

WLD CAD

User Guide

Copyright © Sergey L. Gladkiy

Email: lrndlrnd@mail.ru

URL: www.Sergey-L-Gladkiy.narod.ru

Contents

Introduction.....	3
User Interface.....	4
Drag-Drop-Action interface	4
General Actions	5
Working with Well Logging Data.....	5
Loading and saving data	5
Displaying data graphs	6
Editing data	6
Working with Calculator.....	6
Basic data processing algorithm	7
Formula syntax	7
Building formula	9
Checking formula syntax	9
Calculating results	10
Using functions	10
Using variables	10
Using results	11
Using Well Logging Data in calculations	11
Examples of Well Logging data processing.....	12
Calculation of 3D Accelerometer orientation	12
Converting data values between Units of measurement	14
Data depth axis management	15
Appendix I. Function reference.....	18
Table I.1. Curve specific functions.....	18

Introduction

WLD CAD (Well Logging Data Computer Aided Design) program is a simple and powerful system of well logging data processing. The main goal of the program is to provide almost any operations with well logging data. WLD CAD program is like 'engineering calculator', but it works not with simple real numbers, it makes operations with well logging data. The system supports tens predefined functions and many operators for well logging data. Using these functions and operators allows building very complicated formulae for data processing.

WLD CAD is not intended to create complicated and structured borehole views and reports so it does not provide customizable well logging data viewer. The program provide simple viewer just to see what input data is using for calculations and what result is got by calculation.

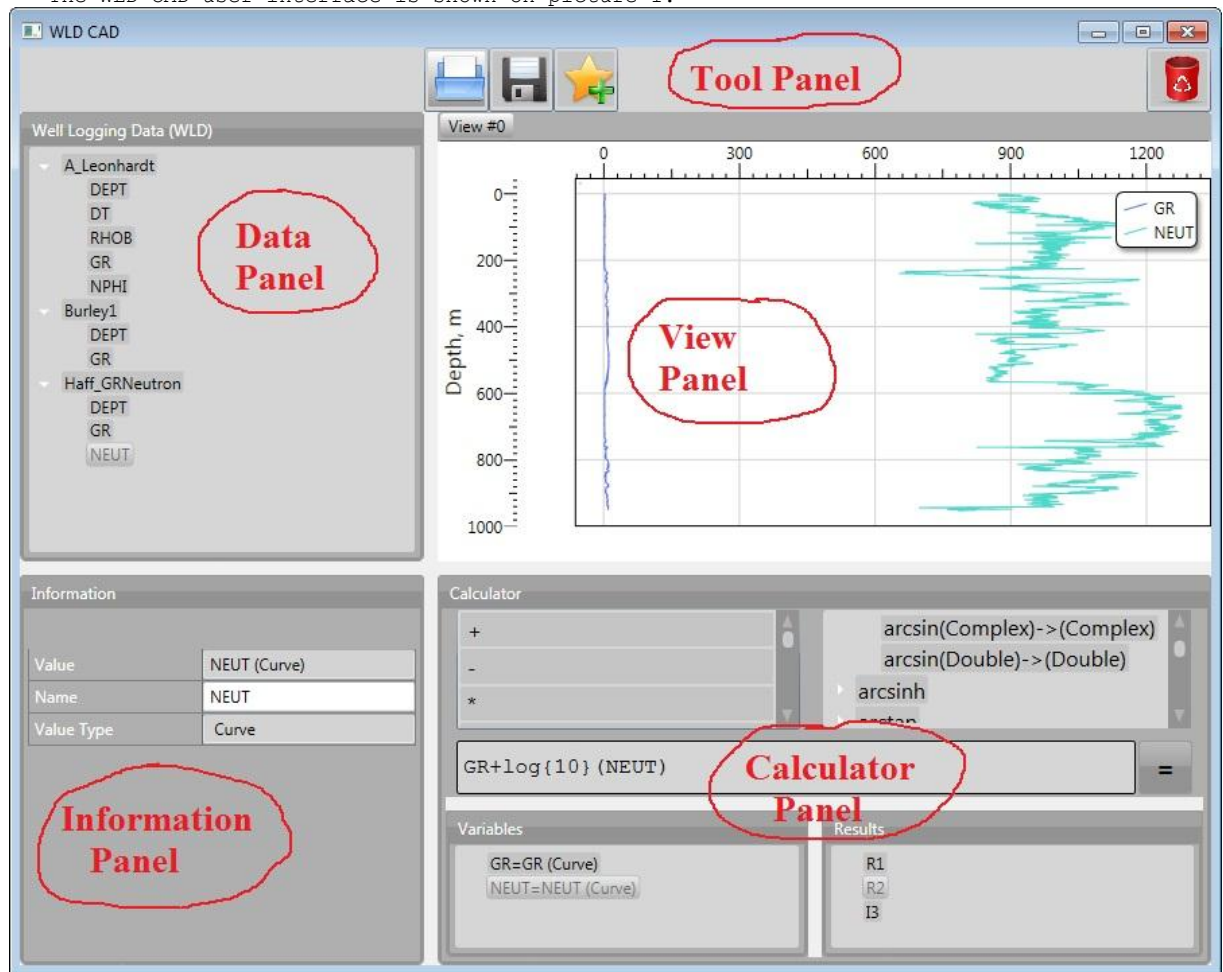
WLD CAD does not include predefined well known algorithms of well logging data processing (like cement bond, borehole volume calculation, mechanical properties calculation and so on). The program allows user to input arbitrary formula for data calculation and get the result by some unique algorithm.

In accordance with said above, WLD CAD program can be effectively used for the following purposes:

1. Fast testing formulae and algorithms for developing new well logging devices on the design stage (without programming and writing complicated code).
2. Fast testing formulae for new well logging data processing algorithms before code them and include in some software (for well logging data processing software developers).
3. Processing data with nonstandard algorithms which are not provided by common well logging processing software but can be presented by some mathematical formula (for geophysicists and well logging data interpreters).

User Interface

The WLD CAD user interface is shown on picture 1.



Picture 1. Program user interface.

The user interface consists of five main panels:

1. Tool Panel - contains tool buttons for user actions (loading data, saving data and so on).
2. Data Panel - shows Well Logging Data tree, currently accessible for operations (user actions, calculations and so on).
3. Information Panel - shows information about selected object.
4. View Panel - allows viewing well logging data as a graph along the depth axis.
5. Calculator Panel - special panel, containing tools to input formula, perform calculations with well logging data, manipulate with variables and calculation results.

Drag-Drop-Action interface

The WLD CAD user interface is mainly based on Drag-Drop-Action (DDA) principle. In short, this principle can be described as 'to initiate some **action** - **drag** something and **drop** it somewhere'. The action initiated depends on **what** was **dragged** and **where** it was **dropped**. For an example, to delete some object - drag it and drop into 'recycle' action button.

This interface have some advantages:





- Intuitive use, based on 'images', not 'text'. There is no need to read menu titles, button captions and so on. Just look at the 'image' and 'feel' what it is for.
- Fast access to the actions. There is no need to open innumerous submenus, there is no dialog for various actions, there is no need to confirm action with many 'ok' and 'next' button clicks.
- Compact interface. As many action types (delete, move, copy and so on) can be performed on objects of different types, there is no need to make different 'actions'. The action does the same, no matter the object type is. So, there is not a lot of user interface elements.

There are three cases, how the DDA works:

1. Some object is dropped into 'action' image. Then this action is performed on the object. As an example, if some object is dropped into 'recycle' image - it will be deleted.
2. Some 'action' was dropped into some panel area. Then this action is used for this panel. As an example, if 'new' action is dropped on the data panel - new empty well logging data will be created.

3. Some object from one panel was dropped into another panel area. Then the 'copy' or 'move' action is performed on the dropped object. As an example, if some well logging data object is dropped into view panel, it will be displayed.

When some object is dragging over some panel area, its image is displayed beside the mouse pointer, and also the drop action image is displayed. The drop actions are the following:

-  - 'Copy to'. The dragging object will be copied here or the dragging 'action' will be performed.
-  - 'Link'. The dragging object will be linked here or the dragging 'action' will be performed.
-  - 'Move to'. The dragging object will be moved here from origin.
-  - 'None'. The dragging object cannot be placed here or the dragging action cannot be performed.

General Actions

The WLD CAD tool panel contains the following 'action' buttons:



- 'Open' action.



- 'Save' action.



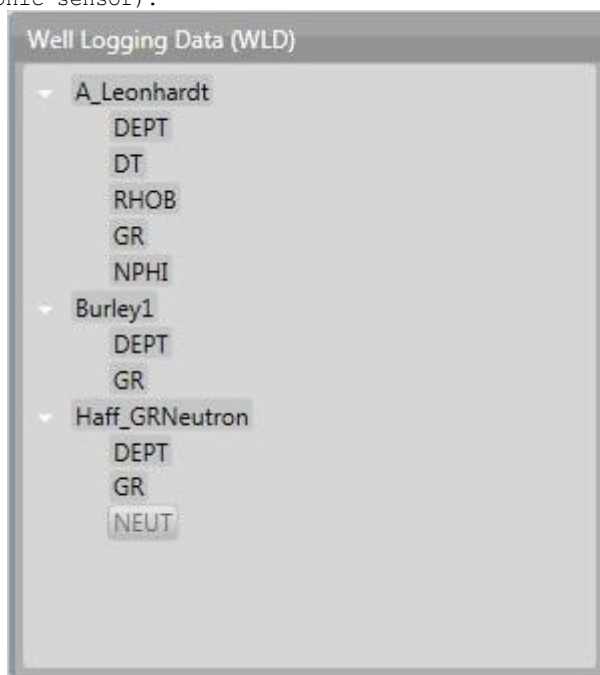
- 'New' action.



- 'Recycle' action.

Working with Well Logging Data

The well logging data is shown on the Data panel as a data tree (picture 2). The most outer tree nodes represent well logs - entire data record, written by a logging device during one borehole measurement. The inner tree nodes represent log items (curves) - data record got from one of the device sensor (gamma ray sensor, sonic sensor).



Picture 2. Well Logging Data panel.

Loading and saving data

To load data from files drag 'load' action image and drop it into Data panel. The open file dialog will be shown. Many files can be selected simultaneously to load. If data loaded successfully - it will be added into data tree, else - the error will be shown.

To save data drag a log and drop it into 'save' action image. The save file dialog will be shown. Each log must be saved in separate file.

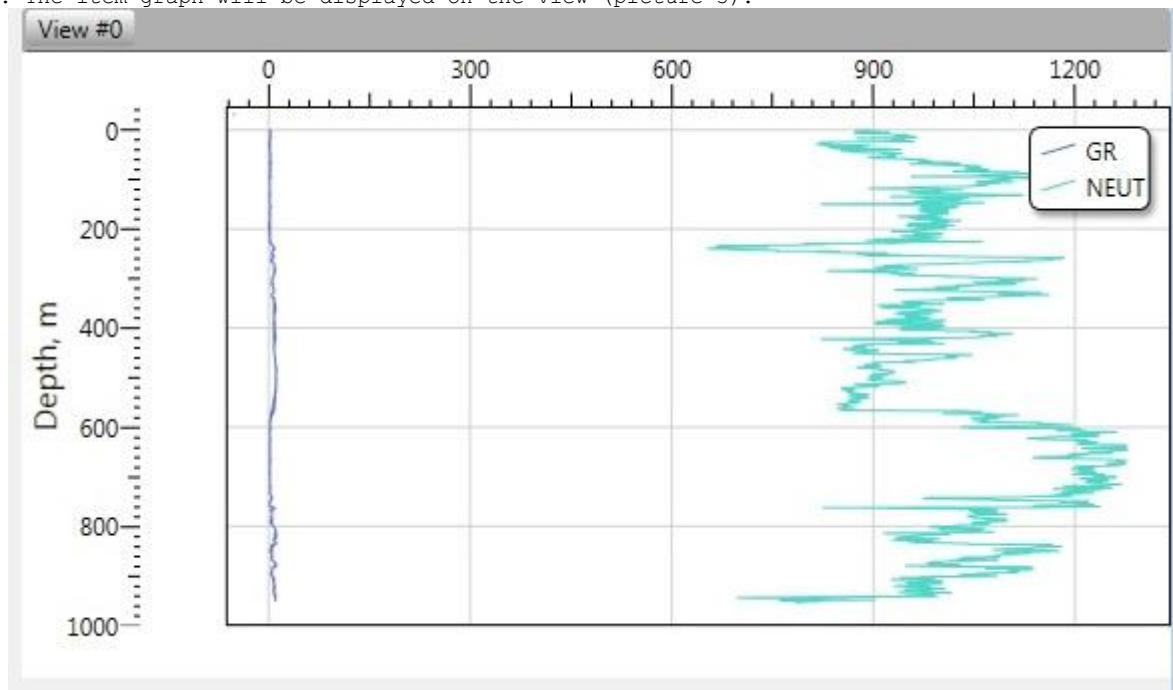
The following file formats are supported to load and save data:

- LAS (Log ASCII Standard);
- LIS (Log Information Standard);

- DLIS (Digital Log Interchange Standard).

Displaying data graphs

To display a log item data as a graph along the depth axis - drag it and drop into the view panel. The item graph will be displayed on the view (picture 3).



Picture 3. Item graph viewer.

All log items graphs are displayed in their own units of measurement along the horizontal axis. So it is recommended to place item graphs on the same view only if they have the same units of measurement. To create new view - right click on the view header and in the popup menu select 'New view'.

The view area can be easily customized. The following view management operations provided.

- Scroll along the Depth axis. To scroll along the Depth just scroll the Mouse Wheel Up or Down.
- 'Real time pan'. To pan view area just put right mouse button down inside the area and move mouse pointer.¹
- Horizontal Scale change. To change Horizontal axis scale press the 'Alt' button and scroll Mouse Wheel inside the graph area.
- Depth Scale change. To change the Depth axis scale just scroll Mouse wheel inside the Depth Axis area.
- 'Zoom window'. To zoom some view area just press 'Ctrl' key and select the area by mouse. The area will be brought into view.
- 'Fit view'. To fit all graphs into view - right click on the view and select 'Fit to view' in the popup menu.

Editing data

Editing data means make operations with existing data without processing it and creating new data (for processing data algorithm see below).

There are the following editing operations:

- **Move log item** from one log to another or inside the same log. To move log item - drag it and drop where it is wanted to be. NOTE: the log item can be moved only to the log with the same depth axis.
- **Delete log item**. To delete a log item - drag it and drop into 'recycle' action image.
- **Delete log**. To delete a log - drag it and drop into 'recycle' action image.

Working with Calculator

Calculator is the main data processor in the program. All data processing operations are performed with this tool.

The Calculator panel is shown on the picture 4. It consists of five sub panels:

1. 'Symbols' panel - contains the list of symbols used in formulae (see below about formula syntax).
2. 'Functions' panel - contains the list of functions, currently realized (see below for detail information about functions).

¹ By default the view area is panning along the Depth axis only, for panning along both axes simultaneously - press 'Alt' key.

3. 'Formula editor' panel - contains text editor to input the processing formula and the 'Calculate' button '='.
4. 'Variables' panel contains the list of variables available for using in formula (see below about variables).
5. 'Results' panel contains the list of calculated result values.



Picture 4. Calculator panel.

Basic data processing algorithm

The WLD CAD program is like 'engineering calculator', but it works not with simple real numbers and with well log items. It means that the main 'unit' of the calculation algorithm is well log item, not number. The simplest example is: if two well log items (curves), say 'x' and 'y' must be added, the formula is 'x+y'. The result of calculation is also well log item with the same depth axis and the values are the sum of corresponding 'x' and 'y' items.

In accordance with said above, the common data processing algorithm is:

- Load input log data.
- Write the formula (in the form of correct mathematical expression) that must be calculated in accordance with processing algorithms.
- Calculate result value (commonly it is a log item) - output data.
- View the result item graph to see what the result is.
- Move the result to some data log.
- Save the data log with the result data.

The algorithm can be more complicated. The result data can be used as input value for other formula calculation, one formula can be calculated many times for different parameters (input data, variable values) and so on. So, the final realized processing algorithm can be as advanced as possible.

Formula syntax

The WLD CAD program uses formula (mathematical expression) to realize all data processing algorithms. So, effective using of the program supposes syntax knowledge to write correct formula.

Syntax is a set of rules to build structured and complicated expressions from simpler ones called **tokens** (or **lexemes**). There are the following tokens: **literals**, **variables**, **operators**, **functions**, **indexing**.

The following part explains the basics of formula syntax elements.

Mathematical expression is a sequence of elements for which its result value can be calculated. A simple example of expression is 'x+y' - knowing the values of x and y we can calculate their sum. Generally, expression contains such elements as **constants (literals)**, **variables**, **operators**, **functions**. The result of an expression calculation depends on elements the expression consists of. For an example if 'x' and 'y' are real numbers (variables) - the result of the given sum expression is a real number, if one of the values is a complex value - result is a complex value too.

Literal is an unnamed constant value or a symbol staying for 'standard' constant value. For an example, in the expression '2*(x+y)' - '2' is a numerical literal. **Real** and **Complex** literals are supported.

Real literals can be written in simple and exponential form. Simple form examples: '100', '1.23', '-45.67'. Exponent form examples: '1.23E-3', '-2.45e+5'.

For complex literals symbol 'I' is used to denote **imaginary unit**. Complex literal consists of Real part and Imaginary part separated by '+' or '-' symbols. Real part is just a real literal, imaginary part is real literal plus imaginary unit symbol (**NOTE**: there is no multiplication symbol between real literal and imaginary unit symbol). Complex literal examples: 'I', '-4I', '2+3.2I', '-2e2-4.45I', '2.45+I', '-1.2e-3+4.5e+2I'.

² Current system separator is used in real literals.

The following 'standard' constants are recognized: 'e' - Euler number, 'n', 'Pi' - Pi number, '∞' - positive infinity.

Variable is a named value (that can be changed during calculation). A variable name can include alpha-numeric symbols only and underscore symbol (the first symbol can be letter only). The value of a variable can be of any type. The variable value is accessed via variable name. That is in an expression the variable name stands for the current variable value. For an example, let the variable 'A' is a real variable whose current value is '1.0', then the result of 'A+1' expression is '2.0'. Changing variable values allows calculation of the same expression for various values.

Operator is a symbol representing some mathematical operation. For an example, operator '+' stands for the sum operation. The data values, the operation is made with, called **operands**.

All operators can be divided into many categories. The most common used are **unary** and **binary** operators. **Unary operators** have one operand and can be **prefix** (stands before its operand) and **postfix** (stands after its operand). An example of unary prefix operator is the **negation operator** ('-x', '-' is the operator, 'x' is the operand). An example of unary postfix operator is the **factorial operator** ('n!', '!' is the operator, 'n' is the operand). **Binary operators** have two operands and commonly stand between them. An example of binary operator is the **addition operator** ('x+y', '+' is the operator, 'x' and 'y' are the operands).

Each operator has such attributes as **precedence** and **associativity**. The **precedence** determines the order of expression calculation. The operators with higher precedence applied to their operands before the operators with lower precedence. For an example, in the expression 'x+y*z' the first operation performed is the multiplication of 'y' and 'z' and then the sum of 'x' and the product result is calculated, because the '*' operator is of higher precedence than the '+' is. The order of calculation can be changed by using parentheses '()'. The **associativity** determines how operators with the same precedence are grouped in expressions. Let us consider the expression '2^3^4' (where the '^' operator stands for the power). The result of the expression depends on how the expression is interpreted - '(2^3)^4=8^4' or '2^(3^4)=2^81'. **Left** associativity means that operators are grouped from left to right (the first case), **right** associativity means grouping from right to left (the second case).

The following operators are defined:

```
'+' addition operator (binary);
 '-' subtraction operator (binary);
 '*' multiplication operator (binary);
 '/' division operator (binary);
 '^' power operator (binary);
 '.' dot operator (binary);
 '-' negation operator (unary, prefix);
 '~' tilde operator (unary, prefix);
 '!' factorial operator (unary, postfix);
 "'" apostrophe operator (unary, postfix);
 '≡' identically equal operator (binary);
 '≈' approximately equal operator (binary);
 '≠' not equal operator (binary);
 '>' greater than operator (binary);
 '<' less than operator (binary);
 '≥' greater than or equal operator (binary);
 '≤' less than or equal operator (binary);
 '&' logical and operator (binary);
 '|' logical or operator (binary);
 '!' logical not operator (unary, prefix);
 '?' question operator (unary, prefix);
 '#' number operator (unary, prefix);
```

Also the following 'Special'³ operators defined:

```
'←' left arrow operator (binary);
 '→' right arrow operator (binary);
 '↑' up arrow operator (binary);
 '↓' down arrow operator (binary);
 '↔' left-right arrow operator (binary);
 '↕' up-down arrow operator (binary);
 '∂' derivative operator (unary, prefix);
 '∫' integral operator (unary, prefix);
 'Δ' delta operator (unary, prefix);
 'Σ' sum operator (unary, prefix);
 'Π' product operator (unary, prefix).
```

General rules for all operators:

- Binary algebraic operators (+, -, *, etc.) have precedence as determined with standard mathematical rules.
- Relational operators (binary) have lower precedence than algebraic ones.
- Binary Logical operators have lower precedence than relational ones.
- Arrow operators (binary) have higher precedence than power operator.
- Binary operators are all left-associative.
- Unary operators have higher precedence than binary operators.
- Postfix operators have higher precedence than prefix operators.

Function is a named operation with some data values called **arguments**. An example is the sine function 'sin(x)', where 'sin' is the name of the function and 'x' is the argument. The function name determines what operation is performed with the arguments.

³ 'Special' operators have no predefined meaning for common data types (real, complex and so on) and can be used for special needs with program specific types. For an example - '↑' operator is used in WLD CAD program to shift data depth axis up.

In addition to arguments, a function can have parameters. For an example, the logarithm of 'a' to base 'b' function $\log_b(a)$ has one argument 'a' and one parameter 'b'.

General rules for functions:

- function arguments enclosed in parantheses '()';
- function parameters enclosed in braces '{}';
- if the function has no parameter, parameter braces are not obligatory, argument parantheses are always obligatory;
- function can have many parameters and/or many arguments;
- function arguments and parameters are separated by spaces ' ';
- there can be many functions with the same name and different number and/or type of arguments and/or parameters.

Some examples of syntactically correct function expressions:

max(a b) - maximum function of two arguments;

random() - random function with no arguments;

log{b}(a) - logarithm of 'a' to base 'b' (one parameter 'b', one argument 'a');

P(n m)(x) - the Associated Legendre function (two parameters 'n' and 'm' and one argument 'x').

Indexing is a method of access to the structured data elements. An example of structured data is array. Each array's element has the unique index by which the element value is accessed. By syntax rules data indexes are written in square brackets '[]'. For an example, the i-th array element can be written as 'A[i]'. For multiple indexes, each index must be written in different brackets. For an example, a matrix element can be written as 'M[i][j]'. Such syntax allows **slicing** implementation. Slicing allows to get a part of structured data. Slicing is implemented via index omitting. Let we have a matrix (two-dimensional array) 'M'. Then 'M[i][]' is the i-th matrix' row and 'M[][j]' is the j-th matrix' column. The 'trail' indexes can be omitted with their brackets. Thus, the matrix' row can be written as 'M[i]' (which is equivalent to 'M[i][]' of the previous example).

General indexing rules:

- indexing can only be applied to variables, the variables must be structured data;
- many indexes allowed, each index must be enclosed in square braces '[]';
- all index expressions must return values of real type only, the values must be (almost) integer;
- some indexes can be omitted that means slicing, slicing can only be applied to variables.

Syntax summary:

- Any expression can contain literals, variables, operators, functions and indexing.
- Literals can be real numbers in simple and exponent form and complex numbers.
- Variable names can include alpha-numeric and underscore symbol only.
- Only predefined (see above) operators can be used.
- Operations are performed in order of operator precedence. Operations order can be changed using parantheses '()'.
- Functions have parameters and arguments. Parameters are enclosed in braces '{}', arguments are enclosed in parantheses '()'. Parameters and arguments are separated by spaces ' '.
- Indexing can only be applied to variables. Indexes are enclosed in brackets '[]' (each index in separate brackets). Slicing of indexed data can be implemented by omitting some indexes.

Some examples of syntactically correct formulae:

'sin(b)^2+cos(b)^2'	- Pythagorean trigonometric identity formula.
'sin(a)*cos(b)+cos(a)*sin(b)'	- Sine of angle sum formula.
'log{c}(a)+log{c}(b)'	- Logarithm of product formula.
'A[n]*r^n*sin(n*a)+B[n]*r^-n*cos(n*a)'	- Laplace's equation solution in polar coordinates.
'A[n]*sinh(n*Pi/L*(x-a))*sin(n*Pi/L*(y-b))'	- Laplace's equation solution in Cartesian system.

Building formula

All data processing algorithms in the program realized via mathematical formulae calculation. The formula must be written in the formula editor (picture 4). It can contain: literals, variable names, operators, functions, indexing. The formula expression is represented as a planar text, so it can be input by using keyboard. But operators, functions and variable names can contain special Unicode symbols those cannot be input from standard keyboard. There are some tools to help building formula expressions (picture 4).

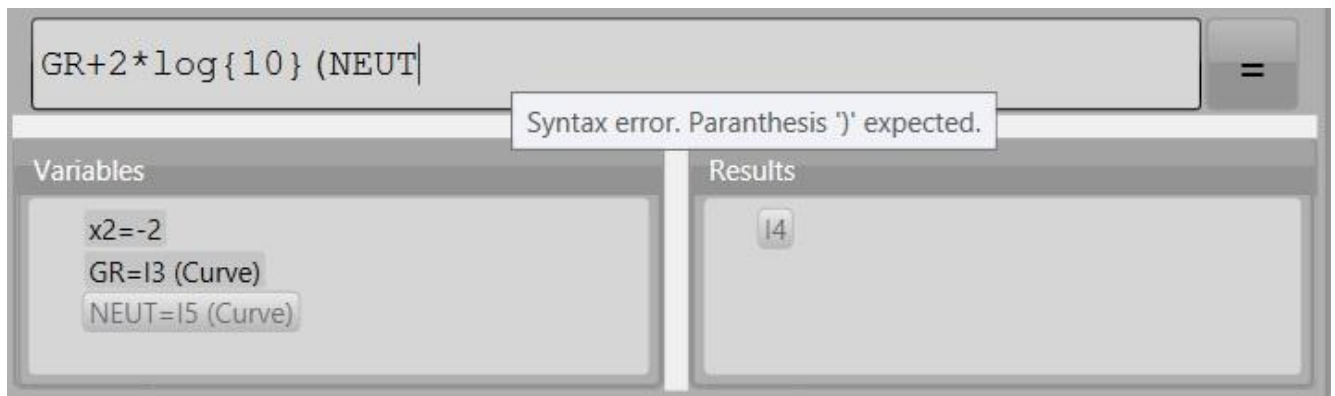
Symbols panel - contains list of symbols for using in formulae: operators, bracket pairs, Greek letters and so on. To paste some symbol in the formula expression - drag the symbol and drop it into formula editor.

Functions panel - contains function list, realized in the program. To insert some function in the formula - drag it and drop inside the formula editor.

Variables panel - contains list of all variables, those can be used in expressions. To use some variable in the formula - drag it and drop into formula editor.

Checking formula syntax

Each time the formula expression is changed, the syntax check is done automatically. If there is some syntax error - the error is shown on the tooltip of the formula editor (picture 5).



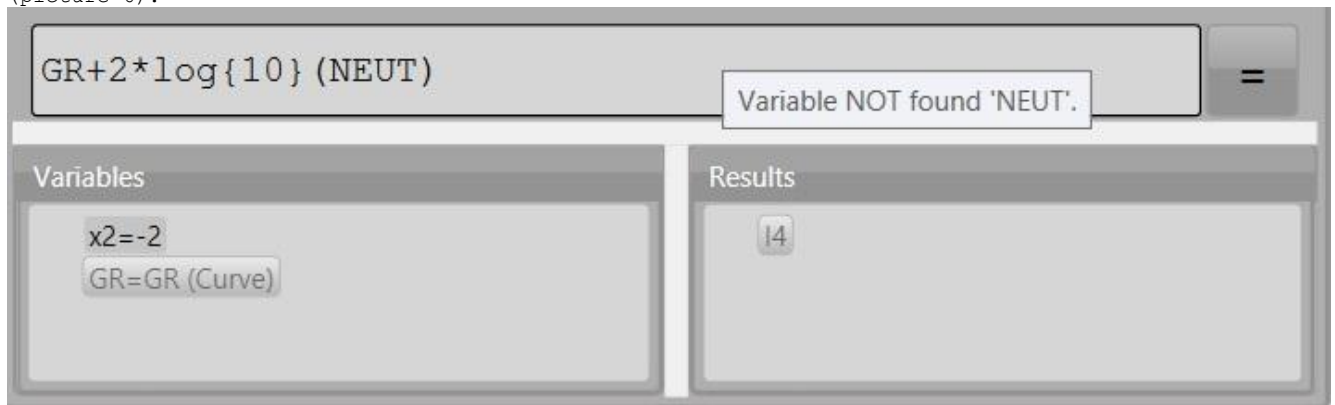
Picture 5. Syntax error tooltip.

On the picture above, the syntax error message 'Paranthesis \')' expected.' is shown. It means that the right parenthesis must be inserted into the formula to follow the syntax rule (all parentheses must be paired).

Calculating results

Press calculate button '=' (picture 4) to get the result for written formula. If the result is successfully calculated it is automatically added to the 'Results' panel (picture 4). To see the result graph - drag it and drop into 'View' panel.

If some error occurred during calculation it is displayed as the tooltip of the formula editor (picture 6).



Picture 6. Calculation error tooltip.

The error, shown on the picture above, means that there is no variable with the name 'NEUT' in the current variable list and then the formula result cannot be calculated.

Using functions

The 'Functions' panel (picture 4) contains the list of functions. The list is sorted by function name. There can be many functions with the same name and different count/type of arguments/parameters. To see all function variants - expand the function list. Each function description contains the function name and types of arguments, parameters and result. Consider the following examples of function descriptions:

1. arcsinh(Double)->(Double)
2. arcsinh(Complex)->(Complex)
3. log{Double}(Complex)->(Complex)

The first example description stays for the function with name 'arcsinh' (hyperbolic arcsine function) with one argument of type 'Double' (double precision real value) and the result of the function is of the same type. The second example shows the description of the 'arcsinh' function with one argument of 'Complex' type, the function result is also 'Complex'. And finally, the third example describes the logarithm function 'log' with one parameter of 'Double' type (the base of the logarithm), one argument of type 'Complex' and the result is also 'Complex'.

When some function is dropped into the formula editor, the function template inserted. For the third example, it is 'log{ }()'. The function template contains underscore symbols '_' staying for parameters and arguments.

Using variables

The 'Variable' panel (picture 4) contains the list of current variables. A variable can be of any type. To see information about some variable - select it in the list, the information will be shown on the 'Information' panel.

To create new variable - drag 'New' action image and drop it into 'Variable' panel. New variable of type 'Double'⁴ will be created and added to the variable list. Using 'Information' panel the **Name** and the **Value** of variable can be changed.

To delete some variable from the list - drag it and drop into 'Recycle' action image. The variable will be deleted.

If some variable is of type 'WLD Item' (curve)⁵ it can be displayed on the view panel. Drag the variable and drop it into some view area to display its graph.

The variable can be used in calculation formulae. All variables must be referenced in the formula by its name. To put a variable in formula expression - drag it and drop into the position it must be placed. The variable name can be written also using keyboard.

Using results

The 'Result' panel (picture 4) contains the list of calculated results. Results can be used in the next calculations, displayed on the 'View' panel, added to the WLD data list or deleted.

If some result is of type 'WLD item' it can be displayed on the 'View' panel - drag the result and drop it into some view area.

To use some result in the next calculations it firstly must be converted to the variable. To convert some result to the variable - drag it and drop into the 'Variable' panel. The result will be deleted and the variable of the same type will be added to the variable list. The name of variable can be changed and the variable can be used then in the calculations.

If some result is of type 'WLD item' and it is 'final' (is what the aim of calculation was) the item can be added to the WLD data and then saved. To move a result to the WLD tree - drag it and drop into the Log it must be placed.

To delete a result - drag it and drop into the 'Recycle' action image.

Using Well Logging Data in calculations

The Well Logging data (shown in the data tree, picture 1) can be used in calculations by two ways.

The first way is using data directly. Drag some WLD item from data tree and drop it into formula editor. The item's designation will be inserted into formula expression. The designation is the indexing (see formula syntax for more information about indexing). The name of indexed variable is 'WLD'. The first index is zero based index of the Log in the data tree. The second index is the zero based index of the item in the Log.

As an example, let's the current data tree is such as on the picture 2. Then the designation of the 'GR' item from the 'Haff_GRNeutron' Log is 'WLD[2][1]' because the log has the index 2 in the current three and the item has index 1 in the log data list.

The second way of using well logging data is to create variables for some data and then use them as described above. To create variable for the well logging items (curves) - drag the item and drop it into the 'Variable' panel. The variable will be created. The variable just reference the well log item data. It means that if the item is deleted - the variable will be deleted to, but not vice versa.

⁴ Real value of double precision.

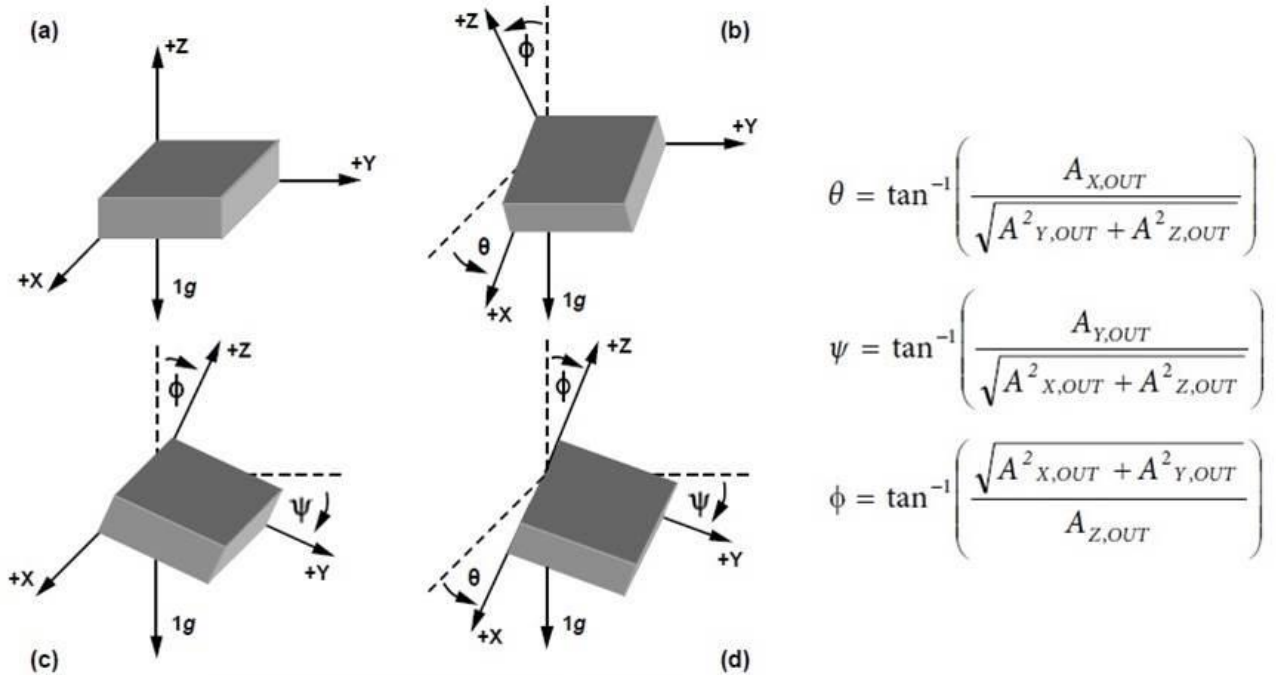
⁵ See below how to get variables of this type.

Examples of Well Logging data processing

This part contains the examples of well log data processing with the WLD CAD program. The example algorithms can be reproduced, the input data files can be found in the 'TestData' folder in the program's directory.

Calculation of 3D Accelerometer orientation

Let's there are data acquired by a well logging device and the data includes the accelerometer measured output. The goal of calculation (data processing) is to find the angles of accelerometer's inclination. The schematic accelerometer and its parameters shown on the picture 7.



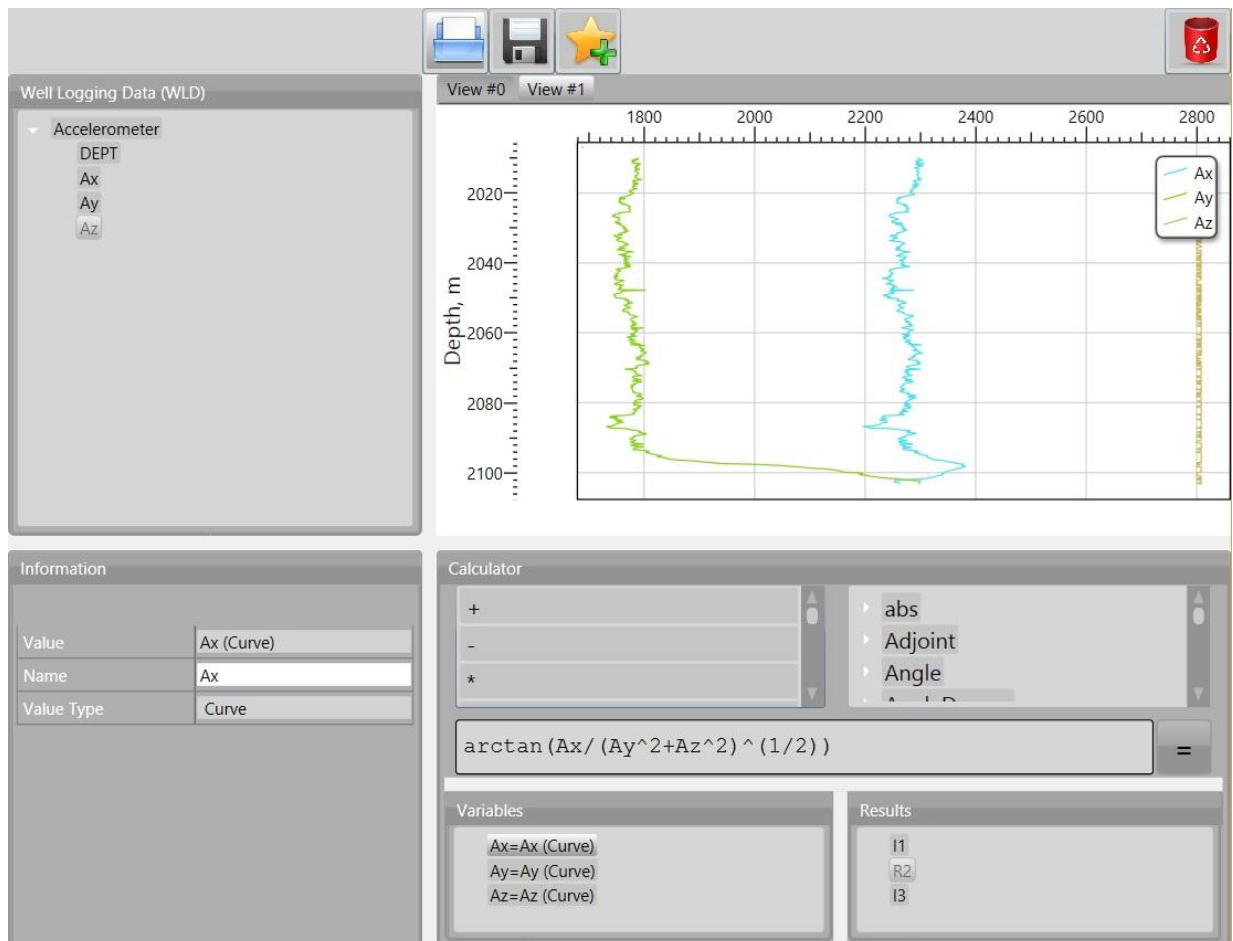
Picture 7. Accelerometer data schema and formulae.

To perform required data processing and get the angles of declination, the following steps can be used. First of all, the input data must be loaded (test data for this example can be found in the 'Accelerometer.las' file). Drag the 'Open' action and drop it into the 'Data' panel. The data will be loaded from the file (picture 8). Then, the variables for A_x , A_y and A_z data must be created. Drag the ' A_x ' item from data tree and drop it into the 'Variable' panel. Do the same for the ' A_y ' and ' A_z ' items (picture 8).

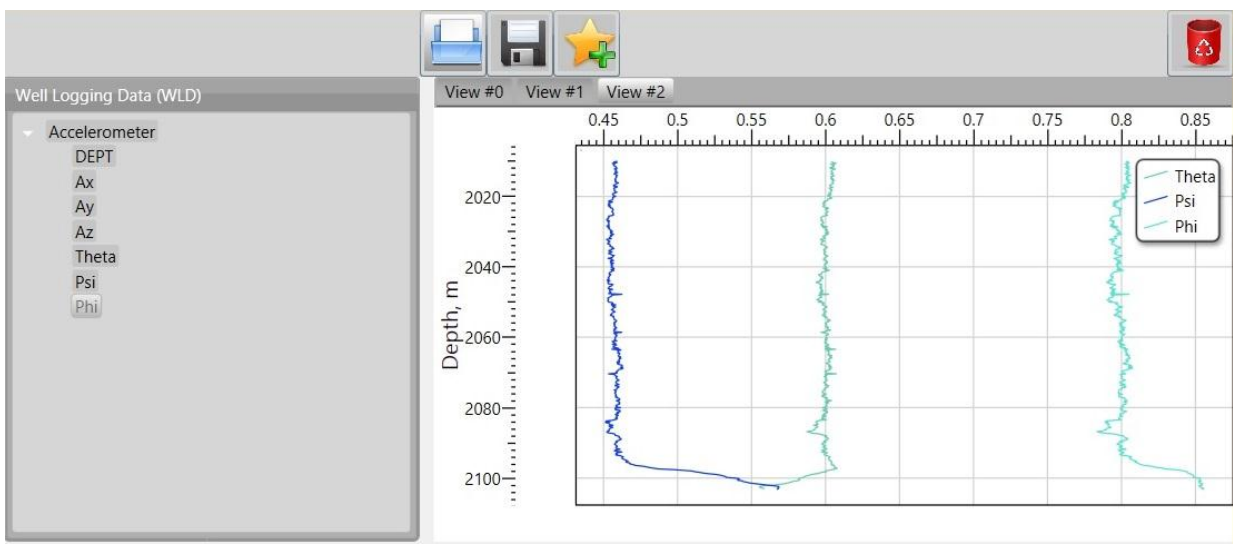
Input formula for θ angle (picture 8) and press '=' button. The result will be added to the 'Result' panel. Select the result and rename it as 'theta'. Input formula for ψ and ϕ angle, calculate results and rename them as 'Psi' and 'Phi'.

Drag the 'Theta' result and drop it into the 'Accelerometer' log. Do the same actions for the 'Psi' and 'Phi' results. The calculated data will be moved to the log data list. Drag new items and drop them into some view area - the result graphs will be shown (picture 9).

And finally, drag the 'Accelerometer' log and drop it on the 'Save' action image. Save the data into some new file.



Picture 8. Accelerometer orientation calculation.



Picture 9. Accelerometer orientation calculation results.

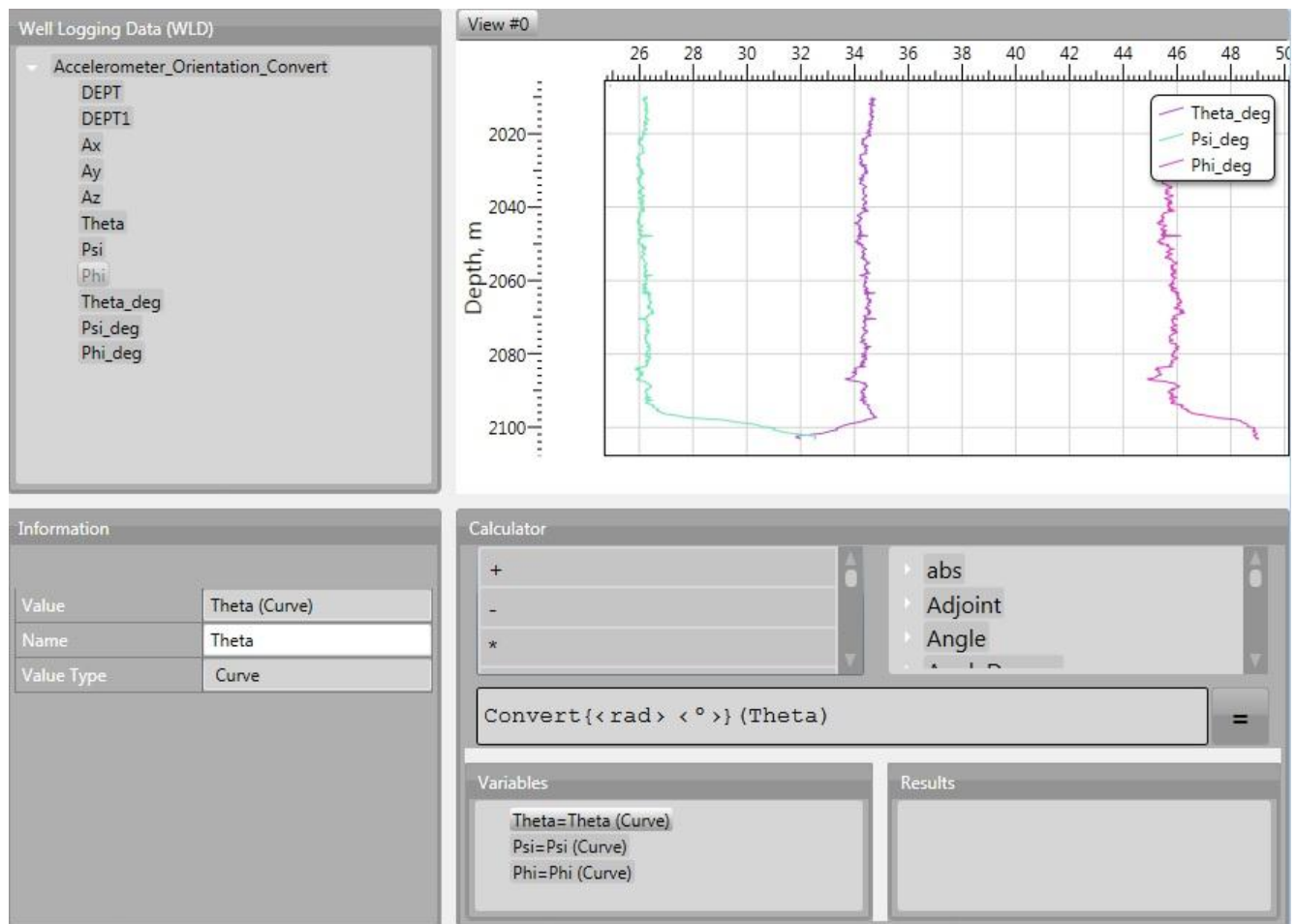
Converting data values between Units of measurement

The WLD CAD program provides functions to convert data from one unit of measurement to other units. The library contains many predefined units of measurement for various physical quantities.⁶

Consider the results of calculation for previous example. All angles of the accelerometer's inclination were calculated in radians. It is rather difficult to for human perception. So it would be better converting result data to degrees.

Let's the results of previous calculation exist (picture 8). Make variables for well logging items 'Theta', 'Psi' and 'Phi', as explained above. Then input the following formula in the editor '**Convert{<rad> <°>} (Theta)**', where 'Convert' is the name of function. The function has two parameters - unit to convert from and unit to convert to, and one argument - data for conversion. All units of measurement in the formula must be enclosed by triangle braces '<>'. The braces and 'degree' symbol can be found on the 'Symbol' panel.

Press '=' button to calculate result. Make the calculation for 'Psi' and 'Phi' variables. Move three calculated results to the 'Accelerometer' log and rename them as 'Theta_rad', 'Psi_rad' and 'Phi_rad'. Drop them into some view area and see the results (picture 10). Now they are expressed in degrees.



Picture 10. Accelerometer orientation data converted to degrees.

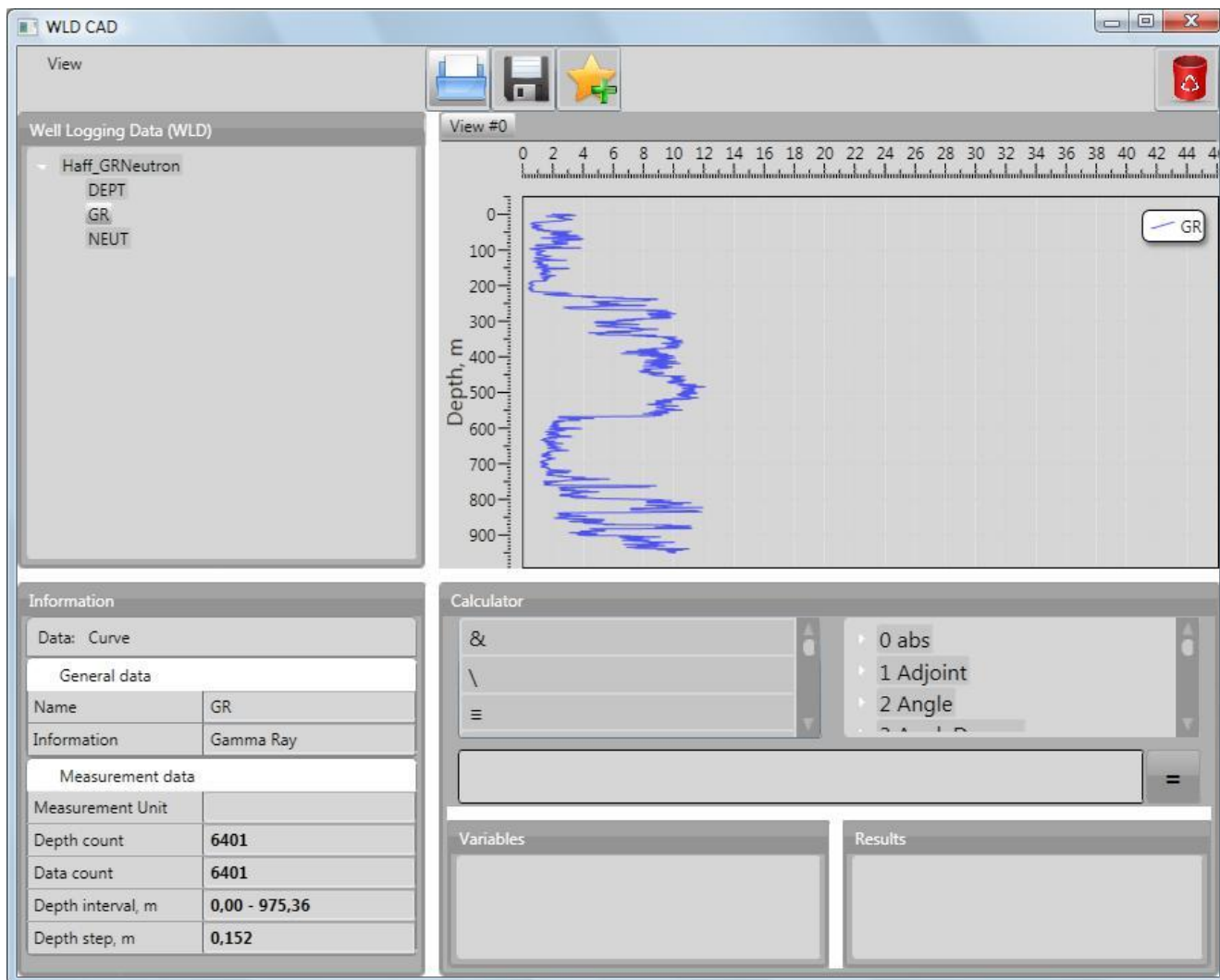
⁶ See guide of 'Physics' library for total information about units of measurement.

⁷ See guide of 'Analytics' library for syntax of using units of measurements in formulae.

Data depth axis management

The WLD CAD program contains special functions to process data depths - shift, scale, getting data in range and so on.

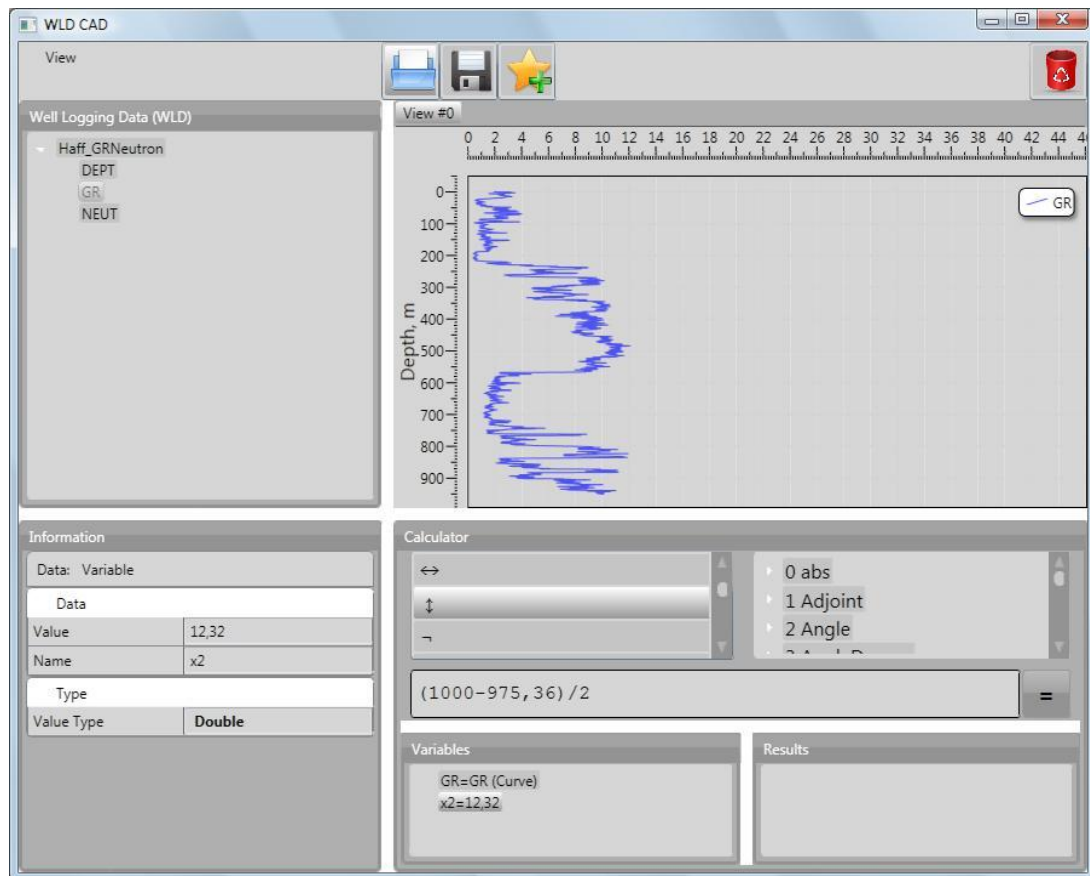
Consider the following example. Let the data loaded in the program from file (the example data can be found in TestData folder - Haff_GRNeutron.las). The initial data depth range is 0 - 975.36 meters (picture 11). Let it is known that the real depth range is 0 - 1000 meters (the wrong bottom value was due to device error). So, the depth axis must be change to get right depth range.



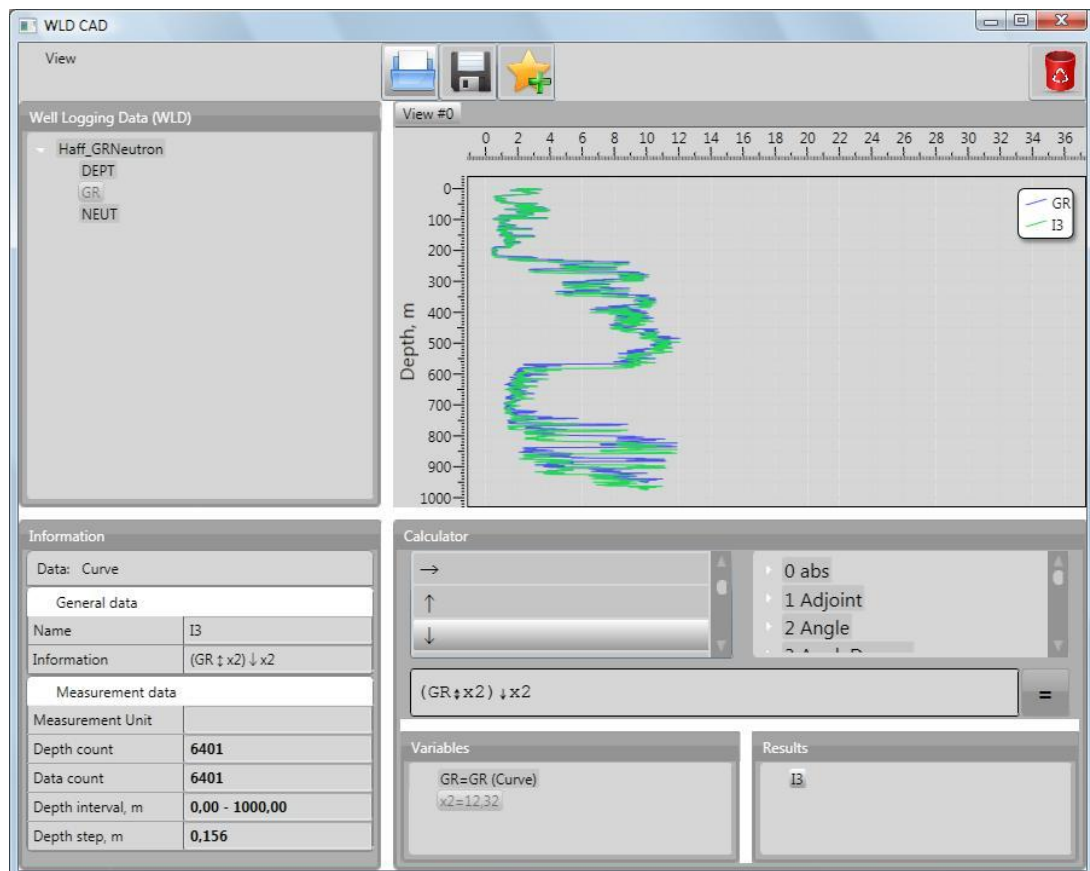
Picture 11. Initial data.

First of all, we must calculate the difference between the right and wrong bottom depths. Input the following formula into the Formula Editor as shown on the picture 12. Press '=' button - the shift value is calculated. Drag it into the Variable panel - the variable is created, say 'x2' (picture 12). When the shift value is calculated, the depth axis can be easily changed. Input the formula shown on the picture 13 and press '=' - the result, say 'I3', is calculated and shown on the same picture. As can be seen from the picture, the result's depth range is now 0 - 1000 meters. The operators '↑' and '↓' used to manage depth axis range. The '↑' operator stretched the depth range up and down by 'x2' value and then the '↓' operator shifted the depth axis down by the 'x2' value.

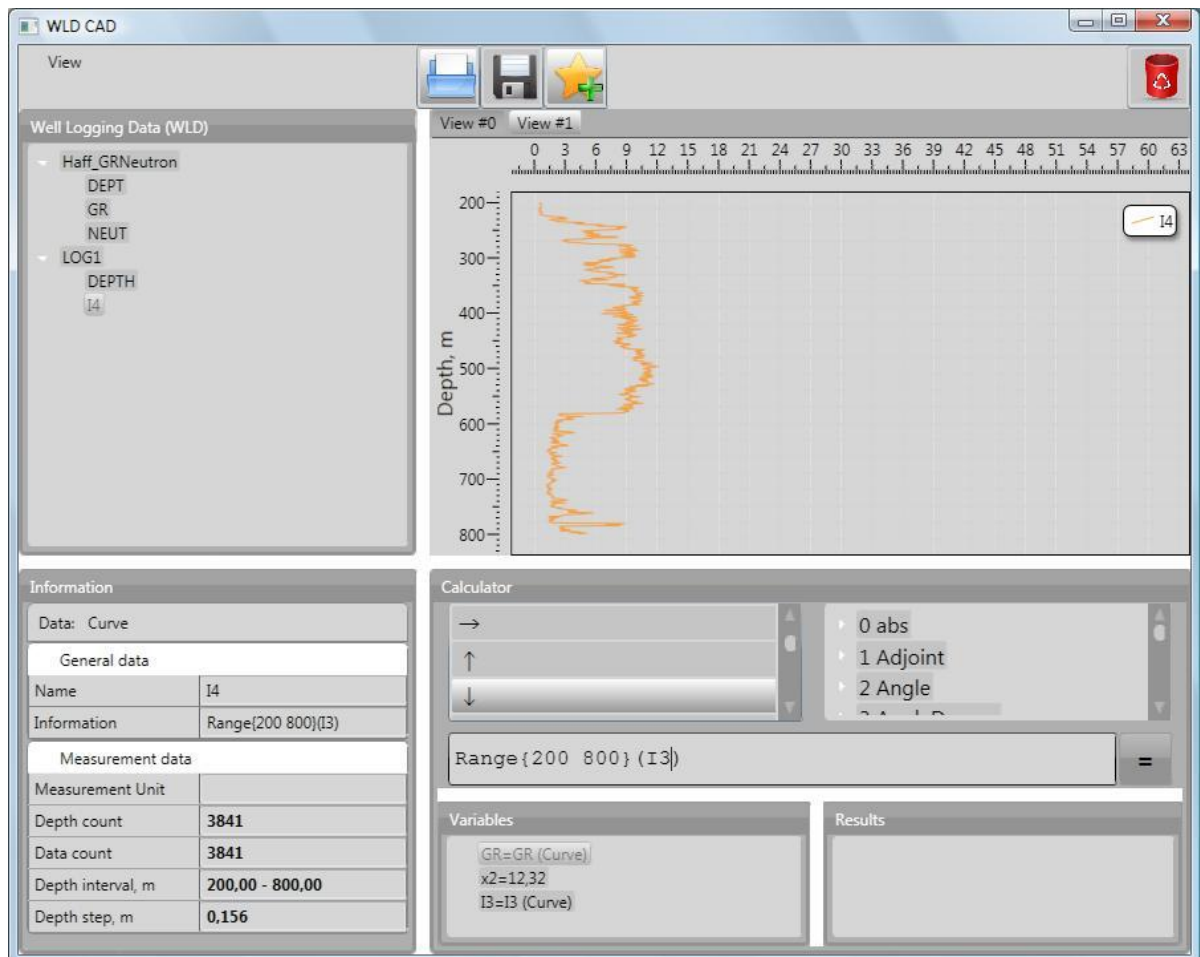
And finally, let the data range for processing must be in interval of 200 - 800 meters. Make the 'I3' variable by dragging it to the Variable panel and then input formula shown on the picture 14. The 'Range' function extracts the data inside the given interval (200 - 800 meters for the case). As can be seen from the picture 14, the result data 'I4' has only values in range of 200 - 800 meters. Then the Depth axis for new data can be extracted by 'Depth' function. The result data and depth axis can be added to the new Log (picture 14) and then saved to new file.



Picture 12. Calculated shift value.



Picture 13. Depth range scaled to right interval.



Picture 14. Result data.

Appendix I. Function reference

Table I.1. Curve specific functions.

Function Name	Variant	Parameters			Arguments			Result Type	Function summary
		#	Type	Description	#	Type	Description		
Convert	1	1	Unit	Unit to convert from	1	Channel	Data	Channel	Converts all channel data from one specified unit of measurement to another
		2	Unit	Unit to convert to	2	-	-		
FFT	1	1	-	-	1	Channel	Data	Channel	Fast Fourier Transform. When applied to 1D channel - the transform is over depth axis, for 2D data the transform applied for each depth signal.
		2	-	-	2	-	-		
FFTi	1	1	-	-	1	Channel	Data	Channel	Inversed Fast Fourier transform (see FFT).
		2	-	-	2	-	-		
Max	1	1	-	-	1	Channel 2D	Data	Channel 1D	Calculates maximum value of signal on each depth.
		2	-	-	2	-	-		
Max	2	1	-	-	1	Channel	Value 1	Channel	Calculates the maximum of two values
		2	-	-	2	Channel	Value 2		
Min	1	1	-	-	1	Channel 2D	Data	Channel 1D	Calculates minimum value of signal on each depth.
		2	-	-	2	-	-		
Min	2	1	-	-	1	Channel	Value 1	Channel	Calculates the minimum of two values
		2	-	-	2	-	-		
Range	1	1	Double	Top depth	1	Channel	Data	Channel	Extracts values from data channel on the specified depth interval.
		2	-	-	2	-	-		
	1	1			1				
		2			2				
	1	1			1				
		2			2				
	1	1			1				
		2			2				
	1	1			1				
		2			2				
	1	1			1				
		2			2				