LIMITED LIABILITY COMPANY
RESEARCH AND PRODUCTION ENTERPRISE «EKRA»

Waves

User’s manual
ЭКРА.00090-01 90 01

Version 4.0 (April 09, 2024)

Waves application is designed for viewing and analyzing oscillograms (disturbance records) taken by digital oscilloscopes and relay protection devices.

The following formats of oscillograms are supported:

Format	Mask	Notes
BE2502/ED5, BE2704/ED7	**.dfr*	One oscillogram can consist of several files (up to 16)
BE2702, BE2702M	*DR???F?.???*	One oscillogram can consist of several files (up to 4)
BE2701	*DATA??*.???*	One oscillogram can consist of several files (up to 4)
EKRA	**.zfr*	Can contain oscillograms of BE2702, BE2702M, BE2704/ED7 and BE2502/ED5
Comtrade	**.cfg + *.dat* **.cff*	All existing standard versions are supported.
Waves	**.waves*	Waves application own format, in which any oscillogram can be saved after its analysis with all introduced changes

Waves is distributed either as an installation package or as a portable archive.
A PC under *Microsoft Windows XP* or later or *Linux* with a *Wine* package is needed to work with the *Waves*.

Getting ready to work with Waves on PC running *Linux* is described in section 17.

CONTENTS

1 Terms and Definitions

1.1 Signals

An oscillogram consists of a set of signals recorded or calculated in a common time interval. *Analog* and *binary* signals are supported.

No data - special sample value, which is not displayed on graph, is supported for all signal types. Such sample value can be acquired in case of errors during its calculation. This value also fills the spaces between the fragments of oscillograms when they are combined.

1.2 Complex signals

In addition to analog signals, Waves supports the so-called analog signals, presented in a complex form. Such signals represent an array (by number of samples in the oscillogram), each element of which is a complex number, calculated during the previous period of a definite frequency. The file's main frequency period (50 Hz as a rule) is used by default, but there is also an option to set the frequency manually.

> For simplicity, further on these signals will be called *complex signals*, when there will be something that has to be specially mentioned regarding this type of signals.

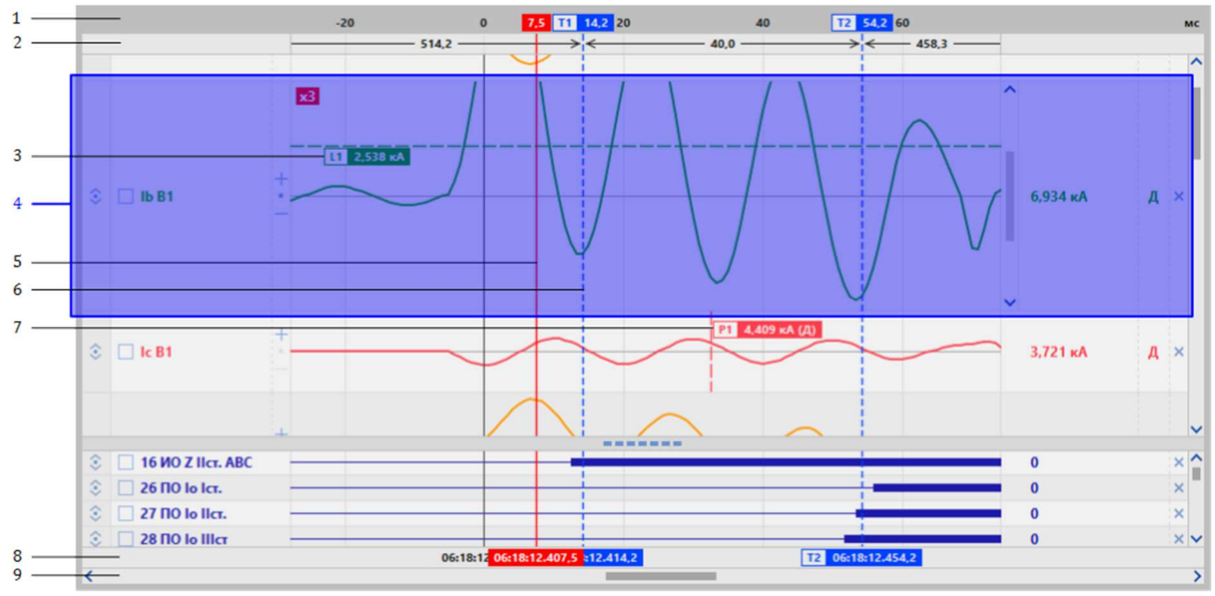
1.3 Three-phase circuits

Analog and complex signals can be combined in three-phase circuits, which can be used for combined (group) scaling of signals by amplitude, as well as in a number of calculations to simplify the setting of parameters. The table of three-phase circuits is presented in the *Three-phase circuits* window, which opens by the *Signals > Three-phase circuits* command. This table shows the composition of three-phase circuits for the opened oscillogram, and allows to create new circuits manually. If the circuit name is displayed on a pink background, it means that it is defined incorrectly and the signals included in it should be analyzed for the identity of their units and transformation ratios. Incorrectly defined circuits are not showed in the circuit selection lists of various tools.

1.4 Actual sample

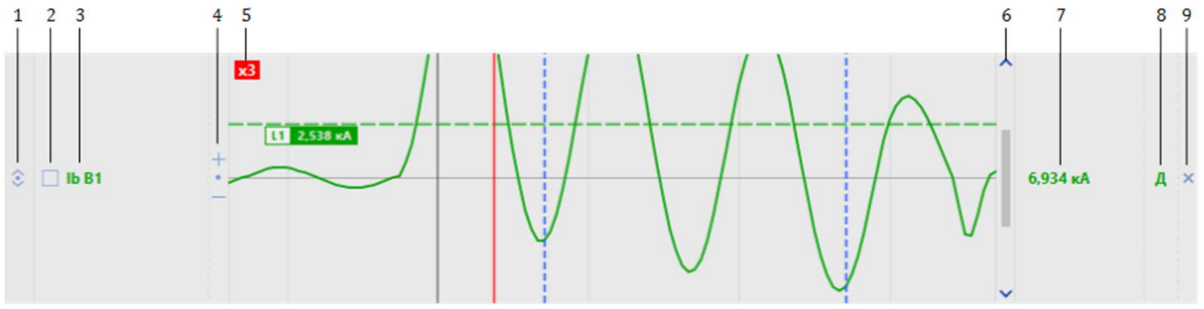
In the *Waves* there is a concept called *actual sample*. It is set by the *cursor* (vertical red line), which is always present in the signal area and which can be dragged along the time axis. Timestamp of the actual sample is displayed in the header of the cursor. The actual sample is used to display signal values in their relevant area, as well as in various analysis tools.

1.5 Elements of oscillogram panel



No.	Element
1	Relative time scale (relative to the start moment)
2	Scale of intervals between adjacent markers
3	*Level* tool
4	Strip
5	Cursor
6	*Marker* tool
7	*Probe* tool
8	Absolute time scale
9	Time scrollbar

1.6 Strip elements



No.	Element
1	Grip zone to move the strip to a new location
2	Signal marking checkbox (available only in special modes)
3	Signal name
4	Strip scaling button
5	Strip scale ratio
6	Strip scroll bar (present only if the strip scale ratio is set)
7	Signal value
8	Mark of signal value type
9	Button to hide the strip (all signals on it)

2 PROGRAM SETTING

2.1 USER INTERFACE SETUP

In order to access the settings you must select the *Settings* command from the main menu of the program.

2.1.1 PAGE: DEFAULT VIEW

On this page you can configure the state of various elements of the oscillogram toolbar, in which they will appear when opening each new oscillogram.

> Except for the files in *Waves* format, which save all the actual state of the interface within themselves

2.1.2 PAGE: MEASUREMENT UNIT SCALE

The page contains a table, where you can set the measurement unit scale, that will display in the *manual* scaling mode of measurement units. This can be made for values, that can be scaled (for example, current can be displayed in A, kA, mA, etc.). The scale will be used by default for calculated signals, and in some cases for signals, downloaded from files.

2.2 ASSOCIATE FILES

Select the *Associate files* command from the main menu of the program to get access to the window where extensions of oscillogram files are associated to the *Waves*.

Associating of extensions allows to open oscillograms of relevant formats by double-click on the file name in the *Windows Explorer*.

Installation version registers all associations during the installation process by itself, but for a portable version, association in program is the only option possible.

> This function will also be useful if, for some reason, data on associations was damaged or rebound to other software in *Windows*, and you do not need that. This function helps to restore all lost associations.

3 Opening of oscillogram

Waves allows opening several oscillograms simultaneously. Each oscillogram is opened in its own tab. To open an oscillogram, execute the *Open* command from the main menu, which opens *Browse files and folders* window. Oscilloscope start time, information about recording device and its location are displayed here for each oscillogram along with its file name.

A quick analysis of each file is done in order to get this information. If the analysis detects the incorrectness, file name is displayed in red without additional information.

> If folder belongs to network or contains many oscillograms, there may be small delay in displaying the file list on the screen.

3.1 Features of oscillograms in EKRA format

3.1.1 Fragmentation

Oscillograms in EKRA formats can consist of several files (*fragments*). In the *Browse files and folders* window, files of one oscillogram are visually grouped.

> Only files with the original name format can be combined into one oscillogram. Therefore, we highly recommend not to rename the files in EKRA formats.

3.1.2 Sampling rate

EKRA devices convert input analog signals into digital form with the rate of 2400 Hz *minimum* and all their calculation algorithms work with such data. During oscillogram recording the data is downscales to the 1200 Hz. As a result, the analysis of EKRA devices behavior based on oscillograms, is not always precise. The discrepancies are more distinct in the transient modes in algorithms with derivatives.

3.2 Comtrade format features

3.2.1 Symbols encoding

Text configuration files (*.cfg) or corresponding compound files' (*.cff) sections of *Comtrade* format, recorded by Russian oscillography devices, as a rule use single-byte ANSI characters. In this case, Cyrillic characters can be recorded in DOS (code page 866) or Windows encodings (code page 1251). In addition, 2013 standard introduces the international UTF8 encoding.

Waves automatically determines file encoding. The determination algorithm is based on probabilistic methods and, on rare occasions, may be inaccurate. In case of wrong encoding determination, open the file and manually change its format via the command of *File > Change encoding*. This operation does not lead to reopening of an oscillogram and does not affect the signals names added after the file opening.

3.2.2 PRIMARY AND SECONDARY VALUES OF ANALOG SIGNALS

Some *Comtrade* files may have no information concerning what values (primary or secondary) are used for recording of analog signal samples. When such file is uploaded, *Waves* will pop-up a window to manually enter this information.

3.2.3 THREE-PHASE CIRCUITS

Comtrade standard doesn’t have an obvious option to combine signals into three-phase circuits; however, when uploading such files, *Waves* analyzes the names and parameters of signals, and sometimes can perform such combination.

4 WORKING WITH OSCILLOGRAM

4.1 PAGE TAB OF OSCILLOGRAM

Page tab shows a file name of an opened oscillogram. When you hover the mouse cursor on a tab, a tip with complete information on the oscillogram is showed. To close an oscillogram, click the cross on the right side of the tag.

Also, by grabbing the tab with the mouse, you can move the oscillogram from one *Waves* window to another, which allows viewing several oscillograms at the same time.

4.2 CURSOR MOVEMENTS

– *Dragging by mouse*. A yellow loop stretches behind the cursor, which indicates the time offset relative to the previous position;

– *Left-click* in the display area of the oscillogram graph moves the cursor to a sample corresponding to the place of the click;

– The *Left* and *Right* keys move the cursor left and right by one sample;

– The button *Move cursor to trigger timestamp* on the toolbar moves the cursor to the moment of trigger;

– The button *Move cursor to specific timestamp* on the toolbar moves the cursor to a moment by specifying its value manually.

4.3 TIME SCROLL

– Keyboard:

Key	Action
Shift + Left	Smooth scrolling to the left
Shift + Right	Smooth scrolling to the right
Shift + Page Up	Page scrolling to the left
Shift + Page Down	Page scrolling to the right
Shift + Home	Scrolling to the beginning of oscillogram
Shift + End	Scrolling to the end of oscillogram

– Use of time scrollbar;

– *Shift + Mouse scroll*;

4.4 Time scaling

– Keyboard:

Key	Action
Ctrl + Left	Scaling up
Ctrl + Right	Scaling down

– The *Zoom in time* and *Zoom out time* buttons of the toolbar;

– *Ctrl + Shift + Mouse scroll*;

4.5 Scrolling of signals list

– Keyboard:

Key	Action
Up	Smooth scrolling up
Down	Smooth scrolling down
Page Up	Smooth page scrolling up
Page Down	Smooth page scrolling down
Home	Scrolling to the beginning of list
End	Scrolling to the end of the list

– Using the vertical scrollbar of the list;

– *Mouse scroll*;

4.6 Signal zoom by amplitude

By default, if opened oscillogram doesn't imply the combination of signals into three-phase circuits, all signals are scaled independently.

Three-phase circuits allow combining of up to three signals (phase or line). This in particular gives an opportunity for joint scaling of signals combined into circuit, even if individual scaling is applied. Nevertheless, the main purpose of the three-phase circuits is simplifying the choice of signals in different tools and calculations, while their scaling function is a handy side-effect.

In many cases, more than three signals have to be combined. *Waves* contains *Scaling groups* tool for this task. Each group can include any signals and three-phase circuits. Everything that is in the scaling group is scaled together and regardless of everything that is not included in it.

Select *Signals > Scaling groups* command to edit scaling groups.

Therefore, graph of analog signal inside the strip can be displayed in one of four modes of zoom by amplitude.

Modes are changed in the drop down *Analog signals zoom mode* list on the toolbar.

Mode	Description
Individually	Each graph is scaled individually regardless of other graphs and takes up the entire strip height
By unit	Signal graphs with the same measuring unit are scaled together (regardless the strip they are currently on)
By group (by units)	Signal graphs of one three-phase circuit or/and one scaling group are scaled together, and all the others, not included neither in a three-phase circuit nor in a scaling group, are scaled by measuring units
By group (individually)	Signal graphs of one three-phase circuit or/and one scaling group are scaled together, and all the others, not included neither in a three-phase circuit nor/not in a scaling group, are scaled individually

Hidden signals are not involved in joint scaling. Therefore, when hiding or showing signals in joint scaling modes, the amplitude range of other visible signals can change.

4.7 Strip zoom value

If an analog signal has high dynamic range, areas with small values are indistinct between areas with high values. In order to examine area with small values, it is possible to change the strip zoom value in range 1 ... 100.

If there is at least one analog signal on the strip, the area for changing strip zoom value becomes available. By default, zoom value equals 1. Graphs on this strip do not exceed its height and are entirely visible.

Button + increases zoom by 1. Button − decreases it by 1. Button ● sets zoom to 1. Fast change of zoom value with the step of 1 is possible by scrolling the mouse when the cursor is in the area of zoom value change.

You can set value manually via the *Change strip's zoom value* command from strip context menu. The zoom value can be input in fractional numbers.

If the zoom value becomes greater than 1, the graphs of signals go beyond the vertical boundaries of the line. At the same time, a vertical scroll bar appears on the right of the graphs, which allows viewing any graph section on enlarged scale.

4.8 Strip height change

It is possible to change strip height if there is at least one analog signal. You can change height of one strip or all strips simultaneously.

In order to change one strip height you should drag the bottom of strip and move it up or down.

You can change the height of all strips simultaneously in the following ways:

– Keyboard:

Key	Action
Ctrl + Down	Increase strip height
Ctrl + Up	Decrease strip height

– *Zoom in by amplitude* and *Zoom out by amplitude* buttons of the toolbar

– *Ctrl + Mouse scroll*;

4.9 SIGNAL MOVING

To move a signal, grab its name with mouse and move it to a new location. While moving, the red horizontal line shows the insertion point. It is possible to arrange signals to separate strips, and combine them in one strip.

To move the entire strip, you need to grab the grab area (symbol on the left side) of the strip with mouse and move it to a new location.

In addition to moving signals and strips within the oscillogram panel, you can drag them to some analysis tools (phasor diagram, spectrum, values table), which is equivalent to selecting signals in the tool settings.

Perform the *Ungroup strip* command from the context menu to quickly disunite the signals from one strip to several individual strips.

4.10 SIGNAL INVERSION

Waves allows to invert signals. The name of the inverted signal is preceded by a "~" (tilde). Repeated inverting returns the samples back to the original state and removes the tilde. Inversion is executed by the *Invert* command of the signal context menu or in the *Properties* window.

4.11 SIGNAL VISIBILITY

Signals that are not currently needed on the screen, can be hidden.

– *Hide strip* command of the strip context menu hides all the signals on it at once;

– *Hide* command of the signal context menu hides the corresponding signal;

– *Signals > Visibility > Change* command marks the signals that has to be visible;

– *Signals > Visibility > Hide non changing* command hides all signals that are unchanged within the oscillogram;

– *Signals > Visibility > Hide zero* command hides all signals that are unchanged within the oscillogram with sample value equal to zero;

– *Signals > Visibility > Hide noise* command hides all signals marked as noise;

– *Signals > Visibility > Show Hidden* command makes all hidden signals visible.

4.12 Signal deleting

It is possible to completely delete information about the signal from the oscillogram. A signal can be deleted only if no other signal depends on it. If some signals cannot be removed, but they interfere with you, hide them.

To delete a signal, execute the *Delete* command from signal context menu.

4.13 Signal properties

Setting the basic properties of the signal is made on the *Properties* page of the *Signal Properties* window, which opened by the *Properties* command in the signal context menu or by double-clicking the signal name with the left mouse button.

> For calculated signals, part of the basic properties can be calculated automatically. Therefore, there are checkboxes in front of their values in the window, cleared by default, which indicates that the property was filled out automatically. Nevertheless, you can define the values of these properties yourself, for this you need to check the corresponding box and set the required value manually.

4.13.1 Main signal features

Feature	Description
Name	Signal name
Type[1)]	Signal type **Analog | Binary | Complex**
Prim[2)]	Primary component of the transformation ratio and its measurement unit
Second[2)]	Secondary component of the transformation ratio and its measurement unit
Inversion	*Invert samples* checkbox performs the signal sample's inversion
Circuit[1,2)]	Name of three-phase circuit to which signal belongs
Color	Signal color **Blue | Yellow | Green | Red | Purple | Turquoise | Brown | Grey | Dark**
Line[2)]	Style of line, representing the signal **Solid | Dashed**

1) reading only
2) absent in binary signals

4.13.2 Additional properties of analog signals

For signals not only an instantaneous value is possible, but any other variants, which can be selected from the drop-down list *Analog signals displayed value* on the oscillogram toolbar. However, by using the *Advanced properties > Show only instantaneous value* checkbox, you can enable a mode that always displays the instantaneous value.

> If the oscillogram format supports the ability to specify the display mode of only the instantaneous value for analog signals, then this checkbox can be set for a particular signal immediately after loading automatically.

For frequency signals, the ability to shift the axis by a specified level is available. By default, the offset is set to the main frequency of the oscillogram (50 or 60 Hz).

4.14 Signal calculation

Waves provides ample opportunities to create calculated signals.

Each calculated signal is associated with complete information on how and on which signals it was calculated. So you can change the calculation parameters at any moment and recalculate the signal. Moreover, if there are signals in the oscillogram calculated based on the changed signal, they also will be automatically recalculated.

Calculations can be made in both primary and secondary values.

> We **DO NOT** recommend choosing calculation in secondary values unless you are firmly convinced that this is what your calculations require. As a rule, calculation in secondary quantities is required for calculating specific formulas that are specially written for secondary quantities and take into account all the differences in the transformation ratios of the signals used in the calculation. For example, when calculating FUSEF, the setting of which is specified only for secondary values.

4.14.1 Signal calculation based on expressions

All calculations in *Waves* are based on expressions. Using the *Calculate > Calculate expression* command you can create a signal by directly entering the needed expressions from the keyboard. This is the most flexible and functional way of calculating new signals, however, it requires knowledge of the rules for writing expressions (see Section 13), and is designed for an experienced user.

4.14.2 Signal calculation using menu

Waves provides a set of the most frequently used calculations in the form of the menu commands. Each such command can create one or several signals, the expressions for calculation of them are generated automatically according the selected settings.

All available commands are collected in a tree, which is located in the *Calculate* section of the menu. Commands are divided into logic groups and have pop-up hints.

Three lowest groups with names of *Transform <signal type> signals* type, contain the commands which work only with one input signal of the corresponding group type. The content of these groups is equal to the content of the *Calculate* group of the context menu of the corresponding signal types.

4.14.3 Editing of calculated expression

In the *Properties* window of the calculated signal there is a button *Edit expression*, click it to call the *Expression editor* window and add changes to the expression used for signal calculation.

4.14.4 Dependencies

In the *Properties* window on the *Dependencies* page there is a signal list, that the current signal depends on (if it is calculated), and also a list of signals which use this signal for own calculations.

4.15 Display of analog signal values

4.15.1 Primary and secondary values

Analog signals can be displayed both in primary and secondary values. The display mode is switched using the *Show primary /secondary values* (P/S) switch on the tool bar.

4.15.2 Analog signals values

The *Analog signals* d*isplayed value* drop-down list on the tool bar allows selecting the value in which the analog signals will be displayed. For convenience, an explanatory label of the selected value will appear next to the value: E is the effective value; M is the mean value; Hn is the effective value of the nth harmonic.

> An effective (instantaneous) value of actual sample is always displayed for analog signals with activated *Show only instantaneous value* mode.

For complex signals which frequency basis differs from the default basis (the first harmonics of oscillogram main frequency), it is displayed in an explanatory label as 2H/25Hz (here the basis is the second harmonics of main frequency 25 Hz)

4.15.3 Signal measurement units

Analog signals have primary and secondary measurement units. Samples of such signals are always stored in the basic measurement unit. Thus, if the signal is voltage, then samples are stored in volts.

Waves has two modes of displaying values, which are switched by selecting one of the sub-items in the *Signals* > *Measurement units* group.

MODE: *AUTOMATIC*

This mode is set as default. When displaying signal values, *Waves* scales the measurement unit automatically so that there are no very large or very small numbers on the screen. For example:

> **110,000 V** is displayed as **110 kV**
> **0.0123 A** is displayed as **12.3 mA**

MODE: *MANUAL*

The measurement unit display is set manually. That is, if the signal is voltage and stores its readings in volts, you can force the program to always display it, for example, in kilovolts.

When switching to this mode, a triangle appears in the signal value field on the oscillogram allowing to select the desired scale for display.

For signals loaded from oscillograms, the scale of measurement units is set in the oscillogram file. For calculated signals, it is assigned when they are created according to the *Measurement units* table of the *General settings* window.

> The Comtrade format allows specifying the scale of the measurement unit only for those values (primary or secondary), in which the file is recorded. The unit scale of the alternative value is assigned according to the *Measurement units* table of the *General settings* window. For example, if the file is recorded in primary values, then the scale of the secondary unit will be taken from the table.

4.16 VISIBLE RANGE RESTRICTION

Oscillogram visible range can be restricted by time. Push the button *Range* on the toolbar and click the graph field in two time points to select the oscillogram section, that has to remain visible. Everything to the left and right from it will be hidden.

Push the button *Reset range* to show the full time range again.

5 OSCILLOGRAM TOOLS

5.1 MARKERS

Marker is a blue vertical dotted line that serves to mark an important moment. The marker can be given explanatory text that will be displayed along the line in the vertical direction.

Each marker has a personal identifier (T1, T2 ...), which can be used for reference in documents.

5.1.1 CREATING

– The *Markers > Create (Shift + ~)* command adds a new marker to the current position of the (red) cursor;

– *Shift + Left click* adds a new marker to the position with a timestamp corresponding to the point of click.

5.1.2 MOVING

Grasp the marker with mouse to move it along the time axis. When moving the marker, a loop stretches behind it, showing the time shift relative to the previous position.

If you capture a marker while holding down the *Shift* key and start moving, the marker itself remains in place, while the new marker moves with its loop, showing the offset relative to the original marker. This function is convenient if you want to add a new marker that is offset from the existing one for a certain period.

5.1.3 SEARCHING

To search for a marker, you need to open the *Markers* menu and double-click on the marker of interest, considering its identifier and timestamp.

5.1.4 EDITING

– Command *Markers > Edit*. Select the marker from the list, considering its identifier and timestamp;

– *Edit* command of context menu of the marker;

– *Double click* on the required marker.

5.1.5 REMOVING

– Command *Markers > Remove*. Select the marker from the list, considering its identifier and timestamp;

– Command *Markers > Remove all*. Deletes all markers added to the oscillogram at once;

– *Remove* command of the marker context menu.

5.1.6 Time intervals

– When two and more markers are added to oscillogram, a scale with time intervals between markers becomes visible.

– *Tools > Intervals table* command manages its visibility on the panel screen, where the time intervals between all the markers added to the oscillogram are shown as a table.

5.2 Probes

Probes is a tool that allows to mark a specific timestamp on a particular signal to display its value at that point of time. The Probe color matches the color of the signal on which it is set. For analog signals, you can select the value in the probe parameters to be displayed as the probe value. The *Auto* option, set by default, defines what value selected from the *Analog signals displayed value* list will be shown on the toolbar.

Each probe has a personal identifier (P1, P2 ...), which can be used for reference in documents.

When the oscillogram is highly compressed in time, the display of probe becomes meaningless, and they automatically cease to be displayed so as not to clutter up the view.

5.2.1 Creating

– *Probes > Create* command allows adding probes to the signals with checked boxes at the cursor position or one of the markers on the oscillogram;

– *Ctrl + left-click* adds a new probe to the signal closest to the click point, at a timestamp corresponding to the click point.

5.2.2 Moving

Grasp the probe with mouse to move it along the time axis.

5.2.3 Searching

To search for a probe, you need to open the *Probe* menu and double-click on the preferred measurement, considering its identifier and timestamp.

5.2.4 Editing

– Command *Probes > Edit*. In this case, the required measurement must be selected from the list that appears. In the list, measurement is represented by its identifier;

– Command *Edit* of the probe context menu;

– *Double click* on the desired probe.

5.2.5 Removing

– Command *Probes > Remove*. In this case, the required probe must be selected from the list that appears. In the list, probe is represented by its identifier;

– Command *Probes > Remove all*. Deletes all probes added to the oscillogram at once;

– Command *Remove* of the probe context menu.

5.3 Levels

Level is a horizontal dashed line that can be moved vertically and which shows the amplitude of the signal at a particular point. The level is bound to a specific signal and has its color.

Each level has a personal identifier (L1, L2 ...), which can be used for reference in documents.

5.3.1 Creating

– *Levels > Create* command adds new levels to the signals with checked boxes;

– *Alt + left click* adds a new level to the signal closest to the point of the click.

5.3.2 Movement

The level can be moved vertically on the signal amplitude, changing the value it shows.

5.3.3 Searching

To search for a level, open the *Levels* menu and double-click on the preferred level, considering its identifier and timestamp.

5.3.4 Editing

– Command *Levels > Edit*. In this case, the required level must be selected from the list that appears. In the list, a level is represented by its identifier;

– Command *Edit* of the context menu of the required level;

– *Double click* on the required level.

5.3.5 Removing

– Command *Levels > Remove*. Select the required level from the list that appears. In the list, a level is represented by its identifier;

– Command *Levels > Remove all*. Deletes all levels added to the oscillogram at once;

Command *Remove* of the context menu of the required level.

6 Analysis tools

Analysis tools are panels for various purposes, which can be displayed on the screen in the form of floating windows or docked to the left /right/bottom from the oscillogram field in the main window. The same analysis tool can be added to display an unlimited number of times. At the same time, each tool instance can be configured independently of others. In particular, a simultaneous analysis of different time sections of the oscillogram is possible.

To add an instance of an analysis tool, select its name in the *Tools* menu.

Each analysis tool contains a window title, a menu icon on the left, and a standard cross for closing on the right. Right-click anywhere on the tool also calls the tool's context menu.

Menu of every analysis tool includes a *Save image* item, which allows to save the actual tool view in an image file.

6.1 Phasor diagram

The mapping of phasors on the complex surface.

FEATURES

– Allows selecting any analog signal as the base phasor of the diagram.

– Displays both the actual moment (determined by the cursor position) and moment fixed with a marker. This makes it possible, in particular, to analyze the same signals at different moments simultaneously, by opening the required number of tool instances that are equally adjusted, but tied to different moments of time.

– Displays the angular zones of the user-defined color on the diagram, which in some cases improves the visual perception of information.

– Supports several scaling modes of phasor lengths in the diagram, and also a mode *Show phasor direction only* in which all phasors, regardless of their module, are displayed equal to the outer circle radius of the diagram scale grid.

– Supports addition of the displayed signals by dragging them (as well as entire signal strips) with mouse.

– Displays signal names at the ends of phasors (convenient for signals with short names) and their serial numbers in the value table (convenient for signals with long names).

– Allows setting the frequency basis of phasors (by default, it is the first harmonics of the oscillogram main frequency)

MENU

Command	Brief description
Settings	Display of tool setup dialog
Time moment	Selection of the moment displayed in the diagram
Show phasor direction only	Turn on/off displaying the direction of phasor only
Show signal names	Turn on/off displaying the signal names in the diagram

SETTINGS WINDOW - SIGNALS

The page allows selecting signals, which phasors have to be displayed, indicate the base phasor and select the frequency basis.

SETTINGS WINDOW - ZOOM

The page allows setting the scaling parameters of phasors.

Phasors with the same measurement unit are scaled together and independently of phasors with other measurement units. That is, currents with currents, voltages with voltages, etc.

Two modes that determine how to scale the signals are supported:

– *Calculate zoom for time moment on this diagram* – only the phasor modules of the moment displayed on the diagram are taken into account. Thus, the largest phasor by module of a certain measurement unit will always have a length equal to the outer circle radius of the scale grid. This makes the viewing of oscillograms' sections with small signal values convenient, because their display does not depend on sections with large values. Visual comparison of the same phasor lengths at different points of time in this mode is incorrect.

– *Calculate zoom for the whole time range* – considers the signals values for the entire duration of the oscillogram. This mode is convenient when it is necessary to analyze the same phasors at different points of time, because it allows setting a common scale for them.

Two options for choosing signals that affect scaling:

– *Use only signals on the diagram to calculate zoom* – calculating zoom, when only the signals that were displayed on the diagram are taken into account.

– *Use all signals to calculate zoom* –calculating zoom, when all oscillogram signals are taken into account, even those that are not on the diagram.

SETTINGS WINDOW - SECTORS

The page allows managing the list of displayed angle sectors

6.2 HARMONICS

Displaying of diagrams for the harmonic content of analog signals.

FEATURES

– Displays both the actual moment (determined by the cursor position) and moment fixed with a marker. This makes it possible, in particular, to analyze the same signals at different moments simultaneously, by outputting the required number of tool instances that are equally adjusted, but tied to different moments of time.

– Supports addition of the displayed signals by dragging them (as well as entire signal strips) with mouse.

– Supports selection of the number of displayed harmonics (within the limits allowed by sampling theorem).

MENU

Command	Brief description
Settings	Display of tool setup dialog
Time moment	Selection of the moment displayed in the diagram

SETTINGS WINDOW - SIGNALS

The page allows selecting the signals which harmonic content should be displayed.

SETTINGS WINDOW - CHART

Allows selecting the number of displayed harmonics and indicating, which value to take as 100%.

6.3 POTIER DIAGRAM

Potier diagram for the analysis of the generator rotor current.

MENU

Command	Brief description
Settings	Display of tool setup dialog

SETTINGS - INPUT

The page allows selecting three-phase groups of voltage and current of generator.

SETTINGS WINDOW - SETTINGS

Page allows specifying the necessary settings, basic values and SC characteristic parameters.
SC characteristic is displayed in the diagram as a bold blue line.

SETTINGS WINDOW - IDLE CHART

This page allows setting the characteristic of the idle run (HH). Support 5-8 sections. HH characteristic is displayed in the diagram as a bold red line.

6.4 Fault locator

Location of faults on electric power lines. Supports complex lines with many segments (including parallel lines) and taps.

FEATURES

– If you work with the oscillogram of BE2704/ED7 or BE2502/ED5, there is an option to upload the line descriptions from the settings of oscillogram body;

– Supports uploading of line descriptions from **.tlc* и **.pline* files, which are used for setting of FLOC function in BE2704/ED7 and BE2502/ED5 devices;

– Supports saving line description in **.wavespowerline* files, and uploading from them;

– Shows all faults possible for this line.

MENU

Command	Brief description
Settings	Display of tool setup dialog
Pre-fault mode	Selection of time of the pre-fault mode data intake

SETTINGS WINDOW

Allows creating and editing the line description.

6.5 Spectrum

Mapping of the amplitude spectrum (hereinafter - spectrum) diagram of analog signals.

FEATURES

– Displays analog signal spectrum along the whole length of the oscillogram, as well as the spectrum of the marked sections of signal uniformity.

– Supports addition of the displayed signals by dragging them with mouse.

– Displays value (as a percentage) in the point of click on the spectrum field.

– Displayed spectrum frequency range is limited according the sampling theorem.

MENU

Command	Brief description
Settings	Display of tool setup dialog
Begin of spectrum	Selection of the beginning of oscillogram uniformity
End of spectrum	Selection of the end of oscillogram uniformity
Zoom out by frequency	Compress the spectrum horizontally
Zoom in by frequency	Stretch the spectrum horizontally
Zoom out by amplitude	Compress the spectrum vertically
Zoom in by amplitude	Stretch the spectrum vertically

SETTINGS WINDOW

The page allows selecting the signals which amplitude spectra should be displayed.

6.6 Statistics

View the statistic information about analog signals for the selected period. A minimum, maximum, mean and median values are shown.

FEATURES

– Supports addition of displayed signals by dragging them (as well as entire signal strips) with mouse.

– Displays data for the whole oscillogram, as well as for the marked section only.

MENU

Command	Brief description
Settings	Display of tool setup dialog
Begin of range	Selection of the beginning of processing area
End of range	Selection of the end of processing area

SETTINGS WINDOW

For selecting the signals that have to be displayed in the table.

6.7 Values table

Viewing the values of marked analog signals at timestamps in the form of a table.

FEATURES

Supports addition of displayed signals by dragging them (as well as entire signal strips) with mouse.

MENU

Command	Brief description
Settings	Display of tool setup dialog

SETTINGS WINDOW

Allows selecting the signals that should be displayed in the table

6.8 RELAY MODELS OF BE2704/ED7 AND BE2502/ED5

Tools of this group are the different relay models of IEDs БЭ2704 and БЭ2502. **Principles of their work are described in the documentation for these types of IEDs or cabinets**, in which they are implemented and are not included in the present manual.

6.8.1 COMPOSITION

Composition of relay models of IEDs BE2704/ED7 and BE2502/ED5:

– Phase-to-ground distance relay;

– Phase-to-phase distance relay;

– Faulty phase selector relay;

– Faulty phase selector compensated relay.

Relay models display the hodograph of the corresponding resistance sampling on complex surface, and show the facts of actuation of the corresponding relay components.

Models create the beginning and ending points of intervals for resistance hodograph display. Oscillogram beginning, actual moment (determined by the cursor) or any moment fixed with a marker can be selected as the beginning point. The end can be represented by the actual moment, the end of the oscillogram, or any moment fixed with a marker.

6.8.2 Setting

MENU

Command	Brief description
Settings	Display of tool setup dialog
Begin of hodograph	A group that allows you choosing a point in time used as the beginning of the interval for displaying resistance hodographs.
End of hodograph	A group that allows selecting a point in time used as the end of the interval for displaying resistance hodographs.
Show hodograph/ show active point	Switches the modes of displaying resistance sampling between hodograph for a selected period and the only value in the oscillogram active point.

SETTINGS - INPUTS

Selection of input signals for the analyzed relay.

SETTINGS - POWERLINE

Powerline parameters that are specified for all relays. If you work with a BE2704/ED7 or BE2502/ED5 oscillogram, then the *Obtain from this oscillogram* button becomes active, which allows loading the relevant settings from oscillogram body.

SETTINGS WINDOW - FAULT TYPE DETECTION

Parameters of fault type detection that are specified for all relays, except for *phase-to-phase distance relay*. If you work with a BE2704/ED7 or BE2502/ED5 oscillogram, then the *Obtain from this oscillogram* button becomes active, which allows loading the relevant settings from oscillogram body.

SETTINGS WINDOW - CHART

An explanatory drawing is given on the parameter-setting page, which shows all the parameters that need to be set.

The available set of parameters allows describing the resistance relay characteristics of any BE2704/ED7 or BE2502/ED5 IED ever manufactured. If you work with a BE2704/ED7 or BE2502/ED5 oscillogram, then the *Obtain from this oscillogram* button becomes active, which allows loading the relevant settings from oscillogram body.

6.8.3 Operation with a complex surface

Within the complex surface, a number of operations using mouse can be performed.

– rotation of the wheel with the mouse cursor over the surface changes its scale;

– having captured the surface with mouse, you can move it within the display window;

– when the mouse cursor stops over any point on the surface, a tooltip with its coordinates is displayed.

6.9 RELAY MODELS OF EKRA200/ED2

Tools of this group are the different relay models of EKRA200/ED2 IED. **Principles of their work are described in the documentation for these types of IEDs or cabinets**, in which they are implemented and are not included in the present manual.

The set consists of one *Phase-to-phase distance relay*. The tool is identical to the BE2704/ED7 and BE2502/ED5 IED tool, but models the phase-to-phase distance relays of EKRA200/ED2, which differs from them in the principle of resistance sampling and used characteristic.

7 PRINTING OF OSCILLOGRAM

The *File > Print* command enables printing of the oscillogram. Selection of this command opens a preview window.

Use the *Printer settings* button to select a printer, configure its settings, set page options, and select a print mode.

The *Print parameters* button specifies a number of additional design options. The results of parameter changes can be immediately seen in the preview area.

A visible range of the oscillogram is sent to printing.

The oscillogram is printed as it is currently present on the screen with all the tools on it.

Hidden signals are not printed.

8 Saving of oscillogram

Oscillograms are saved in Waves own format (*.w*aves)*. All source data that existed in the oscillogram recorded by the device, as well as all actions performed during the analysis, are saved. Therefore, having opened such file later, you can continue to work from the place where you have stopped at the time of saving.

To save an open oscillogram, use the *Save* and *Save As* commands from the *File* menu.

The *File > Save as* command always prompts to specify the path and name for saving the file.

The File > *Save* command works either the same way, if oscillogram has never been saved before, or it saves the file under the current name, if oscillogram is opened from a **.waves* file or has been saved at least once before.

9 Exporting of oscillogram

Export allows to save an oscillogram in one of supported formats.
Only a visible time range of oscillogram is exported.
Only visible (not hidden) signals are exported, as they are currently ordered on the oscillogram panel (from the top down).

9.1 Export as COMTRADE

An opened oscillogram can be exported in COMTRADE format under *File > Export > Comtrade XXXX*, where XXXX is Comtrade version.

> Only the mandatory files (or parts of CFF-file) are formed: CFG and DAT.

When exporting you can:

– select CFG encoding; **DOS / Windows / UTF8**;

– select DAT format: **Text / Binary**;

– select format of analog signal sample: **Int16 / Int32 / Float32**;

– select value type of analog signals: **Primary / Secondary**;

– set the time zone;

– perform resampling to the required rate.

9.2 Export as CSV

An opened oscillogram can be exported in CSV format under *File > Export > CSV*. The file has a form of text table with sample values of all exported signals for the exported period.

> *CSV* file can be used for calculations in *Microsoft Excel* or using programs developed in-house. Reading of oscillogram in *CSV* format, issued by *Waves* is way easier then reading of the same oscillogram in *Comtrade* format.

When exporting you can:

– select encoding; **DOS / Windows / UTF8**;

– select analog signal displayed value: **Instanteous / Effective…**;

– select value type of analog signals: **Primary / Secondary**;

– indicate, if it is necessary to include binary signals in the file.

10 Oscillogram combination

Waves has functions for combination of two and more oscillograms in one. There are two types of combination:

Superimposition - simultaneous analysis of different devices' recordings of the same process.

Joining - joining of oscillograms recorded sequentially by one device.

10.1 Superimposition

A simultaneous analysis of the data for the same accident, recorded by various RPA or ER devices, is often necessary. To perform this analysis, *Waves* offers the superimposition of two oscillograms.

The application supports two modes of superimposition:

– Automatic, by absolute time;

– Manual.

10.1.1 Automatic superimposition by absolute time

Allows to automatically combine several oscillograms in one, guided by astronomical times, indicated in them.

> To use this mode, make sure that devices, which oscillograms are combined, were correctly synchronized when a failure was recorded.

To start, select the *Superimpose > Automatically* command from the main menu.

STEP 1 - SELECT OSCILLOGRAMS

Select oscillograms that have to be combined.

STEP 2 - SELECT SIGNALS

Select the signals from selected oscillograms that have to be included in the combined oscillogram.

The *Group signals with same name* checkbox allows placing signals with the same names near each other.

STEP 3 - ENTER NEW OSCILLOGRAM NAME

Specify the name of a new oscillogram.

10.1.2 Manual superimposition

Allows combining two oscillograms by curve shapes and phasor diagrams. By sequentially performing this operation, you can combine any number of oscillograms.

To start, select the *Superimpose > Manually* command from the main menu.

STEP 1 - SELECT OSCILLOGRAMS

Select two oscillograms you want to combine: the main one (oscillogram 1 - red), that will determine the timeline of the combined oscillogram, and the additional one (oscillogram 2 - blue).

STEP 2 - COMBINATION

Precisely synchronize these oscillograms.

To do this, select by one signal, more convenient for combination purpose, from each of them. In some cases, a calculation of convenient signals may be required.

These signals will be shown together in the chart field of the wizard window (vertical lines of the signal color indicate the trigger moment of the corresponding oscillogram).

Oscillogram signal 1 (red) is stationary, oscillogram signal 2 (blue) can be freely moved along the time axis by grabbing it with mouse.

The same rules of scaling and moving along the time axis as in the main window are applicable in the graph field.

Signals are output in the individual scaling mode. The wizard window size can be changed, size of the graph display area changes along with it.

The following buttons are available on the toolbar under the signal graphs:

Button	Description
Find delta > By trigger timestamps	Performs precise alignment of trigger timestamps. Combines oscillograms that were recorded from one disturbance, even if clocks in the devices were not synchronized
Find delta > By absolute time	Performs a combination of samples according to astronomical time. Combines oscillograms recorded by devices that have been precisely synchronized
Allow free movement	Enables and disables the mode in which the movement of signal 2 is possible without associating to the sampling period

For easier combining there is also a phasor diagram, where signals from combined oscillograms can be added, notably, not necessarily the ones that were selected for graph display.

STEP 3 - SELECT SIGNALS

Mark the signals from both oscillograms that have to be included in the combined oscillogram.

The *Group signals with same name* checkbox allows placing signals with the same names near each other.

STEP 4 - ENTER NAME OF NEW OSCILLOGRAM

Specify the name of a new oscillogram.

10.1.3 Remarks

During combination, the signals of original oscillograms can undergo changes as follows:

– if original oscillograms have different sampling rates, then the combined oscillogram will be generated with the sampling rate, the largest from the original ones;

– if during synchronization samples of the second and subsequent oscillograms are shifted from samples of oscillogram 1 by a value not multiple to the sampling period, then the analog signals of the combined oscillogram taken from them can slightly change their form due to interpolation.

10.1.4 DATA SOURCE

When superimposed, signals of oscillograms are united in a *data source*. The data source contains full information on the oscillogram, including: oscillogram name, information on device that recorded it, all additional data (settings, trigger reason, etc.) from the original oscillograms.

Every data source has its submenu in the *File* menu, with commands relevant only to it.

Data source have their names. By default, the data source name consists of letter **D** with its sequential number in the combined oscillogram. The name can be changed manually, using the *Properties* command of the data source menu. Data source name is shown as a prefix before the signal names relevant to it.

10.2 JOIN

Some RPA and ER devices record several subsequent oscillograms during one accident. Sometimes there can be few seconds' pauses between them.

In particular, EKRA devices in the continuous start mode record two oscillograms: the first oscillogram contains the moment of accident, and the second oscillogram contains the moment of actuation signal clearing.

There are also devices that due to technical reasons do not record long processes into one file, but record several small fragments. Some devices record back-to-back, others - overlapped.

Such oscillograms are convenient to analyze as if they are in one continuous recording. By joining fragments, such combining can be realized.

> Only identical signals that are found in all selected fragments can enter the resulting oscillogram. Signals having the same names, transformation ratios and measurement units are considered to be identical. Of course, if the selected oscillograms are actually recorded by the same device, then all signals will be candidates for getting into the new oscillogram.

The application supports two modes of joining: Automatic by absolute time, and Manual.

10.2.1 AUTOMATIC JOINING BY ABSOLUTE TIME

Allows automatic joining of several fragments, recorded by one device into one oscillogram, guided by the absolute times indicated in them. Allows joining fragments with pauses between them. Pauses between fragments are filled with the value *No data*.

To start, select the *Join > Automatically* command from the main menu.

STEP 1 - SELECT FRAGMENTS

Select fragments that will be automatically joined.

STEP 2 - SELECT SIGNALS

Select signals to be added to a new oscillogram.

STEP 3 - ENTER NAME OF NEW OSCILLOGRAM

Specify the name of a new oscillogram.

10.2.2 MANUAL JOINING

Allows manual joining of two fragments, visually guided by curve shapes, and knows how to find overlap intervals. By sequentially performing this operation, you can join any number of oscillograms.

> Manual joining has to be applied only when the device records the wrong timestamps into the fragments. It is worth to be reminded, that **EKRA devices do not write oscillograms in such way**. This tool has been developed to ease the joining of oscillogram fragments from a number of other manufacturers' devices that have these problems.

To start, select the *Join > Manually* command from the main menu.

STEP 1 - SELECT FRAGMENTS

Select two oscillograms which have to be joined: the main one (beginning - red), which will determine the timeline of the combined oscillogram, and the additional one (end - blue) that needs to be joined.

STEP 2 - COMBINATION

Precisely combine these oscillograms. To do this, select a signal, more convenient to use for combination. This signal from both fragments will be shown together in the chart field of the joining window (vertical lines of the signal color indicate the trigger moment of the corresponding oscillogram).

Signal of the beginning fragment (red) is stationary, and signal of the end fragment (blue) can be freely moved along the time axis by grabbing it with mouse.

The same rules of scaling and moving along the time axis as in the main window are applicable in the graph field. The wizard window size can be changed, size of the graph display area changes along with it.

The following buttons are available on the toolbar under the signal graphs:

Button	Description
Find delta > By absolute time	Joins samples according to astronomical time.
Find delta > By common part	Searches for fragments' common part.
Allow free movement	Enables and disables the mode in which the movement of signal 2 is possible without binding to the sampling period

It may seem that visual joining of fragments, and especially *joining by common part*, are not required. In most cases, it is true, that is why after the second step, the fragments are automatically joint by absolute time and you can press the *Next* button right away. But it appears that a lot of **third-party devices** put incorrect timestamps in joint fragments and overlap them. These fragments can be joined only by finding their common part visually or through a search command, which also sometimes finds the overlap wrongly due to a number of other errors in such oscillograms.

STEP 3 - SELECT SIGNALS

Select signals to be added to a new oscillogram.

STEP 4 - ENTER NAME OF NEW OSCILLOGRAM

Specify the name of a new oscillogram.

10.2.3 REMARKS

If *Comtrade* oscillograms are used as original fragments, microsecond deviations in the timestamps in the second and further fragments, relative to their original timestamps, are possible. This is due to the low accuracy of timestamps in the most existing Comtrade files.

11 OTHER COMMANDS FOR EKRA OSCILLOGRAMS

11.1 INTEGRATION OF DEVICE CONFIGURATION FILE

Oscillograms recorded by BE2704/ED7 and BE2502/ED5 contain settings, effective at the moment of oscilloscope trigger. *Waves* provides a set of functions to use them. To be used, these settings require a configuration file of a corresponding device.

Nowadays, *EKRASMS* software automatically embeds the proper configuration file into oscillogram during reading from IED, but it was not always the same. You can run into an oscillogram without configuration file.

When an oscillogram without configuration file is opened, *Waves* automatically searches for it on the computer. First, in the oscillogram folder. Then, if the file wasn't found, in the folder of *EKRASMS* configuration files. If there are no applicable files, the *File > Embed device configuration file* command will appear in the menu.

The command displays a short guide, describing the integration of device configuration file into the oscillogram, having requested it from the oscillogram supplier. After successful integration, this command disappears from the menu and is replaced by a set of commands described further in this section.

11.2 REASONS FOR TRIGGER

The *File > Reason for trigger* shows a list of binary signals that were the reason for the trigger of oscilloscope in the EKRA devices.

11.3 STATUS AT THE TRIGGER MOMENT

For oscillograms recorded by BE2704/ED7 and BE2502/ED5 devices, the *File > Status at the momet timestamp* command is available. It shows the statuses of all binary signals at the trigger moment.

Not all produced devices record the status of binary signals at the trigger moment, so the list can be empty.

It should noted that the signals are recorded with a delay of 10 ms after the actual trigger moment and, in the general case, may not reflect the situation that occurred at the true trigger moment.

11.4 ADDING SIGNALS FROM DATABASE

For oscillograms recorded by BE2704/ED7 and BE2502/ED5 devices, the *File> Add signals from database* command is available. It allows addition of missing binary signals from files with samples from the *EKRASMS* database to an opened oscillogram. All types of event files, used in the EKRASMS software, are supported: **.timeline, *.archive, *.datapak, *.dbbackup, *.db*.

11.4.1 PROBLEM OF EVENT GAPS

In the process of adding signals to the oscillogram, it is possible to form samples with a sign of inaccuracy. This is because gaps may occur when collecting events in the database. As a rule, this indicates an incorrect configuration of the device registration mask. As a result, events reflecting a change of the signal in one direction can occur sequentially. At the same time, it cannot be said at what moment between them the signal have changed its state in the opposite direction, and therefore, sections of false information are filled with the value *No data*.

11.4.2 PROBLEM OF TIME ROUNDING

Events in the device are generated with the sampling rate of the oscilloscope (usually 1,200 Hz), however, timestamps in them have an accuracy of 1 ms (i.e., events are laid out on a grid with a rate of 1,000 Hz). Time rounding is inevitable, and if the signal changes its value twice in neighboring samples, it may happen that the corresponding events receive the same timestamp in the database. If such conflict is detected, the sample is set to *No data*. It should also be understood that the difference in the sampling rate of the oscillogram and the time grid of the database can lead to errors in positioning of the state changes of the added signal by plus or minus one sample.

11.5 SETTINGS

The *File > Settings* command shows the settings of EKRA device at the moment of oscilloscope trigger.

For oscillograms of BE2704/ED7 and BE2502/ED5 IEDs, the settings will be opened in the *Atlas* application, if the EKRASMS package is installed on the PC.

12 OSCILLOGRAM PROCESSING TEMPLATES

Waves allows saving almost all changes to oscillogram as a template script. Notably, it doesn't matter how the changes were made, through a calculation tree commands, creation of arbitrary expressions (see Section 13) or even developing and implementation of some scripts (see Section 14).

Later the template script can be applied to the other oscillogram received from that (or similar) device and repeat the processing, as for the source file.

Template script includes the following information:

– Changes in names, transformation ratios and measurement units of the recorded signals;

– Signals inversion;

– Adding of calculated signals with information on what formulas and what signals were used for calculation.

– Signal visibility;

– Integration of signals into three-phase circuits;

– Visual parameters of signals (color, line styles)

– Mutual alignment of signals on the display;

Information on markers, probes and levels is **NOT** saved to the template script, because everything that bound to the specific situation or timestamps, **becomes meaningless** for other oscillograms.

Select the *File > Save template* command from the oscillogram menu to save the template script.

For further application of template script select *File > Apply template* command, which execute the chosen template.

As template script represents an ordinary script, you can manually open the script editor, upload a relevant template and execute it. It helps to visually assess the actions to be made before actually starting them, which is useful when the template file name doesn't describe its purpose.

13 Expressions

Waves allows creating signals calculated based on text expressions. In an expression, it is possible to use arithmetic and logical operators, numerical constants, signals, and also built-in functions.

Create a new signal, using the command *Calculate > Calculate expression*. Input of expression in several lines is available for easier perception.

```
Freq(
    Float(
        Complex("Ia") * 2 + Complex("Ib")
    )
)
```

Expressions are not case sensitive. The accuracy is checked only at window closing with button *OK*.

For easier input of **signals** and **functions**, use the ***Ctrl+Space*** combination, which opens a drop-out list to choose the needed elements. They are inserted in the actual cursor position in the expression editor.
In addition, if the cursor is within a certain function, when you press the ***Ctrl + Shift + Space*** combination a tooltip with its prototype is displayed.

13.1 Signals

Signals are entered into expressions with their own names, enclosed in quotation marks. If the oscillogram is obtained through combining of several oscillograms, and as a result contains several data sources, signal names start with the data source name and divider :: (two colons).

```
"Ia"     – for ordinary oscillogram
"D1::Ia" – for combined oscillogram
```

13.2 Operators

13.2.1 Logical operators

Operator	Description	Example
not	NOT	not S1
or	OR	S1 or S2
and	AND	S1 and S2
xor	Exclusive OR	S1 xor S2

Existing binary signals and nested expressions for binary signal calculation, can be operands of logical operators. A binary signal is the result of logical operator execution.

13.2.2 Arithmetic operators

Operator	Description	Example
+	Addition	`S1 + S2`
-	Subtraction	`S1– S2`
*	Multiplication	`3 * S1`
/	Division	`S1 / 3`
-	Unary minus	`-S1`

Operands of arithmetic operators can be existing analog signals, numerical constants, as well as nested expressions for calculating analog signals.

Both operands of binary (having right and left operands) operators must be of the same type, both floating-point or both complex. For using complex operands, their frequency basis has to be the same.

The result of arithmetic operator execution is the signal of the same type as operands.

> The transformation ratio and measurement units of the result are calculated in accordance with the rules for a particular operation. However, this is not always possible. For example, it is impossible to say what result measurement unit will you get, if you for some reason add up the current and voltage. If it is impossible to calculate the property value, it is assigned with the default value Кт = **1**, measurement unit = <not defined>.

13.3 Functions

13.3.1 Function call

To call the built-in function, write its name, and then indicate the values of all its parameters in brackets, separated by commas.

```
Pos("Ia", "Ib", "Ic")
```

The result of function execution is the signal, which type is defined by the function itself.

> If function calculates analog signal, it defines its transformation ratio and measurement units.

13.3.2 Description of functions in the document

Function prototypes are provided during the description of functions.

> Prototype declaration format, accepted in this document, fully consistent with the **Pascal** programming language, except that the key word **function** is not used.

```
MyFunction(S: AnySignal; P1: Int = 0; P2: Int = 5; P3: Int = 0): FloatSignal
```

Here **MyFunction** is a function name, **S, P1, P2, P3** – parameters' names, **AnySignal** and **Int** are parameters' types, values after the equation mark – default values for *optional* parameters, and **FloatSignal** after the closing bracket is the result type.

Three last parameters of function (P1, P2, P3) are declared as *optional*, because they have default values. Only the last one or several last function parameters can be optional.

When using a function in the expression, you can omit the optional parameters if you are satisfied with their default values. It worth reminding that optional parameters can be omitted only starting from one of it and up to the end. For example, the **MyFunction** cannot be called, when P1 is omitted, but P2 is indicated and P3 is omitted. At the same time the call is permissible when P1 is indicated, P2 and P3 are omitted.

Mandatory parameters of the same type, following each other, can be listed separated by commas with single indication of the type for the whole parameter group.

```
MyFunction2(S: AnySignal; P1, P2, P3: Int): FloatSignal
```

13.3.3 Types of function parameters

Type	Description
`Bool`	Logical (possible values `False` and `True`)
`Int`[1)]	Integer number
`Float`[1)]	Floating-point number
`AnySignal`	Any type signal
`BinarySignal`	Binary signal
`FloatSignal`	Analog signal
`ComplexSignal`	Complex signal

1) Allows restrictions on the range of possible values

The function result can be only one of the signal types (**xxxSignal**).

13.4 Constants

13.4.1 Logical constants

Waves supports only two logical constants: **False** (0) and **True** (1).

Logical constants can be used as values for the **Bool** functions parameters, and as binary signals of fixed value, also used as parameters of **BinarySignal** or **AnySignal** type functions.

```
Shift("Ia", -10, True)
```

> Expression consisting only of the logical constant, creates a binary signal filled with value corresponding to this constant.

13.4.2 Integer constants

Integer constants can be used as values for the parameters of **`Int`** and **`Float`** functions, and as analog signals of fixed value, also used as parameters of **`FloatSignal`** or **`AnySignal`** type functions.

```
Abs("Ia" + 10)
```

Expression consisting only of the integer constant creates an analog signal filled with value corresponding to this constant.

13.4.3 Floating-point constants

Floating-point constants can be used as values for the parameters of **`Float`** type functions, and as analog signals of fixed value, also used as parameters of **`FloatSignal`** or **`AnySignal`** type functions.
The decimal separator for floating-point constants is the “**`.`**” (point)

```
12.4 * "Ia"
```

Expression consisting only of the floating-point constant creates an analog signal filled with value corresponding to this constant.

13.4.4 Nil constant

A number of functions (see below) which are available for call from the expression can operate three phase-to-phase signals. As all three phase-to-phase signals are not always available, it is permitted to call these functions while indicating **`Nil`** constant in place of one absent signal.

13.5 Conversion of signal type

13.5.1 Binary

Conversion of any type input signal (S) into a **binary** signal.

```
Binary(S: AnySignal): BinarySignal
```

S type	Result
`BinarySignal`	Copy of input signal
`ComplexSignal`	If the *module* of input signal sample is =0.0, new signal sample is =0. In other cases =1
`FloatSignal`	If the input signal sample is =0.0, new signal sample is =0. In other cases =1

13.5.2 Float

Conversion of any type input signal (S) into an **analog** signal.

```
Float(S: AnySignal): FloatSignal
```

S type	Result
`BinarySignal`	If the input signal sample is =0, new signal sample is =0.0. In other cases =1.0
`ComplexSignal`	Inverse discrete Fourier transform from the frequency basis of input signal
`FloatSignal`	Copy of input signal

13.5.3 Complex

Conversion of any type input signal (S) into a **complex** signal in the frequency basis of the main frequency (F) and harmonics (H).

```
Complex(S: AnySignal; F: Int = 0; H: Int = 1): ComplexSignal
```

Parameter	Type	Min	Max	Description
`F`	`Int`	0	SR[1]/2	Main frequency =0 – the main frequency of oscillogram is used
`H`	`Int`	1	[2]	Harmonics number

[2] – SR (Sampling Rate) – oscillogram sampling rate
[3] – defined with the condition that harmonics frequency cannot be more than SR/2

S type	Result
`BinarySignal`	If the input signal sample is =0, new signal sample is =0.0/0°. In other cases =1.0/0°
`ComplexSignal`	Input signal copy with the possibility to change the frequency basis
`FloatSignal`	Discrete Fourier transform in the set frequency basis

13.6 Signal shift: Shift

Creation of the copy of input signal (S) shifted by the set value (V) to the left (V < 0) or the right (V > 0).

```
Shift(S: AnySignal; V: Int; InSamples: Bool = False): AnySignal
```

Optional parameter **InSamples** sets the measurement units of the parameter **V**. If **InSamples =True** then the shift is set in samples, if **=False** - in milliseconds, which are rounded to the nearest integer of samples during the execution of functions.

13.7 Composition/decomposition of complex signal

13.7.1 ComplexA, ComplexP

ComplexA and **ComplexP** functions are designed to compose a complex signal from separate elements.

ComplexA composes the result from real (Re) and imaginary (Im), and ComplexP - from the module (Mag) and angle (Ang) in degrees.

```
ComplexA(Re, Im: FloatSignal; F: Int = 0; H: Int = 1): ComplexSignal
ComplexP(Mag, Ang: FloatSignal; F: Int = 0; H: Int = 1): ComplexSignal
```

Both functions assign the result with the frequency basis of the main frequency (F) and harmonics (H). For more information on frequency basis, see the **Complex** function.

13.7.2 Re, Im, Mag, Ang

Functions of this section allow decomposing the **complex** signal (S) to separate elements. Every function creates **analog** signal to a relevant element.

```
Re(S: ComplexSignal): FloatSignal  - real part
Im(S: ComplexSignal): FloatSignal  - imaginary part
Mag(S: ComplexSignal): FloatSignal - module
Ang(S: ComplexSignal): FloatSignal – angle in degrees
```

13.8 Symmetrical components

Symmetrical components can be calculated both by phase and phase-to-phase input values (with **2** in the name ending).

The calculation functions of zero sequence contain the last optional parameter **E**, which has the ratio degree **$(\sqrt{3})^E$** for setting the correlation of transformation ratios by star and delta circuits, and can be selected in the range [-2..2].

13.8.1 Pos(2), Neg(2), Zero(2)

This set of functions allows calculating the symmetrical components in complex form (by formulas from Fundamentals of Electrical Engineering).

POSITIVE SEQUENCE

```
Pos(A, B, C: ComplexSignal): ComplexSignal
Pos2(AB, BC, CA: ComplexSignal): ComplexSignal
```

NEGATIVE SEQUENCE

```
Neg(A, B, C: ComplexSignal): ComplexSignal
Neg2(AB, BC, CA: ComplexSignal): ComplexSignal
```

ZERO SEQUENCE

```
Zero(A, B, C: ComplexSignal; E: Int = 0): ComplexSignal
Zero2(AB, BC, CA: ComplexSignal; E: Int = 0): ComplexSignal
```

13.8.2 PosF(2), NegF(2), ZeroF(2)

This set of functions allows calculating the symmetrical components in analog form using the input signal shifts and without discrete Fourier transformation.

> All functions contain an optional parameter **F** - the main frequency of input signals. Value **0** of this parameter uses the main frequency of oscillogram.

POSITIVE SEQUENCE

```
PosF(A, B, C: FloatSignal; F: Int = 0): FloatSignal
PosF2(AB, BC, CA: FloatSignal; F: Int = 0): FloatSignal
```

NEGATIVE SEQUENCE

```
NegF(A, B, C: FloatSignal; F: Int = 0): FloatSignal
NegF2(AB, BC, CA: FloatSignal; F: Int = 0): FloatSignal
```

ZERO SEQUENCE

```
ZeroF(A, B, C: FloatSignal; F: Int = 0; E: Int = 0): FloatSignal
ZeroF2(AB, BC, CA: FloatSignal; F: Int = 0; E: Int = 0): FloatSignal
```

13.9 Power of three phases: Power(2)

Creation of **complex** signal of the power of three-phase feeder using phase (`Power`) or phase-to-phase (`Power2`) voltages.

```
Power(UA, UB, UC, IA, IB, IC: ComplexSignal): ComplexSignal
Power2(UAB, UBC, UCA, IA, IB, IC: ComplexSignal): ComplexSignal
```

13.10 Signal frequency: Freq

Creation of **analog** signal, which samples present a frequency of input **analog** signal (S) in the corresponding timestamps.

```
Freq(S: FloatSignal; AvgPds: Int = 4): FloatSignal
```

> The optional parameter **AvgPds** specifies the *number of averaging periods* when calculating the frequency, which can be selected in the range **[1..16].**

13.11 REMOVING OF LINE CAPACITIVE CURRENT: RMCC

Creation of **analog** signal of the line current of the indicated phase (Ph), in which its capacitive current is removed.

```
RmCC(UA, UB, UC, IA, IB, IC: FloatSignal; B1, B0, Len: Float; Ph: Int;
  F: Int = 0): FloatSignal
```

Parameter	Type	Min	Max	Description
B1	Float	0	-	Capacitive transverse conductivity of the line positive sequence (μS/km)
B0	Float	0	-	Capacitive transverse conductivity of the line zero sequence (μS/km)
Len	Float	0	-	Line length (km)
Ph	Int	0	2	Phase to be calculated **0 -** A, **1 -** B, **2 -** C
F	Int	0	SR[1)]/2	Main frequency =0 – the main frequency of oscillogram is used

[1)] – SR (Sampling Rate) – oscillogram sampling rate

13.12 REMOVING OF LOAD UNBALANCE: RMLD

Creation of **complex** signal, which samples have removed load flow of the input **complex** signal.

```
RmLd(S: ComplexSignal): ComplexSignal
```

13.13 COMPARISON OF ANALOG SIGNALS

13.13.1 CMP, CMPE, CMPG, CMPGE, CMPL, CMPLE, CMPNE

Functions of **Cmp** family are designed for creation of **binary** signal. **1** is entered into the sample of this signal, if the comparison condition for the corresponding samples of input **analog** signals (S1 and S2) is satisfied. Otherwise, **0** is entered into the sample.

Cmp function allows setting the comparison operation by specifying the values of the parameters E, L and G.

```
Cmp(S1, S2: FloatSignal; E, L, G: Bool): BinarySignal
```

Parameter	Type	Description
E	Bool	Compare S1 = S2
L	Bool	Compare S1 < S2
G	Bool	Compare S1 > S2

Dependence of the comparison conditions from the parameter E, L, G:

E	L	G	Condition	Explanation
FALSE	FALSE	FALSE	0	always 0
TRUE	FALSE	FALSE	S1 = S2	equal
FALSE	**TRUE**	FALSE	S1 < S2	less than
FALSE	FALSE	**TRUE**	S1 > S2	greater than
TRUE	**TRUE**	FALSE	S1 <= S2	less than or equal
TRUE	FALSE	**TRUE**	S1 >= S2	greater than or equal
FALSE	**TRUE**	**TRUE**	S1 <> S2	not equal
TRUE	**TRUE**	**TRUE**	1	always 1

The rest of the family functions compare by one of the conditions.

```
CmpE(S1, S2: FloatSignal): BinarySignal  – condition S1 = S2
CmpG(S1, S2: FloatSignal): BinarySignal  – condition S1 > S2
CmpGE(S1, S2: FloatSignal): BinarySignal – condition S1 >= S2
CmpL(S1, S2: FloatSignal): BinarySignal  – condition S1 < S2
CmpLE(S1, S2: FloatSignal): BinarySignal – condition S1 <= S2
CmpNE(S1, S2: FloatSignal): BinarySignal – condition S1 <> S2
```

13.13.2 MIN, MAX

Min and **Max** functions create an **analog** signal, which samples are respectively the minimum or maximum of the corresponding samples of two input **analog** signals S1 and S2.

```
Min(S1, S2: FloatSignal): FloatSignal
Max(S1, S2: FloatSignal): FloatSignal
```

13.14 CONVERSION OF BINARY SIGNALS

Functions of this section convert the input **binary** signal (S) into the output signal, according to the specific algorithm.

13.14.1 TIME DELAY: DELAY

Time delay (T) of the input signal for trip (Return = False) or return (Return = True).

```
Delay(S: BinarySignal; T: Int; Return: Bool = False;
  InSamples: Bool = False): BinarySignal
```

The optional parameter **InSamples** specifies the measurement units of the parameter T. If **InSamples =True** then the delay is set in samples, if **=False** - in milliseconds, which are rounded to the nearest integer of samples during the execution of function.

13.15 Conversion of analog signals

Functions of this section convert the input **analog** signal (S) into the output signal, according to the specific algorithm.

Part of functions contain an optional parameter **F** - the main frequency of input signal. Value **0** of this parameter uses the main frequency of oscillogram.

13.15.1 Absolute value: Abs

Absolute value of the input signal samples.

```
Abs(S: FloatSignal): FloatSignal
```

13.15.2 Arc cosine: ACos

Arc cosine of input signal samples.

```
ACos(S: FloatSignal): FloatSignal
```

13.15.3 Arc cotangent: ACtg

Arc cotangent of input signal samples.

```
ACtg(S: FloatSignal): FloatSignal
```

13.15.4 Arc sine: ASin

Arc sine of input signal samples.

```
ASin(S: FloatSignal): FloatSignal
```

13.15.5 Arc tangent: ATg

Arc tangent of input signal samples.

```
ATg(S: FloatSignal): FloatSignal
```

13.15.6 Constant component: Const

Selection of a constant component on the period of the input signal.

```
Const(S: FloatSignal; F: Int = 0): FloatSignal
```

13.15.7 Cosine: Cos

Cosine of input signal samples. The input signal represents an angle in degrees.

```
Cos(S: FloatSignal): FloatSignal
```

13.15.8 Cotangent: Ctg

Cotangent of input signal samples. The input signal represents an angle in degrees.

```
Ctg(S: FloatSignal): FloatSignal
```

13.15.9 Derivative: Der

Derivative on the period of the input signal.

```
Der(S: FloatSignal; F: Int = 0): FloatSignal
```

> Resulting signal is brought to the measurement unit of input signal, so its scale and measurement unit are the same as of the input.

13.15.10 Mean value: Mean

Mean value on the period of the input signal.

```
Mean(S: FloatSignal; F: Int = 0): FloatSignal
```

13.15.11 Peak detector: Peak

"Peak detector" filter of the input signal.

```
Peak(S: FloatSignal; Tau, Limit: Int): FloatSignal
```

Parameter	Type	Min	Max	Description
`Tau`	`Int`	0	10000	Attenuation time constant (ms)
`Limit`	`Int`	0	10000	Action time restriction (ms)

> Only peaks in the positive direction are processed. Invert the input signal (-1*S) to process negative peaks. To process both negative and positive peaks at the same time, you must first obtain the (**Abs**(S)) absolute value of the signal.

13.15.12 Root-mean-square value: RMS

Root-mean-square value on the period of the input signal.

```
RMS(S: FloatSignal; F: Int = 0): FloatSignal
```

13.15.13 Sine: Sin

Sine of input signal samples. The input signal represents an angle in degrees.

```
Sin(S: FloatSignal): FloatSignal
```

13.15.14 Square root: Sqrt

Square root from the input signal samples.

```
Sqrt(S: FloatSignal): FloatSignal
```

13.15.15 Tangent: Tg

Tangent of input signal samples. The input signal represents an angle in degrees.

```
Tg(S: FloatSignal): FloatSignal
```

13.15.16 Angle normalization: To180

Bringing of the input signal angle in degrees to the range of [-180°..+180°].

```
To180(S: FloatSignal): FloatSignal
```

13.16 Conversion of complex signals

Functions of this section convert the input **complex** signal (S) into the output signal, according to the specific algorithm.

13.16.1 Conjugate: Conj

Conjugate value of the input signal samples.

```
Conj(S: ComplexSignal): ComplexSignal
```

13.16.2 Rotation: Rot

Rotation of input signal samples by the angle (Ang) in degrees.

```
Rot(S: ComplexSignal; Ang: FloatSignal): ComplexSignal
```

`FloatSignal` is used as the `Ang` parameter type, not `Float`, so the angle can be non-constant. However, as the FloatSignal type parameter can take a floating-point constant in some particular cases, the parameter value can be a constant number too.

13.17 Counter: Count

Counter adding (or deducting) 1 on every next sample of the set oscillogram interval.

```
Count(Start: Int = 0; Stop: Int = 0; Initial: Int = 0;
  Countdown: Bool = false): FloatSignal
```

Parameter	Type	Min	Description
Start	Int	0	Sample index of the counting interval start
Stop	Int	0	Sample index of the counting interval end (0 - to the end of oscillogram)
Initial	Int		Counter initial value
Countdown	Bool		Countdown (with step of 1)

Sample indexes can be viewed right on the oscillography graphs, zooming in its time till the moment when the numbering appears

14 SCRIPT

Script is a program written by a user in accordance with specific rules. Thanks to scripts, it is possible to automate routine actions that have to be performed continuously.

For example, you constantly get oscillograms from some device and always perform a set of identical actions before starting the analysis. You hide something, organize signals in a certain way, and make specific calculations. You can describe all these actions in a script (or make them once and save a template script with the *File > Save template* command). Later you will spend much less time on this, reducing it to a couple of clicks.

14.1 CODE EDITOR

A code editor for working with scripts is built into the *Waves*, the visibility of it can be controlled using the *Tools> Script* command.

In addition to the input field, the code editor contains its own menu:

Command	Brief description
File	
Open	Opens a script saved in a file to the editor:
Open from old library[1)]	Opens a script saved in the *Waves 3* library. After opening, this script has to be converted using *Actions > Convert old script to new*.
Save	Saves current script to a file
Edit	
Undo	Undoes the last action in the code editor
Cut	Cuts selected text to clipboard
Copy	Copies selected text to clipboard
Paste	Pastes text from the clipboard at the cursor position in the code editor
Delete	Deletes selected text
Select all	Selects all script text
Actions	
Convert old script to new	Converts script written for *Waves 3* to actual version
Execute	Executes script

1) - There is no such command, if there is no Waves 3 library on the computer

The editor has a contextual insert. It replaces the manual input (when mistakes are possible) of the signal names, commands, constants and functions of expressions for signals calculation with a selection from the drop-down list. The contextual inserts list appears when you press ***Ctrl + Space***, and contains all elements of the specified entities that can be inserted at the current position of the code editor. If the list does not appear when you click a specified combination, it is impossible to enter any of the above in the current position.

> In addition, if the cursor is within a certain command, when you press the key combination ***Ctrl + Shift + Space*** a tooltip with its prototype is displayed.

During script execution, a log appears under the code editor - a list of messages from runtime system issued during the process.

14.2 Comments

Comment is a text intended to be read by the user and passed by the runtime system. Usually comments are used for making notes to the code. The script supports two type of comments.

STRING COMMENT

String comment is a text starting from a couple of symbols “//” (slash, slash) and lasting till the line break:

```
// this is a string comment
```

BLOCK COMMENT

Block comment is a text between the pairs of characters “/*” (slash, asterisk) and “*/” (asterisk, slash), including line breaks:

```
/* this is a block
comment */
```

14.3 Literals

Literal is a constant, included directly in the program text. Literals can be numerical (*integer* and *floating-point*) and *string*.
String literals are set in quotes.

```
B = A + 10;                 // literal - 10
Rtm.Log(Msg = "My message") // literal - "My message”
```

14.4 Variables

Variable is a named object, that stores value or array of values of the one of 3 types **`string`** (character string), **`int`** (integer), **`float`** (floating-point number). Variable type is determined at the moment of indefinite declaration by the acquired value. The variable is available from the declaration point to the end of block in which it is declared.
The variable is declared using the key word **`let`**, followed by its name and, after the symbol "=” symbol (equal) - its initial value. The declaration is ended with the symbol ";” (semicolon).

```
let I = 10;               // integer variable
let F = 10.5;             // floating-point variable
let S = "10";             // string variable
let IL = [1, 2, 3];       // integer array
let FL = [1.0, 2.5, 3.6]; // floating-point number array
let SL = ["1", "2", "3"]; // string array
```

Type of declared variable and size of declared array cannot be changed later.
Value stored by the variable can be changed in the area of its visibility, by acquiring a new value to it.

```
I = 11;
F = 34.7;
SL[0] = "hello"; // the 0th element of array is changed
```

14.5 Dynamic conversion of type

The runtime system supports dynamic value conversion of variables and literals to the assigned variable type (or command parameter).
Rules of the dynamic conversion of values to the target type are shown in the table. Table rows contain the results of assignment of different value types on the script section, where a definite target value type is required.

Target type	**Assigned value**			
	2 (int)	2.8 (float)	"3.2" (string)	"any text" (string)
int	2	2	3	Error
float	2.0	2.8	3.2	Error
string	"2"	"2.8"	"3.2"	"any text"

In particular, it is clear, that an attempt to apply a floating-point value where an integer value is needed, cuts off its fractional part; while applying a string value where any number value type is preferred leads to conversion of string to a number of corresponding type, and if that fails, an execution error is generated.

14.6 Arithmetic operations

Arithmetic operations that are possible to be performed with integer and floating-point variables and literals: addition (+), subtraction (-), multiplication (*) and division (/).

All operators except for *division* result in an integer number, if both operands are integer, and in a floating-point number, if at least on operand is floating-point. Division always results in a floating-point number.

```
let A = 10;
let B = 21;
let C = A * B + 10.8; // variable will be created as floating-point with
                      // value 220.8
```

14.7 Concatenation of strings

String variables and literals can be concatenated (joined). The result is a string, consisting of all substrings that were joined in the order of its sequence in the expression.

```
let A = "Hello";
let B = A + " world!"; // result: "Hello world!"
```

Variables and number literals can be the concatenation operands. In this case, they are automatically converted into the string form.

```
let x = ", ";
let A = "one";
let B = 2;
let C = A + x + B + x + "3" + x + 4.5; // result: "one, 2, 3, 4.5"
let D = 10 + B + " " + A;              // result "12 times"
```

One operator (+) is used both for concatenation of strings and number addition, so it is necessary to remember the following rule to write expressions correctly:

> If at least one of the operands is a string, then operator performs concatenation and it will result in a string (all operations for calculation of variable **C** are concatenations). Otherwise it will perform addition with a number as a result (10 + B for calculation of variable **D** is an addition, the rest of it - concatenations). When additions and concatenations are mixed in one expression, it is recommended to use brackets for explicit indication of operation priorities.

Another some examples:

```
let A = 10 + 1;   // result int: 11
let B = 10 + "1"; // result string: "101"

let C = 0;        // concatenation with conversion to the int. type
C = 10 + "1";     // result int: 101
```

14.8 Blocks

Block is a script section, limited with symbols "{" and "}". Variables declared inside the block are local for it and not available outside it.

```
{
    let I = 10;
}
Rtm.Log(Msg = I) // ERROR: Variable I is not available
```

It is permissible to declare inside a block a variable with the same name that already exists outside it. In this case, these variables will be seen as different by the runtime system. Accessing to such variable inside the block returns it a value declared inside the block, accessing to it outside the block - the variable value declared outside it.

```
let I = 5;
{
    let I = 10;
    Rtm.Log(Msg = I) // output of value 10
}
Rtm.Log(Msg = I)     // output of value 5
```

14.9 Loops

Loop is a block, performed a set number of times.

```
Loop(I, 3)
{
    Rtm.Log(Msg = I)
}
```

Loop is declared using a key word **`Loop`**, followed by a loop variable of **`int`** type (**I**, as in the example above), and number of iterations (**3**, as in the example above) within the brackets and separated by a comma. Loop body (block to be performed a set number of times), is enclosed in curly brackets.

Loop variable doesn't need a special declaration and is available inside the loop block. Before the loop operation, its value is equal to 0 and increases by 1 after each iteration. Therefore, at the last loop iteration its value is 1 less than the set number of iterations.

> An explicit assignment of value to loop variable inside the loop body is ignored.

It is not obligatory to set the number of iterations with a literal; it can be an integer expression too.

```
let A = 5;
Loop(I, 2*(A + 4))
{
    Rtm.Log(Msg = I)
}
```

Loops can be embedded in each other.

```
Loop(i, 3)
{
    .. // only i is available here
     Loop(j, 2)
    {
        .. // i and j are available here
    }
    .. // only i is available here
}
```

14.10 Macros

Macros is an instrument that allows replacing a bulky, frequently used expression with a more compact definition.

Macros is declared using a key word **define**, followed by its formal parameters within the brackets, separated by a comma. This is followed by a symbol "=" (equal) and an expression, that macros will represent in the script. The declaration is ended by the symbol ";" (semicolon). Macros formal parameters can be used in the expression. When called-up they will be replaced by the transferred values.

```
define Desc(a, b, c) = b*b - 4*a*c;
```

When macros is defined, it can be called to perform the relevant calculation.

```
Rtm.Log(Msg = Desc(2, 3, 1)); // calculation result "1" will be included into log
```

Operation principle of macros is similar to that of preprocessor directive **#define** of the **C** language. At execution moment, the macros is replaced with the expression it defines in the call-up point, with arrangement of values of the transferred parameters.

Therefore, having met a macros call-up in the script, the runtime system:

```
Desc(2, 3, 1)
```

... replaces it with a relevant expression, arranging transferred parameter values in it:

```
(3*3 – 4*2*1)
```

... and only after this calculates the result.

14.11 Commands

Commands are the script entities, call-up of which influences the opened oscillogram within which the script is executed.

Command is called by writing its name followed by the named values of its parameters within brackets, separated by a comma. The sequence of named parameters is not important.

```
Sig.Create(Name = "MySig", Expr = Abs(2 * "Ia"), Color = Color.Red)
```

Parameters can be optional. Commands behavior when one of its optional parameters is omitted is defined by the command itself and will be described in its section.

The following types of command parameters are possible:

Type	Description
`Bool`	Logical (possible values `False` and `True`)
`Int`	Integer number
`Float`	Floating-point number
`String`	Character string
`String[]`	Array (list) of strings
`Expression`	A special type used only for the **Expr** parameter of **`Sig.Create`** command. Allows transferring an expression for signal calculation to the command; expressions have to be written according the rules, described in section 13.

Commands do not return values and are divided into several logical categories:

Category	Object of influence
`Rtm`	Script execution runtime system
`Sig`	Signals
`Tri`	Three-phase circuits
`Osc`	Oscillogram as a whole

14.12 RTM CATEGORY COMMANDS

14.12.1 RECORD MESSAGES INTO THE LOG: RTM.LOG

Records a message into the log of script execution. The command can be used for debugging.

Parameter	Type	Description
`Msg`	`String`	Message recorded to log

14.12.2 STOP EXECUTION: RTM.STOP

The command stops the script execution. The command can be used for debugging.

Parameter	Type	Description	If absent
`Msg`	`String`	Message recorded to log	No message for recording

14.13 SIG CATEGORY COMMANDS

14.13.1 SIGNAL CREATION: SIG.CREATE

Creation of a new signal.

Parameter	Type	Description	If absent
`Name`	`String`	Name of the created signal	
`Expr`	`Expression`	Expression for signal calculation	
`Sec`[1)]	`Bool`	Calculation in secondary values	`False`
`Q1`[1)]	`String`	Primary value of transformation ratio	Calculation result
`Q2`[1)]	`String`	Secondary value of transformation ratio	Calculation result
`NonPeriodic`[2)]	`Bool`	Always displays instantaneous value	Calculation result
`Color`	`Int`	Color	`Color.Black`
`Dashed`[1)]	`Bool`	Dashed	`False`
`Invert`	`Bool`	**Check box value:** Invert samples	`False`
`Noise`[1)]	`Bool`	**Check box value:** Signal contains noise	`False`
`Fail`	`Bool`	3)	`False`

1) Not used when creating a binary signal; 2) Used when creating analog signals only;
3) In case of error in `Expr` value, allows interrupting the script execution (`True`) or skipping command (`False`).

`Color` parameter value is a color index that can be seen in the list of signal color selection. Consequently, the length of this list limits its possible values. To ease the setting of color values, script has named constants for every index in the color list:

Constant	Value	Description
`Color.Black`	`0`	Black
`Color.Yellow`	`1`	Yellow
`Color.Green`	`2`	Green
`Color.Red`	`3`	Red
`Color.Blue`	`4`	Blue
`Color.Violet`	`5`	Violet
`Color.Turquoise`	`6`	Turquoise
`Color.Brown`	`7`	Brown
`Color.Gray`	`8`	Grey
`Color.Dark`	`9`	Dark

The `Q1` and `Q2` parameters have the **String** type, and in this case the string has to be composed according to definite rules. String must start from a number value, optionally followed (with or without space) by one of the measurement units from the table below.

Value	Physical Value	Description
	Any	No unit
`pu`	Any	Relative unit
`%`	Any	Relative unit as a percentage
`deg`	Angle	Degree
`C`	Temperature	Degrees Celsius
`Hz`	Frequency	Herz
`V`	Voltage	Volts
`A`	Current	Ampere
`ohm`	Resistance	Ohm
`S`	Conductivity	Siemens
`VA`	Power	Volt-ampere
`W`	Active power	Watt
`var`	Reactive power	Volt-ampere reactive
`m`	Length	Meter

The `Q1` and `Q2` values are input strictly in the basic measurement unit of C system, metric prefixes (kilo, mega, etc.) are not permissible. Measurement unit character's case doesn't matter, but it is recommended to strictly follow the writing from the table above.

```
"110000 V" // 110 kV
"12 Ohm"   // 12 Ohm
```

The `Expr` parameter of `Expression` special type sets the expression for calculation of a new signal. Rules for composition of expressions for signals calculating are described in section 13.

Variables, macros, and script arithmetic can be used in the `Expression`. All these entities are decomposed and calculated, and the result will be put in the insert point.

There is only one script feature that doesn't work in the `Expression` type frames - it is the concatenation of strings. Concatenation comes into conflict with the addition of signals in `Expressions`, which has the priority in this case.

EXAMPLE

Let's assume that three-phase currents of two breakers "Iq1_a", "Iq1_b", "Iq1_c" and "Iq2_a", "Iq2_b", "Iq2_c" exist in the oscillogram. It is necessary to calculate three-phase line currents as a sum of the corresponding phase currents of breakers with constant component that depends on the phase, and create the corresponding signals "I_a", "I_b", "I_c". Variables A and B included in the definition of constant component are assumed to be previously calculated by a relevant algorithm before the discussed code fragment.

```
// Declare phase names
let ph = ["a", "b", "c"];

// Declare name composition macros of breaker’s current
define mkQName(QNum, phase) = "Iq" + QNum + "_" + ph[phase];

// Declare name composition macros of line’s current
define mkLName(phase) = "I_" + ph[phase];

// Declare macros for calculation of constant component
define mkOffset(phase) = (A[phase] + B[phase] * 2);

// Loop for creation of three-phase currents
loop(i, 3)
{
    // Create loop local variables with assembled current names
    // of breakers of the actual phase, because the assembling of them right
    // in the Expression is impossible due to the inhibit of string
    // concatenation
    let I1 = mkQName(1, i);
    let I2 = mkQName(2, i);

    // Create line current signal of actual phase
    Sig.Create(
        Name = mkLName(i),
        Expr = I1 + I2 + mkOffset(i)
    )
}
```

Note that macros (even single used) make the code cleaner and easier to understand. With expressions instead of macros in the call-up points, they would look completely different.

Consider what conversions will be made to loop body code after the macros expansion (for i = 0):

```
{
    let I1 = "Iq" + 1 + "_" + ph[0];
    let I2 = "Iq" + 2 + "_" + ph[0];

    Sig.Create(
        Name = "I_" + ph[0],
        Expr = I1 + I2 + (A[0] + B[0] * 2)
    )
}
```

Let us assume that variables `A` and `B` by the time of execution of our loop, will have the following values (this code is not correct, for demonstration purposes only):

```
A = [10, 20, 30]
B = [40, 50, 60]
```

During the calculation, variables `I1` and `I2` and the used parameters of the `Sig.Create` command will be assigned with the following values (this code is not correct, for demonstration purposes only):

```
I1 = "Iq1_a"
I2 = "Iq2_a"
```

```
Sig.Create.Name = "I_a"
Sig.Create.Expr = "Iq1_a" + "Iq2_a" + 90
```

As we see, calculation of the constant component (90) for the `Expr` parameter was made by the script runtime system and was transmitted to signal calculation system in a completed form. Meanwhile, the concatenation of signal names was not executed.

14.13.2 Signal properties editing: Sig.Edit

Edit of existing signal properties:

Parameter	Type	Description	If absent
`Signal`	`String`	Signal name	
`Name`	`String`	New signal name	Not edited
`Q1`	`String`	see `Sig.Create`	Not edited
`Q2`	`String`	See `Sig.Create`	Not edited
`NonPeriodic`	`Bool`	see `Sig.Create`	Not edited
`Color`	`Int`	see `Sig.Create`	Not edited
`Dashed`	`Bool`	see `Sig.Create`	Not edited
`Invert`	`Bool`	see `Sig.Create`	Not edited
`Noise`	`Bool`	see `Sig.Create`	Not edited

14.13.3 Signal duplication: Sig.Dup

Signal's duplication.

Parameter	Type	Description	If absent
`Signal`	`String`	Name of signal that has to be duplicated	
`Name`	`String`	New signal name	Generated automatically

14.13.4 Signal hiding: Sig.Hide(...)

Command family designed to hide signals.
The **`Sig.Hide`** command is designed to hide the set list of signals and has only one parameter:

Parameter	Type	Description
`List`	`String[]`	List of hidden signal names

Other commands of this family do not have parameters and hide signals that conform to different conditions:

– **`Sig.Hide.All`** – all signals;

– **`Sig.Hide.Analog`** – all analog (and complex) signals;

– **`Sig.Hide.Binary`** – all binary signals;

– **`Sig.Hide.Calc.All`** – all calculated signals;

– **`Sig.Hide.Calc.Analog`** – all calculated analog (and complex) signals;

– **`Sig.Hide.Calc.Binary`** – all calculated binary signals;

– **`Sig.Hide.Const`** – all signals, with constant value throughout the oscillogram;

– **`Sig.Hide.Noise`** – all signals, with checked “Signal contains noise” box;

– **`Sig.Hide.Zero`** – all signals, with zero value throughout the oscillogram.

14.13.5 SIGNAL INVERSION: SIG.INVERT

Signal inversion.

Parameter	Type	Description
`List`	`String[]`	List of inverted signal names

14.13.6 SIGNAL REMOVING: SIG.REMOVE(…)

Command family designed to remove signals.
The **`Sig.Remove`** command is designed to remove the set list of signals and has only one parameter:

Parameter	Type	Description
`List`	`String[]`	List of removed signal names

> Note that some signals cannot be removed due to some other calculated signals depending on them, which will not be removed together with it. If a signal cannot be removed, the command just skips it.

Other commands of this family do not have parameters and remove signals that conform to different conditions:

– **`Sig.Remove.Calc`** – all calculated signals.

14.13.7 SIGNAL SHOWING: SIG.SHOW(…)

Command family designed to show hidden signals.
The **`Sig.Show`** command is designed to show the set list of signals and has only one parameter:

Parameter	Type	Description
`List`	`String[]`	List of names of signals to be shown

Other commands of this family do not have parameters and show signals that conform to different conditions:

– **`Sig.Show.All`** – all signals.

14.14 TRI CATEGORY COMMANDS

14.14.1 CREATION OF A THREE-PHASE CIRCUIT: TRI.CREATE

Creation of a three-phase circuit.

Parameter	Type	Description	If absent
`Name`	`String`	Name of the circuit	
`Phase`	`Bool`	(True) phase or (False) phase-to-phase circuit	`True`
`S0`	`String`	Signal name A or AB	1)
`S1`	`String`	Signal name B or BC	1)
`S2`	`String`	Signal name C or CA	1)

1) – If Phase = TRUE, all 3 phase signals must be set, if Phase = FALSE, at least 2 phase-to-phase signals must be set.

14.14.2 EDITING OF THREE-PHASE CIRCUIT: TRI.EDIT

Editing of a three-phase circuit.

Parameter	Type	Description	If absent
`Triad`	`String`	Edited circuit name	
`Name`	`String`	New circuit name	Not edited
`S0`	`String`	see `Tri.Create`	Not edited
`S1`	`String`	see `Tri.Create`	Not edited
`S2`	`String`	see `Tri.Create`	Not edited

14.14.3 HIDING SIGNALS OF THREE-PHASE CIRCUITS: TRI.HIDE

Hiding signals of three-phase circuits.

Параметр	Тип	Описание
`List`	`String[]`	List of names of three-phase circuits

14.14.4 REMOVING OF THREE-PHASE CIRCUITS: TRI.REMOVE(…)

Command family designed to remove three-phase circuits.
The **`Tri.Remove`** command is designed to remove the set list of three-phase circuits and has only one parameter:

Parameter	Type	Description
`List`	`String[]`	List of names of removed three-phase circuits

Other commands of this family do not have parameters and remove circuits that conform to different conditions:

– **`Sig.Remove.All`** – all three-phase circuits.

14.14.5 SHOWING SIGNALS OF THREE-PHASE CIRCUITS: TRI.SHOW

Showing signals of three-phase circuits.

Параметр	Тип	Описание
`List`	`String[]`	List of names of three-phase circuits

14.15 OSC CATEGORY COMMANDS

14.15.1 TOOLBAR SETTING: OSC.SETUP

Adjusts the switching control elements on the oscillogram toolbar.

Parameter	Type	Adjusts	If absent
`Val`	`Int`	**List:** Displayed analog signal values	Not edited
`Zoom`	`Int`	**List:** Analog signal scaling mode	Not edited
`Sec`	`Bool`	**Switch**: P/S. If value is False – P has to be chosen, if True – S.	Not edited

`Val` and `Zoom` parameters have **int** type and change the index of chosen element in relevant lists. Thereby, values possible for each of them are limited by the list length. To ease the setting of values, script has named constants for every index in both lists.

Possible `Val` values:

Constant	Value	Description
`Osc.Val.RMS`	`0`	Effective value
`Osc.Val.Inst`	1	Instantaneous value
`Osc.Val.Mean`	2	Mean value
`Osc.Val.H1`	3	1st harmonic
`Osc.Val.H2`	4	2nd harmonic
`Osc.Val.H3`	5	3rd harmonic
`Osc.Val.H5`	6	5th harmonic

Possible `Zoom` values:

Constant	Value	Description
`Osc.Zoom.Ind`	`0`	Individually
`Osc.Zoom.Msr`	`1`	By unit
`Osc.Zoom.Tri1`	`2`	By group (by unit)
`Osc.Zoom.Tri2`	`3`	By group (individually)

14.15.2 SIGNAL MOVING: OSC.MOVE(...)

Command family designed for moving signals on the oscillogram pane. All commands of this family have the same parameter list.

Parameter	Type	Description
`Dst`	`String`	Name of the signal relative to which the moved signals will be placed
`List`	`String[]`	List of names of moved signals

Ending in the command name defines the principle of positioning of `List` signals relative to `Dst` signal:

– **`Osc.Move.Before`** - on separate strips **above** the `Dst` signal strip;

– **`Osc.Move.After`** - on separate strips **below** the `Dst` signal strip;

– **`Osc.Move.Into.Before`** - on the `Dst` signal strip **above** it;

– **`Osc.Move.Into.After`** - on the `Dst` signal strip **below** it.

15 Conversion aspects of old scripts

Section 14 mentioned the command *Actions > Convert old script to new* in the script editor menu, that allows converting the script written for *Waves 3* into the *Waves 4* script.

This process always generates a syntactically correct script, but it is not always executable without any adjustments.

The point is that the subsystem of signal calculation in *Waves 3* performed a number of automatic conversions of signal types used in expressions (`Expr` parameter of `Sig.Create` command).

In *Waves 4* we abandoned this practice, because as a result it did more harm than good.

Below are all automatic conversions that applied in *Waves 3* with indicated conversions-wrappers that will allow imitating them in *Waves 4*.

Waves 3 allowed to *multiply* and *divide* `ComplexSignal` type signal to `FloatSignal` type signal, while the latter was automatically converted to the `ComplexSignal` type using the following conversion:

```
ComplexP(FloatSignal, 0)
```

Waves 3 allowed transferring `FloatSignal` type signal to the parameters of `ComplexSignal` type function, while `FloatSignal` was automatically converted to the `ComplexSignal` type using the following conversion:

```
Complex(FloatSignal)
```

Waves 3 allowed transferring `FloatSignal` type signal to the parameters of `BinarySignal` type function, while `FloatSignal` was automatically converted to the `BinarySignal` type using the following conversion:

```
Binary(FloatSignal)
```

Therefore, it is sufficient to add the described above conversions-wrappers to the right places to make the script operable.

16 ETC CONVERTER

The **etc** utility is supplied together with *Waves*. It is designed for conversion of oscillograms in EKRA formats into Comtrade files using a command line:

```
etc "<Path>" [-o "<Dir>"] [-r 1991|1999|2013] [-a] [-c] [-e 0|1|2]
```

Parameter	Description	If absent
`<Path>`	Path to the source oscillogram file in EKRA format	
`-o`	Directory (`<Dir>`), where an object file in Comtrade format will be formed	1)
`-r`	Standard revision year **1991, 1999, 2013**	1999
`-a`	Generate DAT section in ASCII format	BINARY
`-c`2)	Generate CFF file	CFG + DAT
`-e`3)	CFG section encoding **WIN(0), DOS(1), UTF8(2)**	0

1) – If parameter is absent, file is formed in the source file directory
2) – Ignored, if parameter**–r** has value different from **2013**
3) – Value **2** is ignored, if parameter **–r** has value different from **2013**

The object file has the name of the source file with Comtrade extension added:

```
etc 390D5021.zfr            --> 390D5021.zfr.cfg / 390D5021.zfr.dat
etc 390D5021.zfr –r 2013 –c --> 390D5021.zfr.cff
```

Oscillograms in EKRA formats can consist of several fragments. If all fragments have standard names and are in the same directory, **etc** will automatically compile the oscillogram from them and only after this will perform conversion. So it is unnecessary to call **etc** for all fragments by turn, it is enough to do this only for one of them (any).

Launching **etc** without command line displays a help on parameters.

Installation version of **Waves** sets its own installation directory into *PATH* environment variable, so the **etc** utility call-up is possible without indication of the full path to it.

17 WORKING WITH LINUX OPERATING SYSTEMS

Before installing *Waves* to *Linux* it is necessary to:

– Open documentation of your *Linux* distribution. Some of actions described below are performed differently depending on the used distribution, and may differ even in the versions of the same distribution. Actions that depend on the used distribution will be not provided in this document. Instead, we will recommend to refer to your documentation or manufacturer's tech support of your *Linux* distribution.

– Learn how to launch *Terminal* on your *Linux* distribution. *Linux* Terminal is an application software that allows inputting commands to the operating system.

17.1 INSTALLATION OF WINE

Installation and using of *Waves* on *Linux*-based OS is possible only through the *Wine* package. This package is used for launching on *Linux* of software that was written for *Microsoft Windows*. Wine emulates the *Windows API* program interface and *Windows* environment, so applications launched under it in *Linux*, “think” that they were launched in *Windows.*

Information on Wine installation see in the documentation of your *Linux* distribution.

Wine does not emulate *Windows* totally, so in some aspects that will be mentioned later, *Waves* will behave differently from its usual behavior in *Windows*. *Windows API* support is gradually increasing, and probably in newer Wine versions some of the features mentioned below will be eliminated. That is why it is important to use the latest *Wine* version and regularly update it.

17.1.1 WINDOWS ENVIRONMENT

After installation of Wine, a directory `$HOME/.wine` appears in the user home directory. It consists of the directories:

– `/drive_c` – has the recreated usual folder structure of *Windows* system disk.

– `/dosdevices` – has the symbolic links, which represent virtual discs visible for applications working under *Wine*. Drive C:\ - refers to the `/drive_c` directory mentioned above. Drive `Z:\` - refers to the root of *Linux* file system. Upon connection of external drives via *USB* interface, new virtual disks will be added here to provide access to the content of these drives from applications under *Wine*.

17.1.2 DISABLE WINDOW DECORATIONS

By default, *Wine* decorates the window frame of an app under it, making it visually identical to windows of applications running natively under *Linux.* This can lead to various artifacts and unexpected behavior of applications, so it is necessary to disable this *Wine* function at the beginning.

1. Launch *Windows Registry Editor*, by executing this command in *Terminal*:

```
wine regedit
```

2. Create a node `HKEY_CURRENT_USER\Software\Wine\X11 Driver`
3. Create a *string* parameter `Decorated` with value **`N`**.

17.2 Installation/unpacking of Waves

Start the *Waves* setup file/portable archive from the file manager of your *Linux* distribution and install/unpack it in the usual way.

17.2.1 Waves shortcut

In *Windows* we are used to that after installation, the *Waves* quick launch shortcut is created in the **Start** menu. Despite the fact that almost all modern *Linux* distributions have something similar to *Windows* **Start** menu, quick start shortcut is not created there during *Waves* installation, and has to be created manually if needed.

> Information on shortcut creation see in the documentation of your *Linux* distribution.

17.3 Differences in behavior

Overall, from user's perspective, *Waves* in *Linux* retains full functionality with one exception. The docking of floating analysis tools to the main window doesn't work, due to the absence of necessary mechanisms for it in *Wine*.