TIME
- - - - - - - - - - WARNING - - - - - - - - - -

          Temperature alarm in oven 1.

  Current temperature: %temp1  °C
  Setpoint           : %set1  °C
  Upper limit        : %limit1  °C

  Machine is in @FLAG(0,10,0,9,"MANUAL","AUTOMATIC")  mode.

  Solution: Open valve "cold water" using pushbutton "c.w." or
            a screw driver. Then open all windows and call
            the fire department.

          PRESS <ENTER> TO CONTINUE.

#ENDINDEX

#INDEX 5
* * * * * * * * * * ALARM * * * * * * * * * * *

  The setpoint for oven 1 is %set1  °C with an
  upper limit of %limit1  °C. However the actual
  temperature is %temp1  °C. Therefore a general
  shut-down of the factory was initiated. Call
  Mr. boss on phone line 0815-4711 and then try
  to leave the room as fast as possible.

  DO NOT PRESS <ENTER>, INSTEAD LEAVE THE ROOM.

#ENDINDEX
```

```
; To use this file, compile it using the CTRL-F9 key. This
; will create the file VT_EX6.W . Then start VIAUK with the command line
;  VIAUK /a /w111"VT_EX6.W"

; Continue with VT_EX7.MSG
```

8.7 Seventh example

```
; File: vt_ex7.msg
;
; This example introduces the option to log information into a file on disk
;
; To do this, there are two possibilities:
;
;    Use the Logfile output mode of the VIATXTUK program and then the /L-
;    command line parameter of the VIAUK program. In this case, all
;    information that goes to a window with the /W option now goes into a
;    logfile.
;
;    The other possibility is to add #LOGFILE keywords into the text for
;    the window messages. In that case part of the text can be directed
;    to the popup window and other parts of the text can also be logged
;    into a file.
;
; To use the first way, change the OUTPUT MODE using the OUTPUT pull down
; menu MODE to LOGFILE. Then Compile this source file (which is the same
; source file as VT_EX6.MSG regarding the contents between INDEX and
; ENDINDEX).
;

#DEFINE
 %temp1    = @REGISTER(0,162,0,3,1,0.1955,-40)
 %set1     = @REGISTER(0,230,0,3,1,0.1955,-40)
 %limit1   = @REGISTER(0,430,0,3,1,0.1955,-40)

 %AutoDate = ON
 %AutoTime = ON
#ENDDEFINE


#INDEX 4
; Please recognize that this line is comment now : @DATE @TIME
- - - - - - - - - - WARNING - - - - - - - - - -

          Temperature alarm in oven 1.

  Current temperature: %temp1  °C
  Setpoint           : %set1  °C
  Upper limit        : %limit1  °C

  Machine is in @FLAG(0,10,0,9,"MANUAL","AUTOMATIC")  mode.

  Solution: Open valve "cold water" using pushbutton "c.w." or
            a screw driver. Then open all windows and call
            the fire department.

          PRESS <ENTER> TO CONTINUE.

#ENDINDEX
```

```
#INDEX 5
* * * * * * * * * * ALARM * * * * * * * * * * *

  The setpoint for oven 1 is %set1  °C with an
  upper limit of %limit1  °C. However the actual
  temperature is %temp1  °C. Therefore a general
  shut-down of the factory was initiated. Call
  Mr. boss on phone line 0815-4711 and then try
  to leave the room as fast as possible.

  DO NOT PRESS <ENTER>, INSTEAD LEAVE THE ROOM.

#ENDINDEX
;
; To use this file, set Outputmode to LOGFILE and compile using
; the CTRL-F9 key. This will create the file VT_EX7.L .
; Then start VIAUK with the command line
;  VIAUK /a /L111"VT_EX7.L"

; Put the value 4 into register 111 and a logfile will be created having
; a file name and an extension .LOG in the current directory. The content
; is the text of window 4. Every time you set register 111 to 4, this text
; will be APPENDED to the logfile (with the current date and time).


; Continue with VT_EX8.MSG
```

8.8 Eigth example: write data into a logfile

```
; File: vt_ex8.msg
;
; This example shows the second possibility to write into a logfile.
; The file is intended to be used as a window-text file (/W option), so
; the output mode should be WINDOW again.

#DEFINE
 %temp1    = @REGISTER(0,162,0,3,1,0.1955,-40)
 %set1     = @REGISTER(0,230,0,3,1,0.1955,-40)
 %limit1   = @REGISTER(0,430,0,3,1,0.1955,-40)

 %AutoDate = ON
 %AutoTime = ON
#ENDDEFINE


#INDEX 4
- - - - - - - - - - WARNING - - - - - - - - - -

#LOGFILE warnings.log
          Temperature alarm in oven 1.

  Current temperature: %temp1  °C
  Setpoint           : %set1  °C
  Upper limit        : %limit1  °C

  Machine is in @FLAG(0,10,0,9,"MANUAL","AUTOMATIC")  mode.
#ENDLOGFILE

  Solution: Open valve "cold water" using pushbutton "c.w." or
            a screw driver. Then open all windows and call
            the fire department.

          PRESS <ENTER> TO CONTINUE.

#ENDINDEX

#INDEX 5
* * * * * * * * * * ALARM * * * * * * * * * * *

  The setpoint for oven 1 is %set1  °C with an
  upper limit of %limit1  °C. However the actual
  temperature is %temp1  °C. Therefore a general
  shut-down of the factory was initiated. Call
  Mr. boss on phone line 0815-4711 and then try
  to leave the room as fast as possible.

  DO NOT PRESS <ENTER>, INSTEAD LEAVE THE ROOM.

#ENDINDEX
```

```
; To use this file, set Outputmode to WINDOW and compile using
; the CTRL-F9 key. This creates the file VT_EX8.W .
; Then start VIAUK with the command line
;  VIAUK /a /W111"VT_EX8.W"

; Put the value 4 into register 111 and a window will appear that contains
; all the information of index 4. Additionally a logfile "WARNINGS.LOG"
; will be created. Only the text between LOGFILE and ENDLOGFILE will be
; written to this file.
; Every time you set register 111 to 4, this text
; will be APPENDED to the logfile (with the current date and time).


; Continue with VT_EX9.MSG
```

8.9 Ninth example

```
; File: vt_ex9.msg
;
; If output to the same logfile appears in several text definitions, it
; is convenient to define a user variable holding the file name: see entry
; %file1 in the DEFINE section below.

#DEFINE
 %temp1    = @REGISTER(0,162,0,3,1,0.1955,-40)
 %set1     = @REGISTER(0,230,0,3,1,0.1955,-40)
 %limit1   = @REGISTER(0,430,0,3,1,0.1955,-40)

 %AutoDate = ON
 %AutoTime = ON

 %file1    = "warnings.log"
#ENDDEFINE


#INDEX 4
- - - - - - - - - - WARNING - - - - - - - - - -

#LOGFILE %file1
          Temperature alarm in oven 1.

  Current temperature: %temp1  °C
  Setpoint           : %set1  °C
  Upper limit        : %limit1  °C

  Machine is in @FLAG(0,10,0,9,"MANUAL","AUTOMATIC")  mode.
#ENDLOGFILE

  Solution: Open valve "cold water" using pushbutton "c.w." or
            a screw driver. Then open all windows and call
            the fire department.

          PRESS <ENTER> TO CONTINUE.

#ENDINDEX

#INDEX 5
* * * * * * * * * * ALARM * * * * * * * * * * *

  The setpoint for oven 1 is %set1  °C with an
  upper limit of %limit1  °C. However the actual
  temperature is %temp1  °C. Therefore a general
  shut-down of the factory was initiated. Call
  Mr. boss on phone line 0815-4711 and then try
  to leave the room as fast as possible.

  DO NOT PRESS <ENTER>, INSTEAD LEAVE THE ROOM.

#ENDINDEX
```

```
; To use this file, set output mode to WINDOW and compile using
; the CTRL-F9 key. This will create the file VT_EX9.W .
; Then start VIAUK with the command line
;  VIAUK /a /W111"VT_EX9.W"

; Put the value 4 into register 111 and a window will appear that contains
; all the information of index 4.
; Every time you set register 111 to 4, the text between LOGFILE and
; ENDLOGFILE will be APPENDED to the logfile "warnings.log"

; Continue with VT_EX10.MSG
```

8.10 Tenth example: logfile options

```
; File: vt_ex10.msg
;
; This file shows the options that are available for the LOGFILE keyword.
;  INDEX 5 shows how to overwrite a logfile,
;  INDEX 6 shows how to erase a file,
;  INDEX 7 shows how to use a non-fixed file name.
; %file1 in the DEFINE section below.

#DEFINE
 %temp1    = @REGISTER(0,162,0,3,1,0.1955,-40)
 %set1     = @REGISTER(0,230,0,3,1,0.1955,-40)
 %limit1   = @REGISTER(0,430,0,3,1,0.1955,-40)

 %AutoDate = ON
 %AutoTime = ON

 %file1    = "warnings.log"

#ENDDEFINE



; INDEX 4 is the same as in example 9.
#INDEX 4
- - - - - - - - - - WARNING - - - - - - - - - -

#LOGFILE %file1
          Temperature alarm in oven 1.

  Current temperature: %temp1 °C
  Setpoint           : %set1 °C
  Upper limit        : %limit1 °C

  Machine is in @FLAG(0,10,0,9,"MANUAL","AUTOMATIC")  mode.
#ENDLOGFILE

  Solution: Open valve "cold water" using pushbutton "c.w." or
            a screw driver. Then open all windows and call
            the fire department.

          PRESS <ENTER> TO CONTINUE.

#ENDINDEX

; INDEX 5 shows how to overwrite a file
;         Whenever Index 5 is called, the logfile is first erased and
;         then the information is written to the file. So if Index 5 is
;         called repeatedly, only the information of the last call is
;         contained in the file.
;
;         If Index 5 is called, the file is erased first before information
```

```
;         is written to the file. Afterwards Index 4 may
;         be called to append information to the same file.
;
;         To activate OVERWRITE use the /o option in the LOGFILE line.
#INDEX 5
- - - - - - - - - - WARNING - - - - - - - - - -

#LOGFILE %file1 /o
          Temperature alarm in oven 1.

  Current temperature: %temp1 °C
  Setpoint           : %set1 °C
  Upper limit        : %limit1 °C

  Machine is in @FLAG(0,10,0,9,"MANUAL","AUTOMATIC")  mode.
#ENDLOGFILE

  Solution: Open valve "cold water" using pushbutton "c.w." or
            a screw driver. Then open all windows and call
            the fire department.

          PRESS <ENTER> TO CONTINUE.

#ENDINDEX


; INDEX 6 shows how to erase a file
; To erase a file, the option /e may be used. No information can be
; written to the file of course.
#INDEX 6
- - - - - - DELETE LOGFILE - - - - - - - - - -

#LOGFILE %file1 /e
#ENDLOGFILE

          Information in Logfile %file1
          has been deleted.

          PRESS <ENTER> TO CONTINUE.

#ENDINDEX

; INDEX 7 shows how to use a non-fixed file name.
;
;         Information may be output to a file with a runtime dependant
;         file name using the /r option. After the /r option a register
;         number must be specified. If access to a logfile is requested,
;         VIAUK looks into the specified register to determine the
;         file name that has to be used. The contents of the register is
;         appended to the file name specified in the LOGFILE line.
;
;         Example: Register 190 shall contain a value between 1 and 21
;                  (usage: 3 shifts a day, 7 days per week --> 21 log-
;                  files).
;                  The PASE program takes care of register 190 contents.
;                  Index 7 texts will be directed to a file with one of
;                  the following names:
;                        SHIFT1.LOG   up to   SHIFT21.LOG
```

```
#INDEX 7
- - - - - - - - - - WARNING - - - - - - - - - -

#LOGFILE %file1 /r190
          Temperature alarm in oven 1.

  Current temperature: %temp1 °C
  Setpoint           : %set1 °C
  Upper limit        : %limit1 °C

  Machine is in @FLAG(0,10,0,9,"MANUAL","AUTOMATIC")  mode.
#ENDLOGFILE

  Solution: Open valve "cold water" using pushbutton "c.w." or
            a screw driver. Then open all windows and call
            the fire department.

          PRESS <ENTER> TO CONTINUE.

#ENDINDEX


;     The LOGFILE options may be used also together, i.e. the line
;
;     LOGFILE "A.LOG" /r190 /o
;
;     will read register 190 (contents is 12 for example), erase
;     the file A12.LOG if it exists and output new information to
;     that file.
;
;
; To use this file, set Outputmode to WINDOW and compile using
; the CTRL-F9 key. This will create the file VT_EX10.W .
; Then start VIAUK with the command line
;  VIAUK /a /W111"VT_EX10.W"


; Continue with VT_EX11.MSG
```

8.11 Eleventh example: system symbols for logfile definition

```
; File: vt_ex11.msg
;
; Four system symbols are explained in this file.
;   MLFpath   MainLogFile Path
;                The default value for this symbol is set to nothing.
;                When used, the current directory is used.
;                If redefined, the defined value is used. Take care
;                to use valid path names.
;                Example:
;                   %MLFpath = "c:\projekt1\logfiles\"
;
;                Do not forget the \ at the end.
;
;
;   MLFname   MainLogFilename
;                The default value for this symbol is the systemvariable
;                @FILEDATE. If nothing is changed, logfiles have a name
;                that is created out of the current date (september 16th
;                1992 results in "920916").
;                If redefined, any other DOS file name may be specified.
;                Example:
;                   %MLFname = "alarms"
;
;   MLFext    MainLogFile extension
;                ".LOG" is the default value. If you want another
;                extension for the Mainlogfile then redefine this symbol.
;                Example:
;                   %MLFext = ".DAT"
;
;   MLFoptions MainLogFile options
;                The options described in example 10 may be specified
;                for the Mainlogfile. Default is /a (append mode).
;                Example:
;                   %MLFoptions = "/r190 /o"
;
;
;  If the symbols are not defined by the user, their corresponding default
;  value takes effect. The symbols are used to control the logfile name
;  in message files for Logfiles (/L commandlineoption), and whenever
;  there is NO file name specified after the LOGFILE keyword.

;  The first index in this file (INDEX 4) writes to the default logfile
;  (having the current date as file name and LOG as extension), the second
;  index will use a different file because of the redefinition of the
;  MLFxxx symbols.


#DEFINE
 %temp1    = @REGISTER(0,162,0,3,1,0.1955,-40)
 %set1     = @REGISTER(0,230,0,3,1,0.1955,-40)
 %limit1   = @REGISTER(0,430,0,3,1,0.1955,-40)
 %AutoDate = ON
```

```
 %AutoTime = ON

#ENDDEFINE



#INDEX 4
- - - - - - - - - - WARNING - - - - - - - - - -

#LOGFILE
          Temperature alarm in oven 1.

  Current temperature: %temp1 °C
  Setpoint           : %set1 °C
  Upper limit        : %limit1 °C

  Machine is in @FLAG(0,10,0,9,"MANUAL","AUTOMATIC")  mode.
#ENDLOGFILE

  Solution: Open valve "cold water" using pushbutton "c.w." or
            a screw driver. Then open all windows and call
            the fire department.

          PRESS <ENTER> TO CONTINUE.

#ENDINDEX

#DEFINE
   %MLFpath = "c:\test\"
   %MLFname = "example"
   %MLFext = ".dat"
#ENDDEFINE



#INDEX 5
- - - - - - - - - - WARNING - - - - - - - - - -

#LOGFILE
          Temperature alarm in oven 1.

  Current temperature: %temp1 °C
  Setpoint           : %set1 °C
  Upper limit        : %limit1 °C

  Machine is in @FLAG(0,10,0,9,"MANUAL","AUTOMATIC")  mode.
#ENDLOGFILE

  Solution: Open valve "cold water" using pushbutton "c.w." or
            a screw driver. Then open all windows and call
            the fire department.

          PRESS <ENTER> TO CONTINUE.

#ENDINDEX
```

```
; To use this file, set Outputmode to WINDOW and compile using
; the CTRL-F9 key. This will create the file VT_EX11.W .
; Then start VIAUK with the command line
;  VIAUK /a /W111"VT_EX11.W"


; Continue with VT_EX11.MSG
```

8.12 Twelfth example: output to the printer, text boxes

```
; File: vt_ex12.msg
;
; The command-line option /H of the VIAUK program allows definition
; of a control register for printer messages.
; Message files for printer messages are created in almost the same way
; as message files for pop up windows. The only differences are:
;   o    use the OutputMode PRINTER in the OutputMode menu.
;   o    a file with the extension .H will be created
;   o    you can use as many characters on a line as your printer supports.
; This example shows a shift protocol for printout at the end of a shift.
; The graphical characters can be created using the ALT key. The numbers
; to be put in may be retrieved from an ASCII table which is accessible
; through the ≡ pull down menu.
; In this example, it is very useful to have the printout preview in the
; VT_EX12.LST window. Here you can see how the text will be printed on
; the printer without wasting many pages of paper until all lines and
; characters are where you want them to be.
; DO NOT forget to CLOSE the .LST window before you compile again.
; Otherwise more and more .LST windows will open and you will be unable to
; tell which is the latest one. You can have a look at all windows using
; the WINDOW pull down menu and the CASCADE item.

#INDEX 4

┌──────────────────────────────────────────────────────────────────────┐
│ @DATE  @TIME   PRODUCTION PROTOCOL                                   │
├──────────────────────────────────────────────────────────────────────┤
│                                                                      │
│ Machine number  . . . . . . . : @REGISTER(0,180,0,1,0,1.0,0.0)       │
│                                                                      │
│ Number of parts processed . . : @REGISTER(0,2020,0,6,0,1.0,0.0)      │
│                                                                      │
│ Number of defect parts  . . . : @REGISTER(0,2021,0,6,0,1.0,0.0)      │
│                                                                      │
│                                                                      │
│ Tolerance summary:                                                   │
│                                                                      │
│ Minimum value length "a"  . . : @REGISTER(0,2058,0,3,2,0.01,0.0)     │
│                                                                      │
│ Maximum value length "a"  . . : @REGISTER(0,2059,0,3,2,0.01,0.0)     │
│                                                                      │
│ Average value length "a"  . . : @REGISTER(0,2060,0,3,2,0.01,0.0)     │
│                                                                      │
│                                                                      │
│ Adjustment of tools is @FLAG(0,101,0,3,"not","") necessary.          │
│                                                                      │
│                                                                      │
└──────────────────────────────────────────────────────────────────────┘

#ENDINDEX
```

The last example demonstrates the creation of text boxes. Generate the corresponding characters with the ALT key combinations and create in this manner the desired text boxes.